МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
На тему: «Веб-додаток для розв'язання нетривіальних математичних задач з використанням алгоритмів оптимізації»
здобувача групи ІН – 91 Тєлєтова Дмитра Олександровича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Дмитро ТЄЛЄТОВ
(підпис)

Керівник,
доцент,
кандидат технічних наук Сергій ПЕТРОВ _____
(підпис)

Суми – 2023

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-91 Тєлєтова Дмитра Олександровича

1. Тема роботи: «Веб-додаток для розв'язання нетривіальних математичних задач з використанням алгоритмів оптимізації»
затверджено наказом по СумДУ від _____
2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Вибір засобів створення веб-додатка. 3) Розробка програмного забезпечення. 4) Аналіз використання розробленого веб-додатка.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	24.04.2023	
2	*Вибір засобів створення веб-додатка*	01.05.2023	
3	*Розробка програмного забезпечення*	29.05.2023	
4	*Аналіз використання розробленого веб-додатка*	31.05.2023	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	05.06.2023	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 119 стор., 33 рис., 3 додатки, 41 використане джерело

Обґрунтування актуальності теми роботи. Кваліфікаційна робота присвячена створенню веб-додатка, який дозволяє розв’язувати нетривіальні математичні задачі з використанням оптимізаційних алгоритмів. Тема є актуальною, оскільки завдання зі скорочення часу й обсягу розрахунків у процесі виконання тих чи інших математичних обчислень залишається в теорії алгоритмів важливою проблемою.

Об’єктом дослідження є процес розроблення веб-додатка для вирішення нетривіальних математичних задач з використанням алгоритмів оптимізації.

Метою роботи є створення веб-додатка для полегшення обчислення нетривіальних математичних операцій із застосуванням оптимізаційних алгоритмів, за допомогою яких можна суттєво прискорити виконання багатьох математичних функцій.

Методи дослідження – методи порівняльного аналізу, узагальнення, інструменти побудови математичних моделей, технології розробки програмного забезпечення.

Результати – розроблено сучасний веб-додаток з використанням алгоритмів оптимізації, які дозволяють суттєво зменшити обчислювальну складність певних математичних операцій. Додаток має зручний інтерактивний користувацький інтерфейс, високу продуктивність. Розроблений програмний продукт може бути застосований для проведення математичних обчислювань у точних та природничих науках, а також в освітній практиці для вирішення широкого кола математичних задач різного рівня складності.

ВЕБ-ДОДАТОК, НЕТРИВІАЛЬНІ МАТЕМАТИЧНІ ЗАДАЧІ, АЛГОРИТМИ ОПТИМІЗАЦІЇ, ШВИДКЕ ПЕРЕТВОРЕННЯ ФУР’Є, JAVA SCRIPT, REACT

ЗМІСТ

ВСТУП

Протягом останніх десятиріч одним з пріоритетних напрямів науково-технічного прогресу є розвиток методів та засобів інформатики та обчислювальної техніки, зокрема методів комп'ютерного розв'язання сучасних задач обчислювальної математики. «Обчислювальна математика стала основою для реалізації та комп'ютерного розрахунку методів математичного моделювання» [9 , с. 6]. Дана кваліфікаційна робота присвячена розробці веб-додатка для вирішення нетривіальних математичних задач з використанням алгоритмів оптимізації.

Актуальність роботи визначається тим, що сфера застосування методів комп'ютерного розв'язання практичних математичних задач сьогодні стрімко поширюється. Це зумовлює необхідність створення веб-застосунків, які значно полегшили б математичні обчислення та прискорили б їх. Скорочення часу й обсягу розрахунків у процесі виконання перетворень є актуальним завданням. Як зазначають автори книги «Introduction to Algorithms», «не кожна проблема, яка вирішується алгоритмами, має набір потенційних рішень, який легко ідентифікується» [28, с. 9]. Тому виникає необхідність розглянути, яким чином можна застосовувати алгоритми, наприклад перетворення Фур'є, для вирішення різних практичних завдань з мінізованою часовою складністю. «Пошук засобів і підходів для скорочення різних мікроітерацій є однією з важливих задач автоматизації процесів та покращення методик ідентифікації і реалізації програмних алгоритмів» [12 , с. 98].

Метою роботи є створення веб-додатка для полегшення обчислення нетривіальних математичних операцій та аналіз можливості застосування оптимізаційних алгоритмів, за допомогою яких можна суттєво прискорити виконання багатьох математичних функцій.

Для досягнення цієї мети необхідно виконати такі **завдання**:

1) здійснити огляд сучасних літературних джерел за тематикою роботи;

2) проаналізувати схожі застосунки, які розв'язують подібні завдання;

3) розглянути доступні й обрати оптимальні інструменти для розробки програмного забезпечення;

4) створити додаток із використанням алгоритмів оптимізації, що дозволить суттєво зменшити обчислювальну складність операцій.

Об’єкт дослідження – процес розроблення веб-додатка для вирішення нетривіальних математичних задач із застосуванням алгоритмів оптимізації

Предмет дослідження – методи й технології розробки математичного та програмного забезпечення, яке дозволить оптимально розв’язувати математичні задачі.

Гіпотеза дослідження ґрунтувалась на припущенні, що використання алгоритмів оптимізації, застосованих при створенні веб-додатка, дасть змогу значно скоротити час і обсяг розрахунків у процесі вирішення нетривіальних математичних задач.

Новизна отриманих результатів полягає у створенні сучасного програмного забезпечення, яке відрізняється дружнім до користувача інтерфейсом, на основі описаних у роботі математичних моделей, що суттєво зменшує обчислювальну складність розв’язання нетривіальних математичних задач.

Практична значущість одержаних результатів полягає в можливості використання розробленого програмного продукту для проведення математичних обчислювань у точних та природничих науках. Веб-додаток може також знайти застосування в освітній практиці при розв’язанні математичних задач різного рівня складності.

Апробація. Результати дослідження було апробовано на Міжнародній науковій конференції молодих учених «IMA-2023» (Суми – Астана, 24–28 квітня 2023 року) та висвітлено в науковій публікації: Тєлєтов Д. О. Можливості застосування швидкого перетворення Фур’є для розв’язання практичних задач у computer science. *Інформатика. Математика. Автоматика*. Суми, 2023. С. 83–84.

Структура. Дана робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У вступі обґрунтовується вибір теми, актуальність проблеми, зазначається гіпотеза, об’єкт, предмет, мета та завдання дослідження. У першому розділі здійснюється аналітичний огляд науково-практичних досліджень з використання математичного апарату в різних сферах життєдіяльності людини, характеризуються аналогічні програмні продукти, формулюється постановка задачі. У другому розділі обираються засоби створення веб-додатка, інструменти відображення математичних виразів, аналізується можливість застосування алгоритмів оптимізації. У третьому розділі дається опис програмної реалізації та використання веб-додатка. У висновках підбиваються підсумки та узагальнюються результати проведеної роботи.

Загальний обсяг роботи складає 119 сторінок, з них основного тексту – 57. Список використаних джерел містить 41 найменування.

1 АНАЛІЗ ДОСЛІДЖУВАНОЇ ПРОБЛЕМИ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Використання математичного апарату в різних сферах життєдіяльності людини

Три століття тому, як зазначав відомий британський програміст, математик та фізик Стівен Вольфрам, науку змінила драматична нова ідея про те, що правила, засновані на математичних рівняннях, можна використовувати для опису світу природи [41, с. 1]. З кожним роком математичні розрахунки використовуються в усіх сферах життєдіяльності людини все більше й більше. Сьогодні інформаційні технології дозволяють застосовувати математичний апарат для вивчення не тільки існуючих у природі об’єктів, явищ та процесів, а й штучних, тобто таких, що створені людиною.

Математизація науки є однією з найхарактерніших її особливостей на сучасному етапі розвитку. Математичний апарат застосовується в широкому спектрі наук про живе. На використанні математичних методів для описання біологічних систем базується, наприклад, така галузь біології, як біоматематика. Вона займається дослідженням та моделюванням біологічних процесів, перевіркою біологічних гіпотез вивчає чисельний аналіз фундаментальних проблем цих наук на підставі математичного моделювання біологічних процесів і їхніх досліджень математичними методами.

У таких науках, як екологія та епідеміологія, прості математичні моделі можуть застосовуватися для відображення і прогнозування динаміки популяцій. Так, наприклад, ще в XIII столітті італійський математик Леонардо Пізанський (більш відомий як Фібоначчі) встановив послідовність, яка описує ріст популяції кроликів. Це можна вважати одним з перших випадків використання математики для вирішення біологічних проблем. З часом дана послідовність, відома всім як числа Фібоначчі, стала застосовуватися й в інших науках, зокрема в ботаніці. Окрім теоретичної біології, математичні методи допомагають вирішити різні проблеми біофізики, біоінженерії, біоінформатики тощо.

Загалом у біології знаходять своє використання диференціальні рівняння та рівняння з частинними похідними, теорія збурень, математичній і функціональний аналіз, теорія динамічних систем, чисельні та нелінійні методи, теорія ймовірностей, комбінаторика, статистика, лінійна й абстрактна алгебра, теорія графів, теорія фракталів, алгебрична геометрія, топологія, теорія розпізнавання образів, теорія оброблення зображень та сигналів, теорія кодування [15]. Для розв'язання цих задач біологи можуть використовувати пакети комп'ютерного моделювання (Matlab, Wolfram Mathematica, Mapple та ін.).

У своїй практичній роботі математичний апарат постійно використовують і хіміки. За допомогою математичних методів вони визначають характеристики хімічного об'єкту, описують різні багатофакторні хіміко-технологічні процеси, обробляють результати дослідів, планують хімічний експеримент. Відзначимо, що перші спроби такого використання математичних методів були зроблені ще на початку 20-х років XX століття англійським математиком, генетиком та теоретиком-еволюціоністом Рональдом Фішером. Після 1951 р. методи оптимального планування дослідів набули подальшого поширення завдяки Дж. Боксу та К. Вілсону, праці яких сприяли прискоренню темпів розвитку теорії планування. Математичний інструментарій особливо часто використовується в таких нових галузях хімічної науки, як фізична хімія, хімічна термодинаміка й кінетика, квантова хімія тощо. Загалом при розв'язанні хімічних задач застосовуються теорія груп, функціональний аналіз, диференціальні рівняння, методи топології та диференціальної геометрії, теорія графів, теорія ймовірності тощо.

За допомогою математичних методів досліджуються також фізичні явища і процеси. Вперше методи математичної фізики були сформульовані ще в XVII столітті відомим англійським фізиком і математиком Ісааком Ньютоном. Подальший їх розвиток у XVIII ст. пов'язаний з іменами Жозефа-Луи Лагранжа, Жана ле Рона Д'Аламбера, П'єра-Симона Лапласа, Леонарда Ейлера, Готфріда

Лейбніца, Карла Фрідріха Гаусса, Якоба Бернуллі. Наступне століття характеризувалося інтенсивним вивченням рівнянь, які виникли в теорії теплопровідності, дифузії, хвильових процесів, стійкості руху, електродинаміці, гідродинаміці, газовій динаміці, оптиці (Жан Батист Фур'є, Сімеон-Дені Пуассон, Оґюстен Луї Коші, Жуль Анрі Пуанкаре, Жак Адамар, Петер Густав Лежен Діріхле, Бернгард Ріман, Густав Роберт Кірхгоф, Давид Гільберт, Джеймс Клерк Максвелл та інші). XX століття ознаменувалося науковими дослідженнями у квантовій фізиці, релятивістській механіці, астрофізиці, що призвело до формулювання низки нових рівнянь (Ервін Шредінґер, Людвіг Больцман, Вольфганг Паулі тощо), нових математичних методів, які дають змогу глибше зрозуміти природу об'єктів, що вивчаються, нових можливостей для фізичних досліджень ширшого кола природних явищ. «Тобто математика почала відігравати не лише роль інструмента для опису фізичних явищ, але й виявилося, що вона має здатність проникнення в їхню сутність» [5]. Отже, математика є не тільки засобом для проведення різноманітних обчислень, але й методом одержання нового знання.

Математичні методи досить часто застосовуються в економічних науках (економіці підприємства, бізнес-плануванні, менеджменті, маркетингу, логістиці та ін.). Керування економічними процесами може здійснюватися лише з використанням так званих економіко-математичних методів у будь-яких сферах господарства країни – від пошуку корисних копалин (кам'яного вугілля, залізної руди, нафти, газу тощо) до аналізу попиту як на промислові товари, так і на товари широкого вжитку, готельно-ресторанні та туристичні послуги, від дослідження потреби в робочій силі в тих чи інших галузях до планування логістичних ланцюжків та каналів розподілу кінцевої продукції тощо.

У наш час вивчення різних економічних явищ неможливе без використання математичних моделей, які враховують найсуттєвіші чинники того чи іншого явища, яке досліджується [1]. Застосування математики в прогнозуванні, плануванні, організації та управлінні економічною діяльністю дозволяє виділити

й описати за допомогою формул і математичних співвідношень важливі зв'язки між економічними змінними, суб'єктами та об'єктами економічної діяльності. Дослідження економіки, як і будь-якої складної системи, вимагає високого ступеня абстракції. Наприклад, завдяки методам математичної статистики можна отримати нові знання про економічний об'єкт, що аналізується, дати оцінку ступеня залежності між його змінними, якими описуються наявні спостереження.

Математичний апарат застосовується до економічних розрахунків, прогнозів, економічного аналізу, моделювання природничо-економічних процесів тощо. При розробці економіко-математичних моделей використовуються елементи лінійної та векторної алгебри, аналітичної геометрії, функції однієї або багатьох змінних, диференціальні та інтегральні числення, диференціальні рівняння тощо [14 с. 51]. Отже, використання математичних методів дозволяє розв'язувати ті чи інші практико-економічні завдання, аналізувати економічні явища, досліджувати конкретні економічні ситуації, створювати їх моделі, виявляти математичні залежності в існуючих економічних процесах сьогодення.

Вище наведене свідчить про нагальну потребу в створенні веб-ресурсів, які допомогли б ефективніше застосовувати математичний апарат і значно полегшили б математичні обчислення. Оскільки чисельні розрахунки потребують багато часу, актуальним є питання розробки ефективних програм, здатних розв'язувати математичні задачі та швидко здійснювати обчислення [13, с. 216]. Застосування математичних обчислень для одержання необхідної інформації за допомогою програмного забезпечення звільняє від рутинної роботи, дозволяє скоротити час тих чи інших розрахунків, полегшити їх проведення, уникнути помилок у них.

1.2 Характеристика аналогічних програмних продуктів

Останнім часом для математичних розрахунків було розроблено різноманітні засоби і програмні продукти, зокрема: калькулятори, системи комп'ютерної математики, статистичне програмне забезпечення, розв'язувачі конкретних математичних задач.

Серед них можна виділити:

WolframAlpha. Сервіс використовує широкий обсяг знань та має високу обчислювальну потужність. Він може допомогти розв'язати задачі з елементарної арифметики (дроби, відсотки, розрядні та текстові задачі), алгебри (знаходження коренів, розкладання на множники або спрощення математичних виразів – від поліномів до полів і груп, візуалізація функцій, рівнянь та нерівностей в одно-, дво- або тривимірному просторі, побудова полярних та параметричних графіків), математичного аналізу (обчислення інтегралів, похідних та границь, а також аналіз сум, добутків та рядів; розв'язання диференціальних рівнянь будь-якого порядку), комплексного аналізу (аналіз функцій та виразів, що містять уявні числа або комплексні змінні), тригонометрії (тригонометричні обчислення, дослідження властивостей тригонометричних функцій і тотожностей), лінійної алгебри (вектори, матриці і векторні простори), теорії чисел (аналіз цілих чисел, підмножин цілих чисел, наприклад простих чисел), дискретної математики (дослідження послідовностей та рекурентних співвідношень, вирішення загальних задач комбінаторики та обчислення властивостей графів і ґраток, оцінка булевих логічних виразів і виразів, що містять множини та оператори множин, розв'язання булевих рівнянь, обчислення таблиць істинності, створення діаграм Венна), геометрії (обчислення властивостей геометричних об'єктів різного типу у дво- або тривимірному просторі), теорії ймовірностей (обчислення ймовірності настання певних подій, спільних, непересічних або умовних ймовірностей), статистики (обчислення властивостей наборів даних, виконання статистичних висновувань або

моделювання даних, робота з розподілами ймовірностей та випадковими величинами) (рис 1.1).

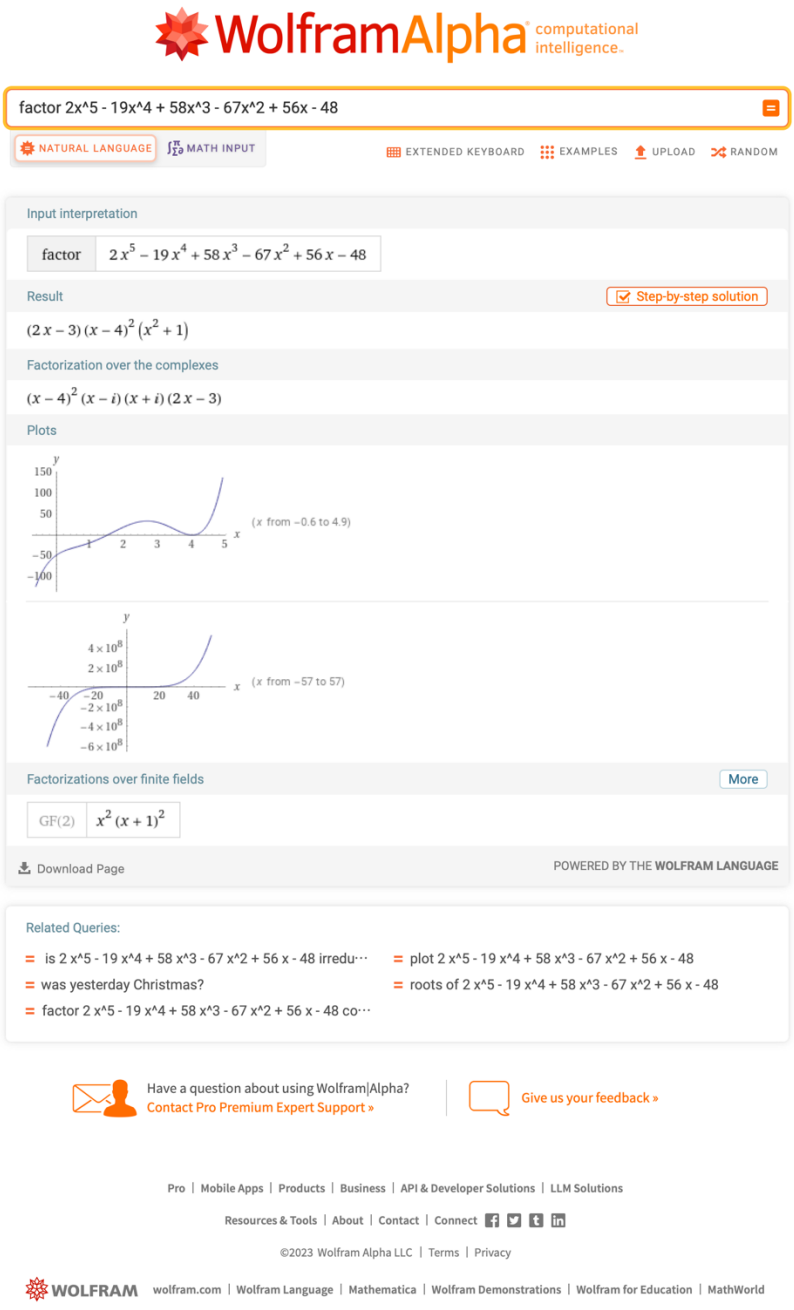

Рисунок 1.1 – Приклад використання сервісу WolframAlpha

WolframAlpha також може використовуватись при роботі з різними числами, для перевірки приналежності до більших множин, таких як раціональні чи трансцендентні числа, перетворення між системами числення; вивченні властивостей математичних функцій, таких як неперервність, сюр’єктивність і

парність, використанні відомих спеціальних функції або функції теорії чисел [40].

Сервіс впорається з пошуком інформації про відомі математичні проблеми, гіпотези, теореми та парадокси, дасть визначення математичних понять дозволить дізнатися про алгоритми тощо.

Натомість основним недоліком є машинно-орієнтованість додатка, внаслідок чого він є незручним та незрозумілим для звичайного користувача.

GeoGebra. Сервіс є динамічним геометричним середовищем, що використовується для підтримки вивчення та викладання математики, зокрема геометрії, алгебри та статистики. Функціональний потенціал, що надається Geogebra, передбачає, що він може бути чудовим медіа-сервісом, який допоможе користувачам швидко, точно та ефективно візуалізувати абстрактні геометричні об’єкти та маніпулювати ними [39]. Там можна знайти більше мільйона розробок, моделей, вправ, уроків та ігор з математики та інших наук. Ця комп’ютерна програма полегшує учням розуміння геометрії, тому особливо часто використовується при вивченні саме цього розділу математики. Приклад використання додатку проілюстровано на рисунку 1.2

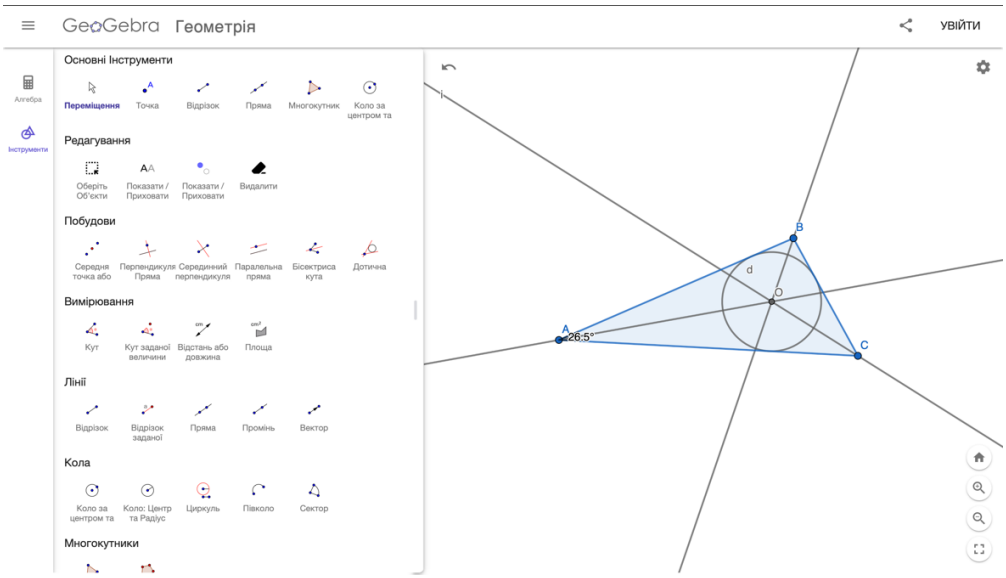

Рисунок 1.2 – Приклад використання додатку GeoGebra

Сервіс містить віртуальну платформу GeoGebra Classroom, за допомогою якої вчителі можуть створювати інтерактивні та захоплюючі завдання для

студентів, стежити за поточним прогресом учнів, переглядати, чи розпочали вони працювати над тим чи іншим завданням, ставити запитання як всьому класу, так і конкретному учню й отримувати їхні відповіді [26].

До основних недоліків можна віднести те, що він спеціалізується лише на графічному представленні та вирішенні математичних задач, що суттєво обмежує можливості його використання. Він часто застосовується як інструмент і засіб підтримки освіти. Зрозуміло, що для цього вчителям потрібні не тільки достатні знання, а ще й навички роботи з цим конкретним програмним забезпеченням.

Symbolab. Цей додаток використовується переважно як навчальний інструмент з математики. Він дозволяє користувачам вивчати різні математичні теми, практикуватися у розв'язанні математичних задач. Symbolab надає автоматизовані покрокові рішення різнорівневих завдань (алгебраїчних, тригонометричних та інших обчислень). Додаток пропонує безліч інтелектуальних калькуляторів, зокрема: рівняння, нерівності, функції, похідні, інтеграли, границі, дотичні лінії тощо (рис 1.3). Symbolab використовує алгоритми машинного навчання, щоб досягти заявленої мети сайту – зробити науковий контент доступним для пересічного користувача [38].

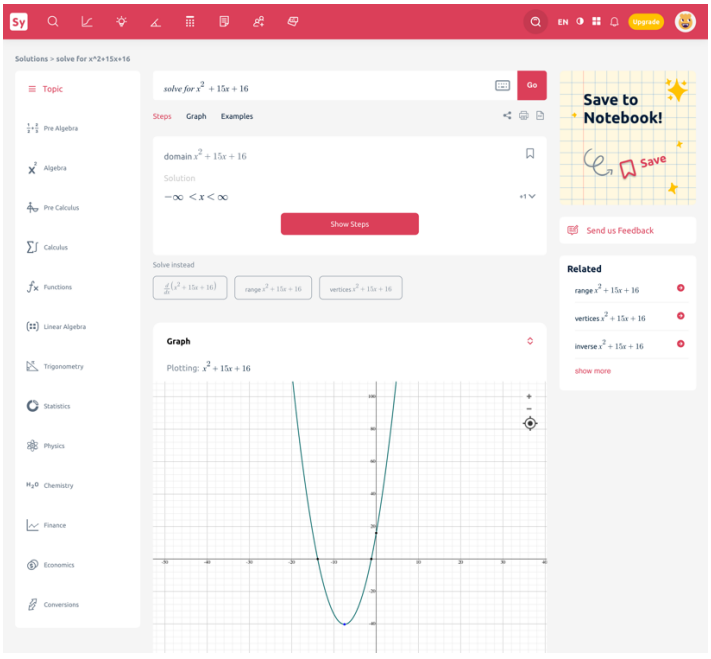

Рисунок 1.3 – Приклад використання веб-додатку Symbolab

Проте сервіс має не дуже широкий спектр можливостей застосування, здебільшого він орієнтований на використання в сфері освіти. У зв'язку з цим у 2020 році компанія, яка займається розробкою та підтримкою програмного забезпечення Symbolab, була придбана американським веб-сайтом Course Hero, що спеціалізується на освітніх технологіях.

Desmos. Даний застосунок являє собою прогресивний графічний калькулятор, який можна використовувати для побудови графіків різних функцій, зокрема: постійної функції, кусково-заданих функцій, функцій з параметром, складних функцій, графіків у полярній системі координат, а також залежності абсциси від ординати та нерівностей [24]. Приклад використання зображено на рисунку 1.4.

Сервіс також надає можливість зберігати створені графіки та отримувати постійне посилання на них, друкувати та експортувати їх зображення або HTML-код. Додаток має зручний інтерфейс, багато прикладів та детальну інструкцію для користувачів.

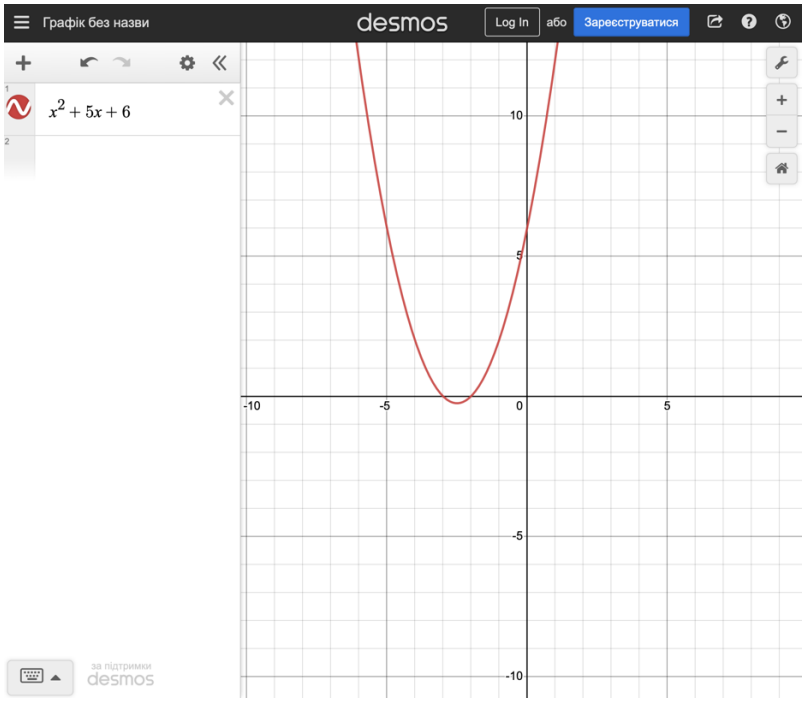

Рисунок 1.4 – Приклад використання веб-сервісу Desmos

Згодом функціональній потенціал був розширений. Були додані науковий, арифметичний, матричний калькулятори та геометричні інструменти. Але вони

поки що є не достатньо якісними й потребують подальшого удосконалення. Слід також відзначити, що Desmos, як і інші сервіси, має навчальну платформу (Desmos Classroom), яка пропонує учням різні види навчальної активності. Також вчитель або викладач може створювати свої власні завдання. Платформа може використовуватися як для аудиторної роботи, так і для дистанційного навчання.

Mathcad є програмним забезпеченням для вирішення задач інженерної математики. Додаток дозволяє вирішувати математичні завдання, аналізувати, документувати та ділитися своїми інженерними розрахунками. Mathcad являє собою комплексний сервіс, інтуїтивно зрозумілий для користувача. Програмний продукт не тільки виконує точні обчислення, але й дозволяє демонструвати свою роботу, застосовуючи різноманітні параметри форматування разом із сюжетами, текстом і зображеннями в одному документі. Тобто сервіс дозволяє документувати свої розрахунки в інженерному блокноті, при цьому використовуючи, окрім тексту, математичну нотацію, властиву різним наукам (математиці, фізиці, техніці, хімії, біології, економіці тощо). Візуальне представлення математики, за допомогою інтерфейсу користувача, схожого на зошит (рис. 1.5), полегшує розуміння та маніпулювання розрахунками [30].

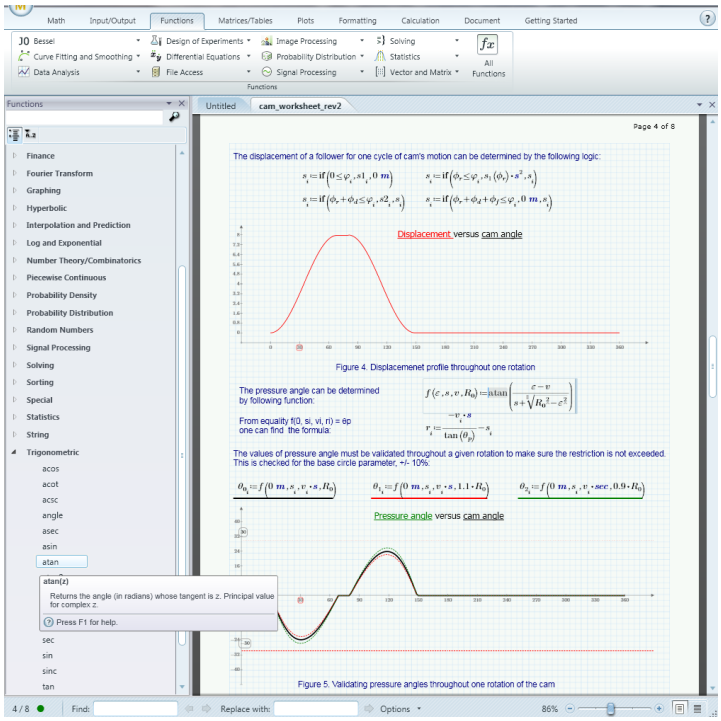

Рисунок 1.5 – Приклад використання веб-сервісу Desmos

До переваг цього програмного продукту слід віднести можливість його застосування для колективної роботи. Проте широкою функціональністю характеризується лише платна версія. Основним недоліком ми вважаємо те, що програмне забезпечення Mathcad є десктопним додатком, який необхідно встановлювати, що зручно не для всіх користувачів і не дозволяє використовувати його на мобільних пристроях.

Отже, проаналізувавши зазначені програмні продукти, можемо констатувати, що сьогодні існує достатньо різноманітних сервісів для проведення математичних обчислень. Кожний з них має свої переваги, проте жоден з них не позбавлений певних недоліків. Деякі з цих програм є десктопними, другі мають недоліки в інтерфейсах, треті громіздкі або не підтримуються. Незручність та незрозумілість для звичайного користувача (а робота з більшістю з описаних сервісів потребує певних знань та навичок), їх вузька спеціалізація, обмежений спектр можливостей застосування, недостатня швидкість розрахунків та ін. зумовлює актуальність розробки нового веб-додатка для розв’язання нетривіальних математичних задач з використанням алгоритмів оптимізації.

1.3 Постановка задачі

Аналітичний огляд сучасних літературних джерел та аналіз схожих застосунків дозволили дійти висновку, що задача з розробки веб-додатка для розв’язання математичних задач є актуальною. Це зумовило **мету** даного дослідження – створення веб-додатка для полегшення обчислення нетривіальних математичних операцій із застосуванням оптимізаційних алгоритмів, за допомогою яких можна суттєво прискорити виконання багатьох математичних функцій.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1) визначити вимоги, що висуваються до розроблюваного веб-додатка;

2) обрати оптимальні інструменти для створення програмного забезпечення;

3) розглянути алгоритми оптимізації та проаналізувати можливість їх застосування для прискорення швидкості обчислень;

4) визначити формат зберігання математичних даних;

5) розробити програмний код додатка;

6) перевірити працездатність створеного програмного продукту.

Окреслимо вимоги до нашого веб-додатка. Вони мають включати визначення підтримуваних математичних задач, необхідних функцій та можливостей додатка, інтерфейсу користувача тощо.

На нашу думку, розроблюваний додаток повинен мати такий функціонал:

– забезпечення можливості введення та редагування математичних виразів;
– підтримування різних типів математичних задач, зокрема: операції над поліномами, операції теорії чисел, побудова графіку функцій тощо;
– забезпечення точних та швидких результатів розв'язання математичних задач;

Окрім цього, при розробці веб-додатка потрібно врахувати такі вимоги до інтерфейсу користувача:

– зручність, легкість та простота використання додатка;
– інтуїтивно зрозуміле введення математичних виразів;
– наявність контекстної допомоги.

Додаток повинен бути надійним, безпечним та доступним, забезпечувати стабільну роботу без збоїв та помилок, обробляти помилкові ситуації та виводити зрозумілі повідомлення про помилки користувачу, а також мати можливість розширення функціональності для підтримки нових типів математичних задач.

2 МЕТОДИКА ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАДАЧ

2.1 Вибір засобів для створення веб-додатка

Для досягнення поставленої мети нам необхідно визначитися із типом програмного забезпечення, яке використовуватиметься. На нашу думку, для реалізації заявлених у постановці задачі функцій можна застосувати як десктопні (настільні) додатки, так і веб-застосунки.

Основною відмінністю веб-сервісів від настільних програм є те, що доступ до них здійснюється через веб-браузер з підтримкою Інтернету. Отже, користувачеві не потрібно їх завантажувати чи встановлювати. Для багатьох це стає вирішальним фактором при виборі того чи іншого програмного продукту.

Серед основних причин використання веб-програм науковці також називають кросплатформеність, легкість обслуговування, можливість розширення, доступність тощо [21]. Веб-програми можна застосовувати на будь-якій операційній системі (Windows, Linux, Mac), оскільки у цих платформ немає проблем з сумісністю із основними сучасними браузерами. Порівняно з настільними програмами, веб-програми легше підтримувати. Регулярні оновлення не потребують від користувача жодних додаткових дій, у тому числі повторних скачувань, тому в нагадуванні користувачам про оновлення своїх програм немає потреби [25]. До того ж доступ до необхідних даних можна здійснювати не лише через комп’ютер, а й з допомогою мобільного пристрою. Отже, якщо можуть виникати проблеми з підключенням до Інтернету, краще обрати настільний додаток. Веб-застосунок же є оптимальним варіантом, якщо потрібен доступ з будь-якого пристрою. Для вирішення поставлених у роботі задач вважаємо за необхідне обрати веб-версію.

Наступним кроком при виборі програмного інструменту став вибір типу веб-сервісу. З цією метою ми провели порівняння веб-сайту з веб-додатком.

Веб-сайт – це сукупність глобально доступних веб-сторінок, які пов’язані між собою і мають одне доменне ім’я. Сторінки сайту можуть містити контент різного характеру, зокрема: текстова інформація, зображення, відео, аудіо

тощо [19]. Веб-сайти мають різноманітне призначення. Бізнес-структурами вони використовуються для демонстрації власних товарів та послуг, брендування своїх торговельних та товарних марок, для підтримки клієнтів та заохочення потенційних споживачів, презентації того чи іншого рекламного продукту. В адміністративно-управлінській діяльності та соціальному менеджменті сайти застосовуються з метою інформування, юридичної підтримки, надання соціальних послуг тощо. Веб-сайт може розташовуватися на одному, двох або більше серверах. Доступ до сайту здійснюється через Інтернет або приватну локальну мережу. Характерними прикладами веб-сайту є сайти компаній, мас-медіа, блоги тощо. Такі ресурси є здебільшого статичними [22].

Веб-додаток – це повноцінне програмне забезпечення, доступне за допомогою будь-якого веб-браузера. Тобто це той самий веб-сайт, тільки динамічний, такий, що відрізняться широким функціоналом та має певний набір інтерактивних елементів. Працюючи з цим типом веб-сервісу, користувач має можливість маніпулювати певними даними. Загалом додаток є більш клієнтоорієнтованим.

Інтерфейс веб-додатка, так само як і сайту, створюється, як правило, за допомогою таких технологій, як HTML, CSS, JavaScript, які підтримуються основними браузерами, проте інтерактивність додатка детермінує більшу значущість та більш професійний рівень використання мови програмування JavaScript.

Для математичних обчислень, вирішення математичних задач інтерактивність є необхідною умовою, тому з веб-сервісів для виконання поставленого завдання найкраще підходить саме веб-додаток.

Обрання того чи іншого виду веб-застосунку залежить від того, для яких завдань він буде використовуватися. Існує три основні підходи до розробки веб-додатків: односторінкові (SPA), багатосторінкові (MPA) та прогресивні (PWA) [27]. Розглянувши їх переваги та недоліки, ми вирішили обрати SPA підхід, адже він дозволяє розробляти швидкі, динамічні веб-системи, створювати

кросплатформні застосунки, а отже, добре підходить для нашого проєкту. Архітектура односторінкової програми (SPA) побудована таким чином, що при перемиканні вкладок оновлюється не весь вміст, а лише його частина, тому немає потреби повторно завантажувати ті чи інші елементи. Це дуже зручно як для розробників, так і для користувачів. Односторінкові додатки відрізняються високою швидкістю завантаження, широким набором функцій, здатністю безперервно взаємодіяти з користувачем. Завдяки кешуванню даних SPA за потреби можуть працювати й в offline режимі, що підвищує зручність застосування веб-додатка й дає можливість користуватися ним у будь-яких умовах. Серед недоліків односторінкових додатків слід назвати погану SEO оптимізацію та трохи довше, ніж у звичайного сайту, завантаження під час першого відвідування, що не є суттєвим для нашого проєкту. Отже, SPA дозволяє створювати сучасні та конкурентоспроможні веб-програми, які характеризуються високою продуктивністю й кращим інтерактивним користувацьким досвідом. Для розробки SPA використовується JavaScript, що на сьогодні є однією з найпопулярніших мов програмування.

Основними фреймворками для реалізації SPA підходу є Angular, React та Vue. Серед них для реалізації нашого проєкту більше підходить саме React, тому що даний фреймворк відрізняється, по-перше, ефективністю та швидкістю роботи, яка забезпечується через застосування віртуального DOM та алгоритмів ефективного оновлення стану. Додаток, у якому використовується React, працює плавно та швидко навіть при зміні складних математичних виразів [17]. По-друге, React пропонує достатньо потужну систему компонентів, що дозволяє логічно поділити функціональність додатку на малі блоки, які можна повторно використовувати. Це дає змогу легко розширювати та модифікувати функціональність веб-додатку. React має велику кількість пакетів, бібліотек та розширень, що дозволяють інтегрувати інші інструменти.

2.2 Огляд інструментів відображення математичних виразів

Наступним кроком є вибір засобу для відображення математичних формул. Базові можливості HTML та CSS не повністю можуть задовольнити потреби у відображенні математичних виразів, рівнянь, формул. Тому довгий час найпоширенішим способом відображення математичної інформації на веб-сервісах було перетворення всіх математичних формул, які не можна описати звичайним текстом, у відповідні рисунки, тому що формули зі спеціальними символами при передачі їх засобами HTML-стандарту втрачають свій природний вигляд. [16]

Слід зауважити, що HTML-стандарт включає розмітку MathML, але підтримка всіма основними браузерами з'явилася досить недавно (у січні-березні 2023 року), до цього розмітка підтримувалась лише Firefox та Safari [31]. До того ж одна й та сама формула може виглядати неоднаково в різних браузерах, що може ускладнити розробку.

Тому сьогодні здебільшого користуються спеціалізованою мовою програмування TeX (від грецьк. τέχνη – «мистецтво», «майстерність»), яка була створена відомим ідеологом програмування Дональдом Кнутом саме для набору текстів, що містять математичні вирази та формули. Фактично ця система комп'ютерної верстки є золотим стандартом оформлення наукових публікацій з математики та суміжних галузей науки [11, с. 4].

На базі TeX американським дослідником у галузі математики та інформатики Леслі Лампортом було розроблено комп'ютерну видавничу систему LaTeX. Серед її переваг слід назвати гнучкість та зручність засобів для створення документів та можливість працювати на будь-якому комп'ютері, незалежно від його потужності та операційної системи [10, с. 5].

Оскільки LaTeX є лише системою верстки, то для його використання у веб-додатку необхідні відповідні пакети. Такими пакетами є, наприклад, MathJax, KaTeX, MathQuill тощо. Серед них можна виділити KaTeX, тому що він відрізняється:

- простотою API;
- автономністю (не має залежностей);
- швидкістю (KaTeX виконує математичні обчислення синхронно й не потребує перекомпонування сторінки);
- можливістю візуалізації на стороні сервера (KaTeX створює однаковий результат незалежно від браузера чи середовища, що дозволяє попередньо відобразити вирази за допомогою Node.js і надіслати їх як звичайний HTML) [29].

2.3 Аналіз можливості застосування алгоритмів оптимізації

З метою прискорення математичних обчислень нам необхідно обрати алгоритми, які ми зможемо застосувати при розробці веб-додатка. Розглянемо можливість їх використання детальніше.

Одним з таких алгоритмів є перетворення Фур'є. Воно являє собою сукупність математичних інструментів, що застосовують у різноманітних галузях науки: фізиці, геометрії, комбінаториці, статистиці, теорії ймовірності, криптографії тощо. Воно передбачає отримання коефіцієнтів («амплітуд») при розкладанні вихідної функції на елементарні складові – гармонійні коливання з різними частотами. Цей метод був відкритий французьким математиком Жаном Батистом Фур'є, який використав його для опису механізму теплопровідності. Згодом вченим, зокрема Клоду Нав'є, Софі Жермен та ін., вдалося розширити сферу досліджень, вивівши їх за межі теорії теплопровідності. Загалом ці дослідження проводилися протягом двох століть, у результаті чого була остаточно сформована дана теорія.

У наш час вивчення зазначеного питання зводиться до виявлення ефективних методів, що дозволяють перейти від функції до її перетвореного вигляду (пряме перетворення Фур'є) і назад (обернене перетворення Фур'є).

Для полегшення розрахунків таких перетворень потрібне спеціальне програмне забезпечення. Найрозповсюдженішим серед алгоритмів прискореного обчислення дискретного перетворення Фур'є є метод Кулі-Тьюкі,

відомий як «швидке перетворення Фур'є». Цей алгоритм обчислює ДПФ за час $O(n \log n)$, що значно краще за час $O(n^2)$, який досягається тривіальним алгоритмом множення.

Швидке перетворення Фур'є дозволяє прискорити процес за рахунок поділу вектора коефіцієнтів на два вектори, рекурсивного обчислення ДПФ для них й об'єднання результатів в одне ДПФ. Зазначений метод був описаний ще в 1965 році в роботі Джеймса Кулі та Джона Тьюкі «An Algorithm for the Machine Calculation of Complex Fourier Series» [23]. Він дає змогу при проведенні розрахунків суттєво заощаджувати час.

Із поширенням сучасних комп'ютерів метод швидкого перетворення Фур'є знайшов широке застосування. Сьогодні реалізації ШПФ є практично в усіх математичних пакетах і бібліотеках. Протягом останніх п'ятдесяти років було чимало спроб підвищити його ефективність. Однією з таких спроб є FFTW, що являє собою бібліотеку підпрограм C для обчислення дискретного перетворення Фур'є (ДПФ) в одному або кількох вимірах, вхідних даних довільного розміру, як дійсних, так і комплексних (а також парних / непарних даних, тобто дискретних косинусних / синусних перетворень, або DCT/DST). Пакет FFTW був розроблений у MIT Маттео Фріго та Стівеном Г. Джонсоном. Контрольні тести, проведені на різних платформах, показують, що продуктивність FFTW зазвичай перевершує продуктивність іншого загальнодоступного програмного забезпечення FFT і навіть конкурує з кодами, налаштованими постачальниками. Проте, на відміну від них, продуктивність FFTW є портативною: одна й та сама програма може добре працювати на більшості архітектур без змін. Звідси й дещо ексцентрична назва «FFTW», що означає «найшвидше перетворення Фур'є на Заході».

У 2012 році на симпозіумі з дискретних алгоритмів ACM дослідники з MIT, зокрема Х. Хассаніе., П. Індик, Д. Катабі, Е. Прайс, представили новий алгоритм швидкого перетворення Фур'є – sFFT (Sparse Fast Fourier Transform) [33]. Автори нового алгоритму зазначають, що він працює швидше за FFTW, а при виконанні

деяких задач може бути в десятки або навіть у сотні разів швидшим за класичний метод ШПФ. SFFT створений на основі двох існуючих фільтрів (фільтр Гаусса та фільтр Чебишева) і націлений на те, щоб швидко знайти фрагменти з «розрідженим» сигналом (sparse signal) та визначити вихідну амплітуду в кожному з них. Сигнал розбивається на фрагменти (rapid sampling) доти, доки залишиться розріджений сигнал з єдиною амплітудою. А вже там новий алгоритм виявляє її в 10 тисяч разів швидше за класичний ШПФ.

Такий спосіб не є універсальним, і сьогодні науковці намагаються визначити, конкретно в яких програмах збільшення його продуктивності виявиться найзначнішим.

Одним з напрямів сучасних досліджень є виявлення можливостей застосування швидкого перетворення Фур'є для вирішення практичних завдань у *computer science*. Так, наприклад, сьогодні активно вивчаються методи, які дозволяють зменшувати час реалізації операції множення надвеликих чисел для системи захисту інформації [2].

Проаналізуємо можливість використання алгоритму обчислення дискретного перетворення Фур'є для вирішення поставленої в роботі задачі.

Спочатку розглянемо дискретне перетворення Фур'є. Воно представлено у формулі 2.1.

$$y_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}nk} \tag{2.1}$$

Для спрощення запису позначимо $\omega_n = e^{-\frac{2\pi i}{n}}$, тоді перетворення матиме вигляд (формула 2.2):

$$y_k = \sum_{n=0}^{N-1} x_n \omega_N^{nk} \tag{2.2}$$

Позначимо дискретне перетворення Фур'є як DFT (формула 2.3):

$$y_k = DFT(x_n, N) = \sum_{n=0}^{N-1} x_n \omega_N^{nk} \tag{2.3}$$

Також будемо використовувати обернене перетворення Фур'є (формули 2.4 та 2.5), позначимо його як IDFT (формула 2.6)

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} y_k e^{\frac{i2\pi}{N} nk} \tag{2.4}$$

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} y_k \omega_N^{-nk} \tag{2.5}$$

$$x_n = IDFT(x_n, N) = \frac{1}{N} \sum_{k=0}^{N-1} y_k \omega_N^{-nk} \tag{2.6}$$

Для розрахунку кожного з N елементів треба обчислити суму, яка складається з N елементів, тому складність даного обчислення можна оцінити як $O(n^2)$. Дана часова складність є досить великою, що робить недоцільним використання звичайного ДПФ у більшості випадків.

Далі спробуємо отримати швидку реалізацію дискретного перетворення Фур'є, що ймовірно значно скоротить час обчислення.

Аналіз швидкої реалізації ДПФ (алгоритм Кулі-Тьюкі)

Усі застосування ДПФ значною мірою залежать від наявності швидкого алгоритму для обчислення дискретних перетворень Фур'є та їх обернених. Розглянемо цей алгоритм детальніше.

Розіб’ємо ДПФ (формула 2.2) на суми парних та непарних членів та зробимо подальші перетворення (2.7):

$$
\begin{aligned}
&y_k = \sum_{n=0}^{N-1} x_n \omega_N^{nk} = \\
&= \sum_{m=0}^{N/2-1} x_{2m} \omega_N^{k2m} + \sum_{m=0}^{N/2-1} x_{2m+1} \omega_N^{k(2m+1)} = \\
&= \sum_{m=0}^{N/2-1} x_{2m} \omega_{N/2}^{km} + \omega_N^k \sum_{m=0}^{N/2-1} x_{2m+1} \omega_{N/2}^{km} \Longrightarrow \\
&\Longrightarrow y_k = \sum_{m=0}^{N/2-1} x_{2m} \omega_{N/2}^{km} + \omega_N^k \sum_{m=0}^{N/2-1} x_{2m+1} \omega_{N/2}^{km}
\end{aligned}
\tag{2.7}
$$

Кожен з доданків є дискретним перетворенням Фур’є для наборів розміром $N/2$. Необхідно зазначити, що N повинно бути степенем двійки, інакше на якомусь рівні ітерації N не зможе поділитися націло на 2. Для $N = 1$ розбити вираз на два доданки не вийде, але можна обчислити за вихідною формулою (2.8):

$$
y_k = \sum_{n=0}^{1-1} x_n \omega_1^{nk} = x_0 \omega_1^{0 \cdot 0} = x_0 \tag{2.8}
$$

Отже, загальна формула (2.9) для підрахунку швидкого перетворення Фур’є виглядає так:

$$
\begin{cases}
DFT(x_n, N) = DFT(x_{2m}, N/2) + \omega_N^k DFT(x_{2m+1}, N/2), N > 1 \\
DFT(x_0, 1) = x_0, N = 1
\end{cases}
\tag{2.9}
$$

Відповідно отриманій формулі застосуємо принцип «розділяй і володарюй» (англ. *divide and conquer*), який полягає в рекурсивному розбитті задачі, що розв'язується, на декілька схожих підзадач, які мають менший розмір, і об'єднанні їх розв'язків [28, с. 65]. Щоб порахувати значення y_k у N точках x_n, порахуємо значення рекурсивно в $N/2$ точках x_{2m} та $N/2$ точках x_{2m+1} та скомбінуємо їх. Тепер значення y_k відновити досить просто.

Час роботи виражається рекурентною формулою (2.10).

$$T(n) = 2T(n/2) + O(n) \tag{2.10}$$

Це досить відоме співвідношення, яке розкривається в $O(n \log n)$. Тобто глибина рекурсії – $\log_2 n$ рівнів, на кожному рівні виконується $O(n)$ операцій.

Застосування ШПФ для множення поліномів

Розглянемо можливість застосування швидкого перетворення Фур'є для множення поліномів. Поліном – це скінченна формальна сума виду (2.11):

$$\sum_{j=0}^{n} a_j x^j \tag{2.11}$$

Добуток поліномів, як відомо, є також поліномом. Степінь добутку поліномів дорівнює сумі степенів співмножників (формула 2.12).

$$\left(\sum_{j=0}^{n} a_j x^j\right) \cdot \left(\sum_{j=0}^{m} b_j x^j\right) = \left(\sum_{j=0}^{n+m} \sum_{l+k=j} a_l b_k x^j\right) \tag{2.12}$$

Тепер, для того щоб застосувати ШПФ, по-перше, два поліноми слід привести до одного степеня, доповнивши коефіцієнти одного з них нулями. По-

друге, у результаті добутку двох поліномів степеня n виходить поліном степеня $2n$, тому, щоб результат вийшов коректним, попередньо потрібно подвоїти степені кожного полінома, також доповнивши їх нульовими коефіцієнтами. По-третє, для використання саме ШПФ треба привести поліноми до степеня, який рівняється степеню двійки, таким же способом. Слід відзначити, що доповнення нульовими коефіцієнтами жодним чином не змінює поліном. Позначимо новий спільний степінь усіх поліномів N. Тепер задача зводиться до аналізу можливості застосування дискретного перетворення Фур'є.

Припустимо, дано два поліноми $A(x)$ і $B(x)$. Порахуємо ДПФ для кожного з них: $DFT(A)$ і $DFT(B)$ – це два вектори-значення багаточленів.

Вектор A:

$$DFT(A) = \sum_{n=0}^{N-1} a_n \omega_N^{nk} \tag{2.13}$$

Вектор B:

$$DFT(B) = \sum_{n=0}^{N-1} b_n \omega_N^{nk} \tag{2.14}$$

Тепер застосуємо ДПФ до множення многочленів:

$$DFT(A \times B) = DFT\left(\sum_{p+q=j} a_p b_q\right) = \sum_{j=0}^{N-1}\left(\sum_{p+q=j} a_p b_q\right)\omega_N^{jk} \tag{2.15}$$

Поелементно перемножимо ДПФ для кожного з поліномів та порівняємо

$$
\begin{aligned}
&DFT(A) \times DFT(B) = \\
&= \left(\sum_{p=0}^{N-1} a_p \omega_N^{pk} \right) \times \left(\sum_{q=0}^{N-1} b_q \omega_N^{qk} \right) = \\
&= \sum_{p=0}^{N-1} a_p \omega_N^{pk} \sum_{q=0}^{N-1} b_q \omega_N^{qk} = \\
&= \sum_{p=0}^{N-1} \sum_{q=0}^{N-1} a_p b_q \omega_N^{(p+q)k} = \\
&= \sum_{j=0}^{N-1} \left(\sum_{p+q=j} a_p b_q \right) \omega_N^{jk} + \sum_{\substack{p+q \geq N \\ p<N \\ q<N}} a_p b_q \omega_N^{(p+q)k} = \\
&= DFT(A \times B) + \sum_{\substack{p+q \geq N \\ p<N \\ q<N}} a_p b_q \omega_N^{(p+q)k}
\end{aligned}
\tag{2.16}
$$

Через те, що поліном був доповнений до степеня N нулями, $a_j = 0$, $b_j = 0$, при $j \geq \frac{N}{2}$, тому $a_p b_q = 0$, при $p + q \geq N$. Тоді можна зробити висновок (2.17) щодо тотожності виражень (2.15) та (2.16).

$$
DFT(A) \times DFT(B) = DFT(A \times B) + \sum_{\substack{p+q \geq N \\ p<N \\ q<N}} 0 \cdot \omega_N^{(p+q)k} = DFT(A \times B) \tag{2.17}
$$

Тобто це означає, що якщо ми перемножимо вектори $DFT(A)$ і $DFT(B)$, просто помноживши кожен елемент одного вектора на відповідний елемент

другого вектора, то ми отримаємо не що інше, як ДПФ від багаточлена $A \times B$ (формула 2.18):

$$DFT(A \times B) = DFT(A) \times DFT(B) \tag{2.18}$$

Зрештою, застосовуючи обернене ДПФ, отримуємо формулу 2.19:

$$A \times B = IDFT(DFT(A) \times DFT(B)) \tag{2.19}$$

У формулі (2.19) під добутком двох ДПФ розуміють попарні добутки елементів векторів. Такий добуток, очевидно, вимагає обчислення тільки $O(n)$ операцій. Таким чином, якщо обчислити ДПФ та обернене ДПФ за час $O(n \log n)$ (що відповідає асимптотичній складності ШПФ), то й добуток двох поліномів можна знайти за ту ж асимптотику.

Можна зробити висновок, що використання ШПФ для множення поліномів можливе за формулою 2.19 і доцільне, оскільки дозволяє скоротити час обчислення з $O(n^2)$ до $O(n \log n)$.

Отже, успішна реалізація алгоритму дасть можливість значно економити час обчислень із поліномами та довгими числами.

Окрім перетворення Фур'є, можна застосувати й інші алгоритми. Розглянемо можливість прискорення обчислення n-го елемента послідовності Фібоначчі, яка є однією з найбільш відомих числових послідовностей. Як зазначає Н. М. Василенко, «числа Фібоначчі виникають у результаті розв'язання багатьох математичних задач і тісно пов'язані з не менш відомим «золотим відношенням»» [3, с. 9].

Послідовність Фібоначчі (або числа Фібоначчі) – це послідовність, яка виражається рекурентним співвідношенням (2.20)

$$F_n = F_{n-1} - F_{n-2} \tag{2.20}$$

з початковими значеннями $F_0 = 0$ та $F_1 = 1$ (або $F_1 = 1$ та $F_2 = 1$).

Числа Фібоначчі можуть бути репрезентовані в такому матричному вигляді (2.21):

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}. \tag{2.21}$$

Для доведення цього виразу використаємо метод математичної індукції. База індукції (2.22):

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^1 = \begin{pmatrix} F_2 & F_1 \\ F_1 & F_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}. \tag{2.22}$$

Перехід індукції: припустимо, що (2.23)

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^k = \begin{pmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{pmatrix}, \tag{2.23}$$

тоді (2.24)

$$\begin{aligned} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{k+1} &= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{pmatrix} = \begin{pmatrix} F_{k+1} + F_k & F_k + F_{k-1} \\ F_{k+1} & F_k \end{pmatrix} \\ &= \begin{pmatrix} F_{k+2} & F_{k+1} \\ F_{k+1} & F_k \end{pmatrix}, \end{aligned} \tag{2.24}$$

що й потрібно було довести.

Отже, згідно з формулою (2.22) для знаходження n-го числа Фібоначчі достатнього обчислити $\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1}$, шуканим числом Фібоначчі буде елемент на перетині першого рядка та першого стовбця.

Але щоб отримати виграш у часі, необхідно прискорити піднесення матриці до степеня. Цього можна досягнути, використавши алгоритм повторюваного піднесення до квадрата. Асимптотична складність даного алгоритму $O(\log n)$.

Отже, вище викладене дає змогу дійти висновку, що описані алгоритми оптимізації можуть бути використані при розробці веб-додатка для розв’язання нетривіальних математичних задач з метою пришвидшення розрахунків.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Визначення формату зберігання математичних виразів

Оскільки додаток повинен надавати користувачу можливість зберігати математичні вирази, то з’являється необхідність описати формат файлу, який буде використовуватися з цією метою. На роль бази для нього більше за інших підходить текстовий формат JSON. Він був створений спеціально для мови JavaScript (тому його буде легко застосувати) і широко використовується для обміну даними, в основному між сервером та браузером або між двома серверами.

Розроблений формат обов’язково повинен містити таку інформацію про математичний об’єкт:

– тип математичного виразу (тільки для верхнього рівня вкладеності) або назва математичного оператора;
– операнди, на які поширюється дія оператора, або інша інформація, яка точно відповідає типу математичного виразу (табл. 3.1). Операнди, у свою чергу, теж є математичними виразами, що зумовлює ієрархічність структури.

Таблиця 3.1 Формат запису математичних операторів

Тип	**Поля (окрім спільного поля 'type')**	**Значення поля**	**Приклад**

Унарний математичний оператор	value	Об’єкт математичного виразу, що відповідає операнду	{
			name: 'sin',
			value: {…}
			}

Бінарний математичний оператор	left	Об’єкт математичного виразу, що відповідає лівому операнду	{
			name: 'add',
			left: {…},
			right: {…}
			}

	right	Об’єкт математичного виразу, що відповідає правому операнду	

Змінна	variable	Літера або рядок, що ідентифікує змінну	{ name: 'variable', variable: 'x' accent: 'hat', index: 1 }
	accent	Спеціальні символи над змінною	
	index	Індекс для пронумерованих змінних (x_1, x_2 і. т. д.)	
Число	number	Число, що відповідає формату JSON	{ name: 'number', number: 3.14 }
Оператор порівняння	left	Об’єкт математичного виразу, що відповідає лівому операнду	{ name: 'less', left: {…}, right: {…} }
	right	Об’єкт математичного виразу, що відповідає правому операнду	
Поліном	value	Масив коефіцієнтів поліному	{ name: 'polynomial', value: […] }

У результаті отримаємо свій формат на основі JSON, який можна зручно використовувати в JavaScript для опису математичних виразів, їх зберігання, маніпулювання, обміну і навіть створення файлів вручну.

3.2 Опис програмної реалізації

Першим кроком створення програмного забезпечення було встановлення платформи Node.JS. Ця програмна платформа призначена для виконання JavaScript коду поза браузером. У нашому випадку вона виконує роль веб-сервера для фреймворку React.js. Далі за допомогою пакетного менеджеру було створено новий React-проєкт.

Потім потрібно було встановити необхідні бібліотеки. Ми обрали такі:

– **react-router-dom** – для управління навігацією;

– **@mui/material, @emotion/react, @emotion/styled, @mui/icons-material** – складові бібліотеки Material UI, яка спрощує імплементацію

дизайну (реалізує Material Design від Google), являє собою потужний набір готових компонентів для React [32];

- **katex** – для відображення математичних формул;
- **react-dnd, react-dnd-html5-backend** – для створення елементів, які можна перетягувати мишкою (drag and drop);
- **yup** – для валідації даних;
- **complex.js** – для роботи з комплексними числами;
- **function-plot** – для побудови графіка функцій;
- **react-color** – для вибору кольору;
- **copy-to-clipboard** – для зручного копіювання тексту в буфер обміну;
- **file-saver** – для зручного скачування файлів;
- **html-to-image** – для перетворення html-розмітки в зображення.

Усі ці бібліотеки містяться в папці node_modules, яка є обов'язковим елементом структури будь-якого проєкту на Node.JS.

Далі необхідно було розробити класи, які відповідають описаному раніше формату (додаток А). Усі класи успадковуються від базового класу *Operator*, який містить спільні методи для всіх операторів. Від нього успадковуються класи *UnaryOperator* (для операторів з одним операндом), *BinaryOperator* (для операторів з двома операндами), *NullaryOperator* (для опису сутностей, які не мають операндів, наприклад чисел, математичних змінних тощо). Від цих класів вже успадковуються усі класи, що відповідають сутностям, які використовуються в обраному форматі зберігання виразів. Класи операторів містять серед інших методи для створення tex-нотації виразу, формування необхідної для побудови графіку інформації тощо.

Наступним кроком є реалізація алгоритмів (додаток Б), зокрема алгоритмів оптимізації, описаних у розділі 2. Підрахунок швидкого перетворення Фур'є та оберненого до нього було реалізовано у вигляді однієї функції через мінімальну різницю між ними. А для збільшення ефективності роботи алгоритму було вирішено відмовитися від рекурсії та проводити обчислення, не створюючи

тимчасових масивів, тобто без використання додаткової пам’яті. Для цього необхідно було застосувати алгоритм побітової перестановки (*bit-reversal permutation*). Далі швидке перетворення Фур’є імплементовано у функцію множення поліномів, яка приймає два масиви, що відповідають коефіцієнтам поліномів, доповнює їх нулями до розміру необхідного для використання ШПФ, виконує пряме перетворення, поелементне множення, обернене перетворення відповідно до описаного у розділ 2 алгоритму.

Також було реалізовано й інші операції поліноміального обчислення: сума, різниця, ділення, ділення з остачею; створено функції для матричного множення, швидкого піднесення до степеню, обчислення n-го числа Фібоначчі та їх суми за допомогою піднесення відповідної матриці до степеню; застосовано алгоритм Евкліда для обчислення найбільшого спільного дільника та найменшого спільного кратного, скорочення дробів.

Потім було створено необхідні для графічного інтерфейсу React-компоненти, що дозволяють розбити його на незалежні части, які можна повторно використовувати (додаток В). Усі компоненти оголосили як функції, що є більш сучасними підходом. Після 2018 року з виходом нової версії React з підтримкою хуків функціональні компоненти отримали можливість використовувати майже весь функціонал класових, а їх більша простота зумовила поширення саме цього підходу в розробці SPA-додатків.

При створенні програмного забезпечення було застосовано такі React-хуки, як:

- useState – для додавання змінної управління станом до компоненту;
- useEffect – для управління віджетом, відмінним від React;
- useMemo – для кешування результату обчислень між повторними рендерами компоненти;
- useCallback – для кешування оголошення функції між повторними рендерами компоненти;
- useContext – для глибокої передачі даних у дерево компонентів;

– useRef – для управління DOM, за допомогою посилання.

У результаті виконаної розробки був створений веб-додаток із сучасним користувацьким інтерфейсом, який виконує усі заявлені в постановці задачі функції.

3.3 Використання розробленого веб-додатка

При запуску програми користувач спостерігає стартову сторінку програми, на якій представлено навігаційні блоки (рис. 3.1–3.2).

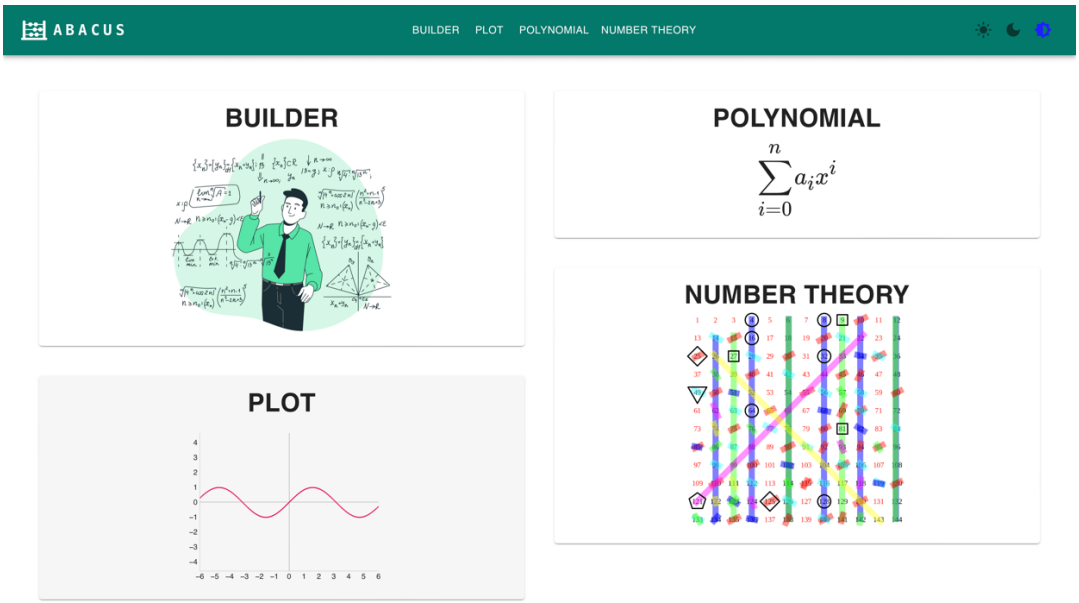

Рисунок 3.1 – Головна сторінка та «шапка» сайту (світла тема)

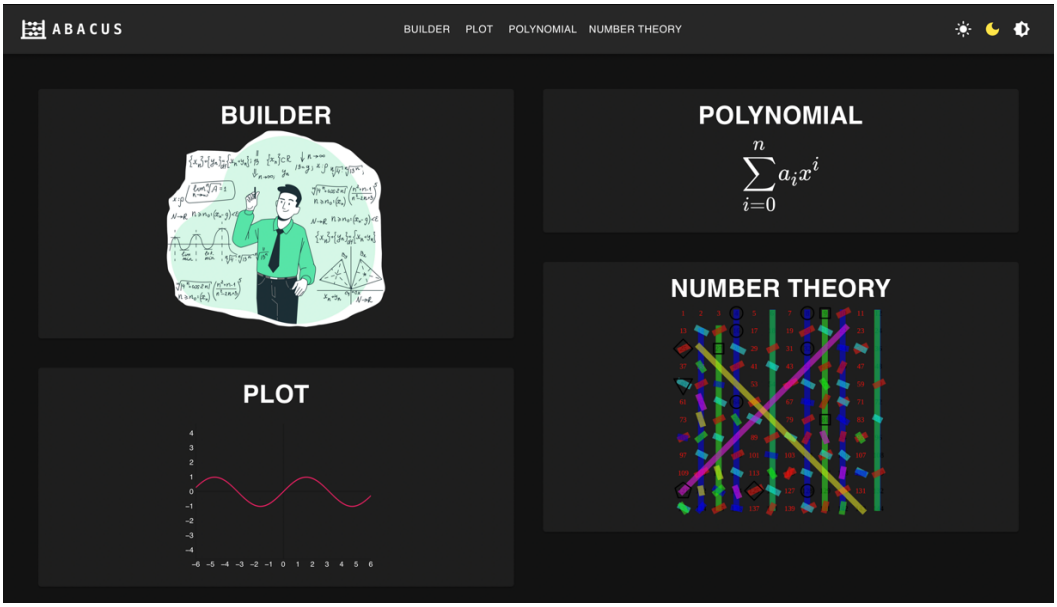

Рисунок 3.2 – Головна сторінка та «шапка» сайту (темна тема)

Проілюстрована навігація дає змогу перейти на такі сторінки веб-додатка: сторінку конструктора виразів (*builder*), поліноміальних операцій (*polynomial*), операцій теорії чисел (*number-theory*) та побудови графіків (*plot*). Також меню навігації знаходиться в «шапці» сайту. Її можна побачити на кожній із сторінок веб-додатка. Окрім навігаційної панелі, у «шапці» репрезентовані логотип додатка, кнопки зміни теми (рис. 3.1–3.2). Логотип також може використовуватися для навігації на головну сторінку. За допомогою кнопок зміни теми можна обрати світлу, темну або автоматичне налаштування теми в залежності від часу доби, і цей вибір зберігається в браузері на наступні сесії роботи з додатком.

На сторінці *builder* представлено список математичних виразів та інструменти для їх створення (рис. 3.3–3.4).

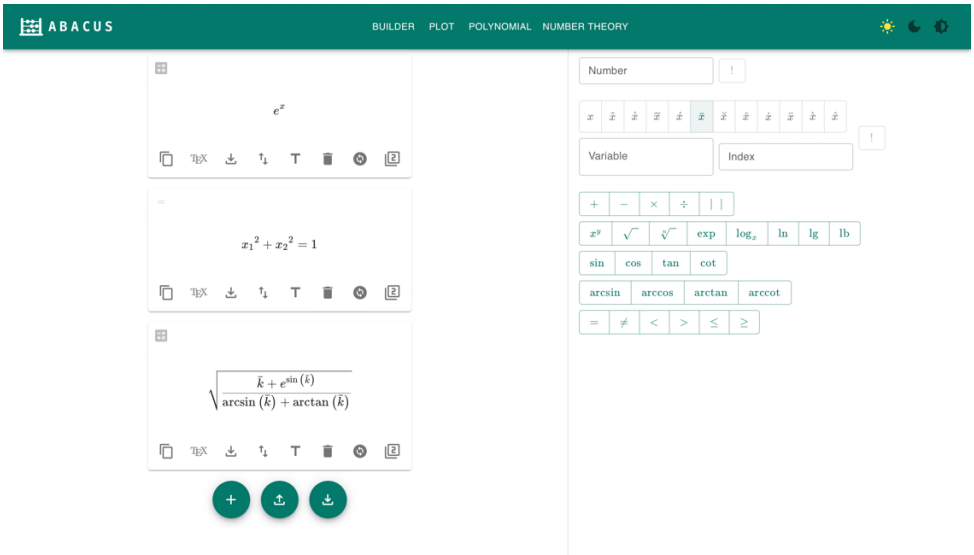

Рисунок 3.3 – Сторінка *builder* (світла тема)

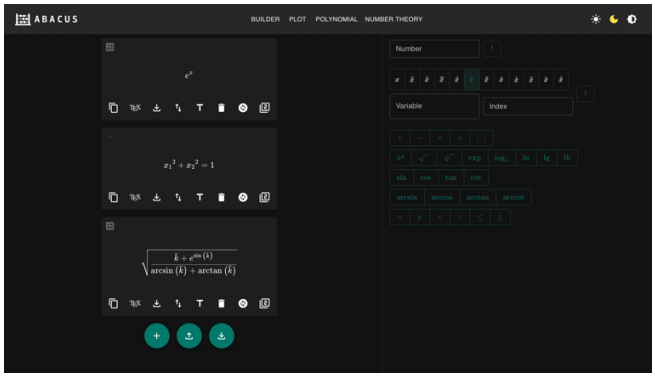

Рисунок 3.4 – Сторінка *builder* (темна тема)

Математичні вирази відображаються на спеціальних карточках (рис. 3.5), кожна з яких містить набор функціональних кнопок для зручної взаємодії.

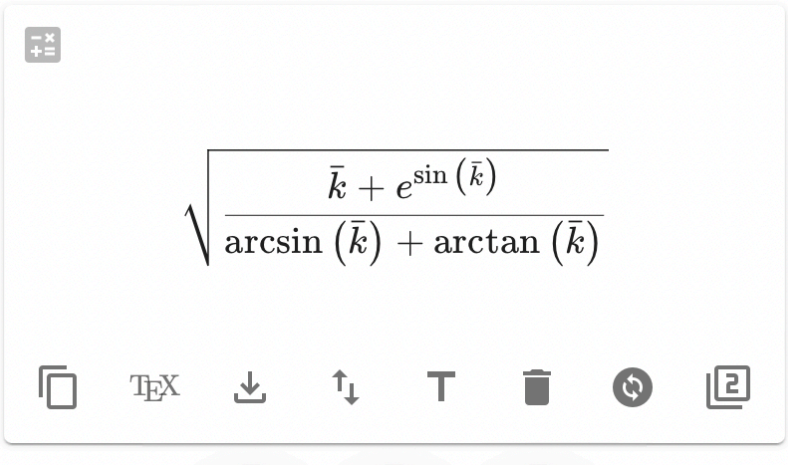

Рисунок 3.5 – Карточка виразу

Загалом наявні такі кнопки, які використовуються для:

– копіювання виразу в буфер обміну;
– копіювання tex-нотації в буфер обміну;
– завантажування файлу з виразом;
– експорту в зображення;
– додавання заголовку для виразу;
– вилучення карточки із списку;
– обміну лівого та правого операнду (для бінарних операторів);
– дублювання карточки.

У верхньому лівому куті представлено значок, який символізує тип математичного виразу. При увімкненні спеціальних режимів (вибору виразу, побудови графіків) у правому верхньому куті залежно від режиму може з’являтися та чи інша кнопка. Для режиму вибору це кнопка для застосування оператору саме до поточного виразу, а для режиму побудови графіків – це кнопка, яка додає відповідну функцію до графіку.

Слід зазначити, що в додатку підтримується спосіб оперування елементами drag-and-drop для виразів, тобто можна потягнути за вираз та відпустити його над іншим, і за можливості перший вираз буде підставлено в другий (рис. 3.6–3.7).

Під час перетягування підсвічується частина виразу, в яку буде підставлено нове значення (рис. 3.6).

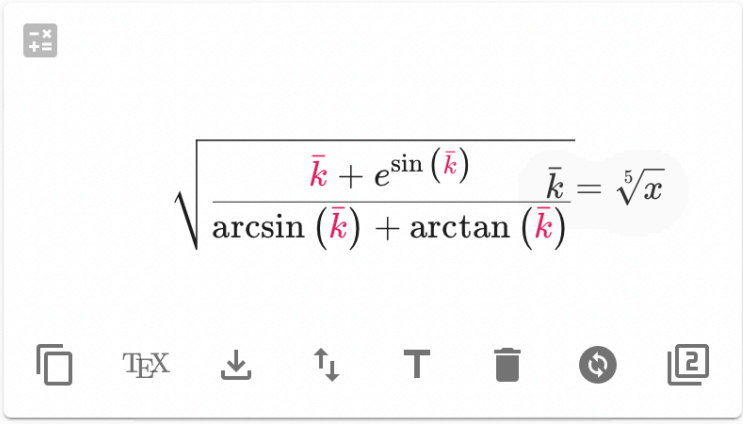

Рисунок 3.6 – Робота функції drag-and-drop

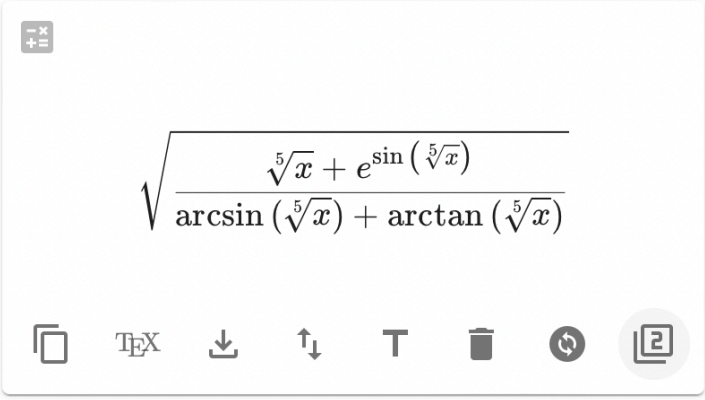

Рисунок 3.7 – Результат роботи функції drag-and-drop

Також у списку знаходяться кнопки для управління ним, за допомогою яких можна додати новий пустий вираз, завантажити файл з одним або декількома виразами з комп’ютера, завантажити усі вирази на комп’ютер (рис. 3.8).

Рисунок 3.8 – Кнопки управління списком виразів

Панель інструментів створення виразів включає три складові. Перша представлена полем для вводу чисел та кнопкою додавання числа до виразу

(рис. 3.9). Кнопка є неактивною, доки користувач не введе валідне число, якщо воно невалідне, то кнопка підсвічується червоним (рис. 3.10).

Рисунок 3.9 – Валідний ввід числа

Рисунок 3.10 – Невалідний ввід числа

Друга складова репрезентована полями для вводу математичної змінної та індексу змінної за необхідністю, полем вибору з доступних знаків пунктуації в ролі індексу (теж за необхідністю) та кнопкою додавання змінної до виразу, яка так само, як і кнопка додавання числа, є неактивною при невалідному вводі й підсвічується червоним (рис. 3.11–3.12).

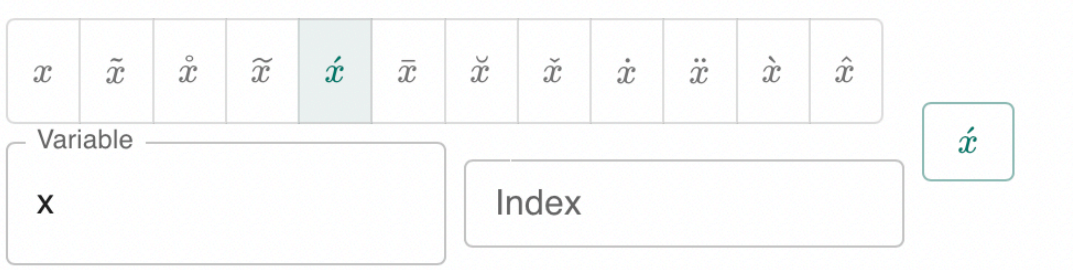

Рисунок 3.11 – Ввід змінної

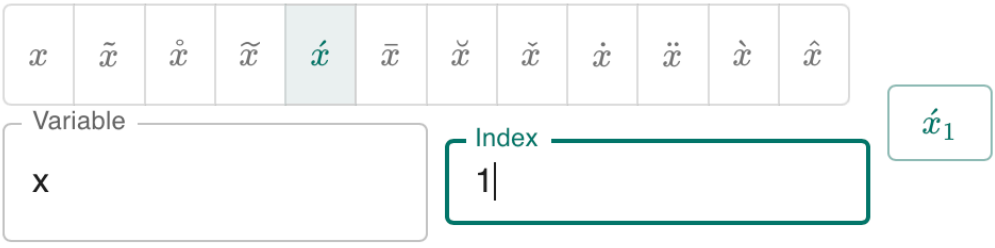

Рисунок 3.12 – Ввід змінної з індексом

Третя складова представлена сукупністю наборів кнопок, кожна з яких відповідає окремому математичному оператору (рис. 3.13).

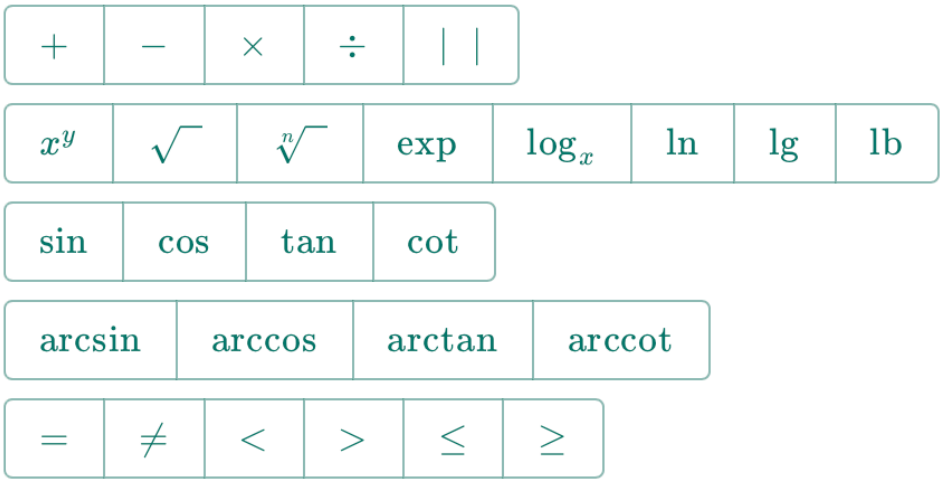

Рисунок 3.13 – Панель операторів

Усі кнопки при натисненні вмикають режим вибору (рис. 3.14), завдяки якому можна обрати конкретний вираз для застосування оператора.

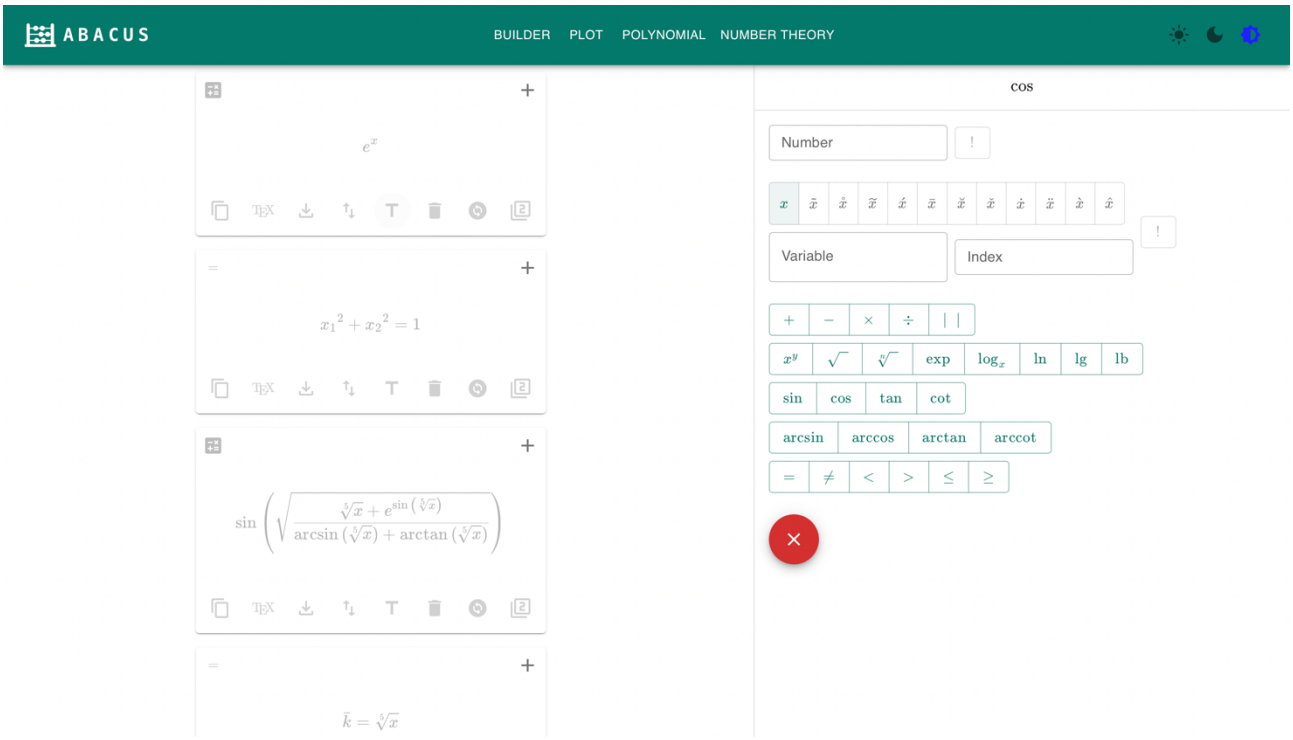

Рисунок 3.14 – Режим вибору

Усі кнопки панелі також підтримують drag-and-drop (рис. 3.15–3.16), що дає можливість застосовувати оператори перетягуванням і значно спрощує процес конструювання виразу.

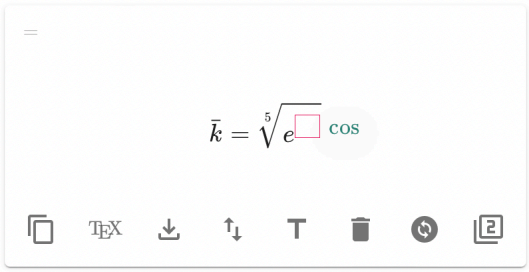

Рисунок 3.15 – Робота функції drag-and-drop для оператора

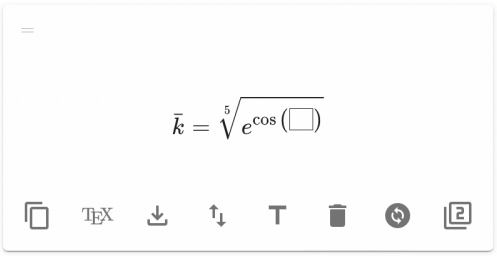

Рисунок 3.16 – Результат роботи функції drag-and-drop для оператора

Усі вирази зберігаються в глобальному сховищі, тому не зникають при переході з однієї сторінки на іншу.

На сторінці *plot* надається можливість побудови графіків функцій (рис. 3.17).

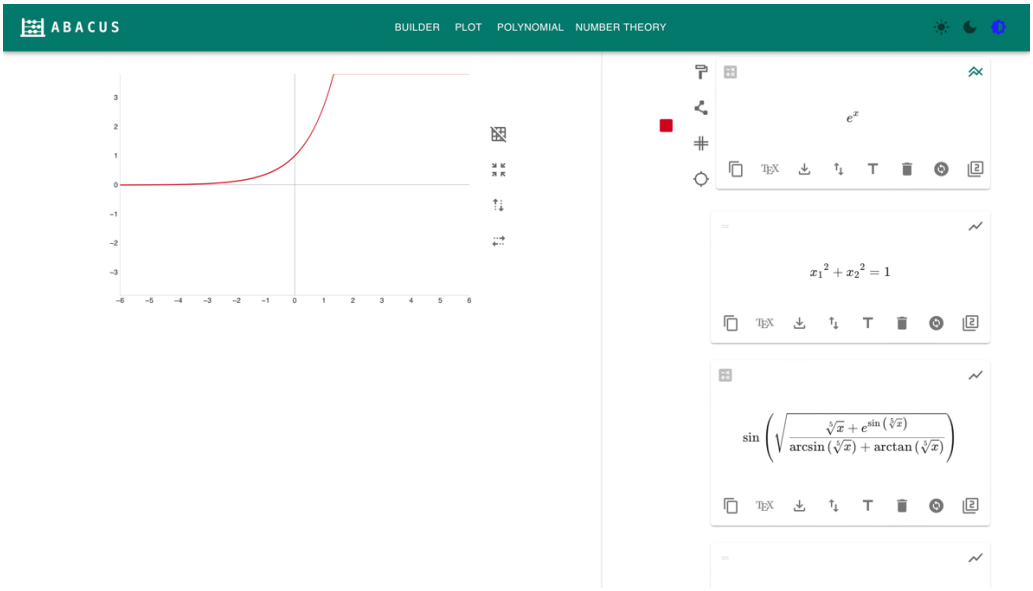

Рисунок 3.17 – Сторінка *plot*

На сторінці присутній той же список виразів, але в режимі побудови графіків. Відмінність полягає в тому, що в правому верхньому куті наявна кнопка для додавання виразу до графіку функцій (рис. 3.17–3.18), при натисненні на яку зліва від карточки з’являється набір кнопок для налаштування графіка (рис. 3.18).

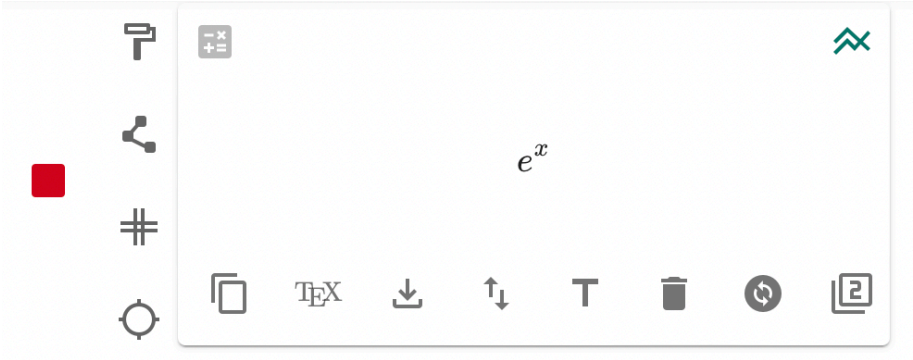

Рисунок 3.18 – Карточка виразу з кнопками графіку

За допомогою цих кнопок можна обрати:

– колір;
– систему координат (декартову чи полярну);
– чи замальовувати площу під графіком;
– буде графік неперервним чи складатиметься з набору точок;
– чи показувати підказку з координатами для цього графіку.

Справа від графіку функції розташовується набір кнопок, відповідальних за налаштування всього графіку (рис. 3.19).

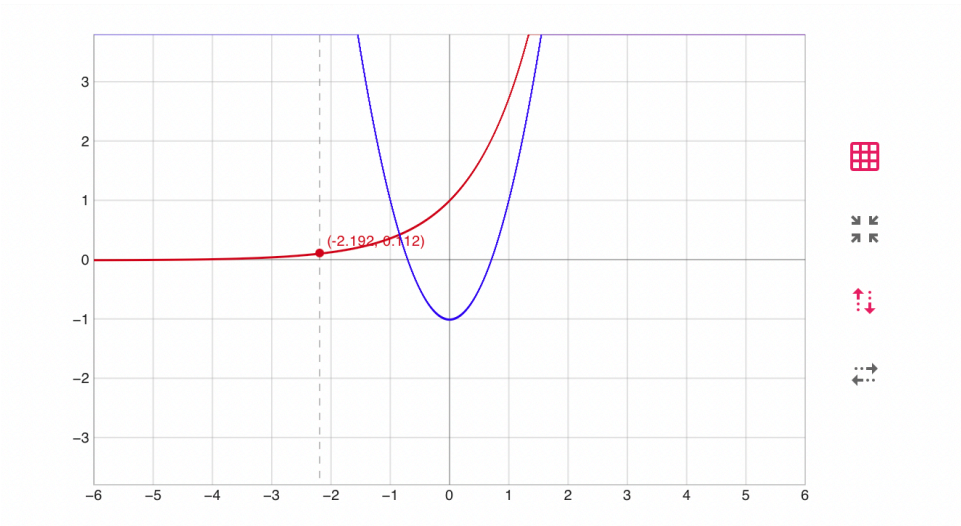

Рисунок 3.19 – Графік з кнопками для налаштування

Користувачеві надається можливість вибору:

– чи показувати сітку;

– чи надавати можливість для зміни масштабу;

– чи показувати штрихову лінію-підказку для *x* та *y* координат.

На сторінці *polynomial* подаються інструменти для обчислення операцій над поліномами (рис. 3.20).

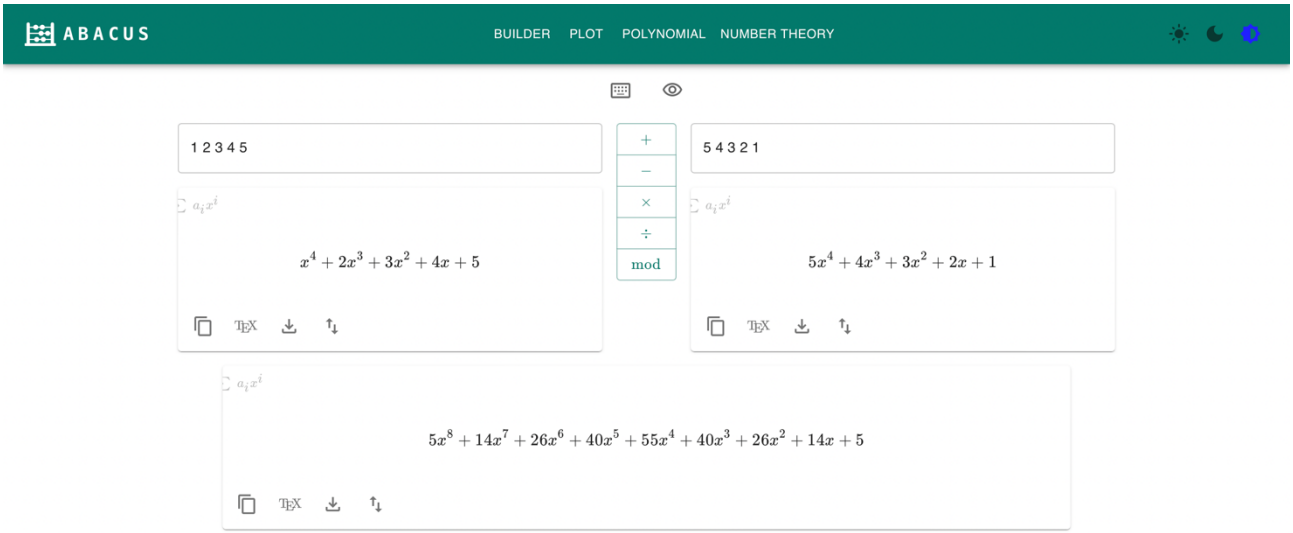

Рисунок 3.20 – Сторінка *polynomial*

Сторінка складається з:

– двох полів, результат вводу до яких одразу відображається у відповідних карточках під ними;

– групи кнопок операцій (сума, різниця, множення, ділення, остача від ділення), при натиску на які обчислюється результат;

– карточки для результату обчислення;

– кнопок вибору режиму вводу та відображення.

Користувач має можливість обрати режим вводу поліномів текстом (вводяться відповідні коефіцієнти через пробіли) та з файлу (коефіцієнти можуть бути представлені у квадратних дужках через кому або у вищезгаданому форматі

файлу) (рис. 3.21–3.22), а також режим, при якому поліноміальні вирази не відображаються, а тільки повідомляється про їх статус (цей режим знаходить своє застосування при обчисленнях з досить великими поліномами, що дозволяє не витрачати ресурси на відображення многочленів, які навіть не можуть поміститися на екран) (рис. 3.23).

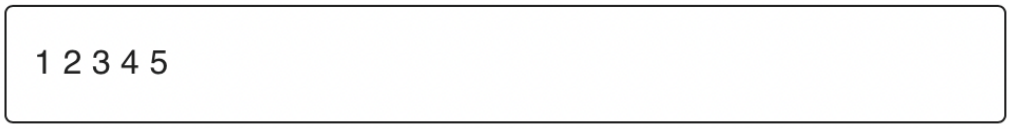

Рисунок 3.21 – Поле текстового вводу

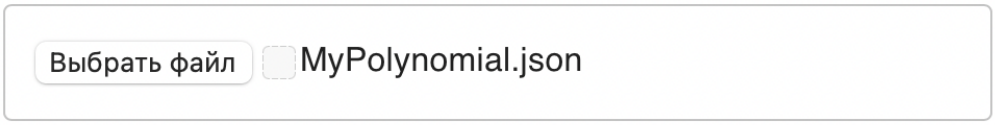

Рисунок 3.22 – Поле файлового вводу

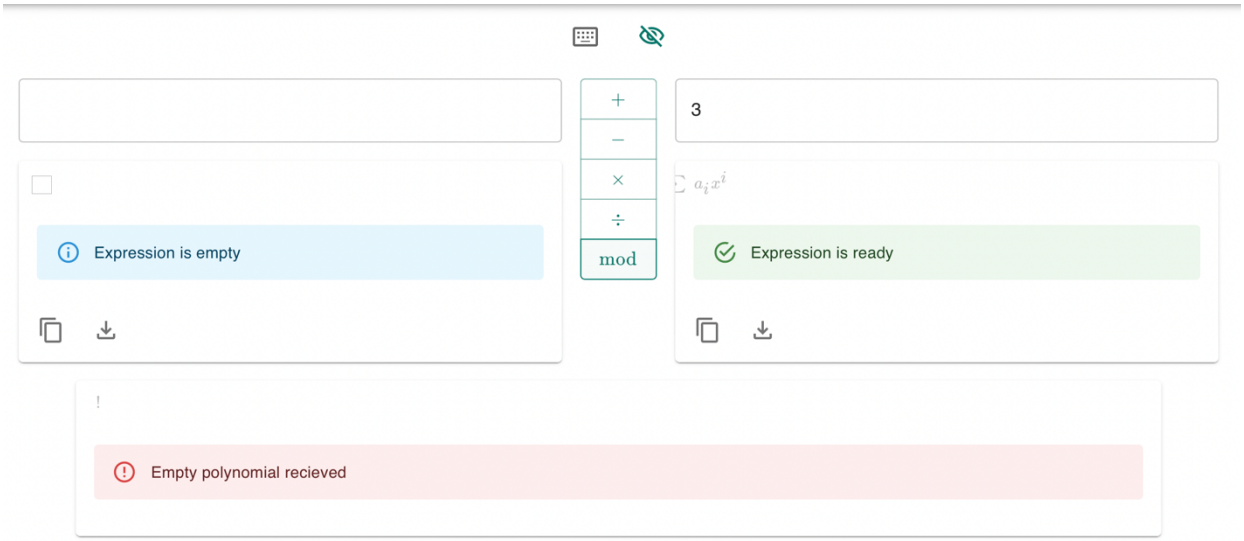

Рисунок 3.23 – Режим відображення статусів поліному

Карточки поліномів мають лише кнопки для копіювання виразу або його tex-нотації у буфер обміну, завантаження у файл, експорт у зображення.

Сторінка *number-theory* містить список операцій теорії чисел, які згруповані за напрямами, та набір карточок, що додаються натисканням на операцію (рис. 3.24).

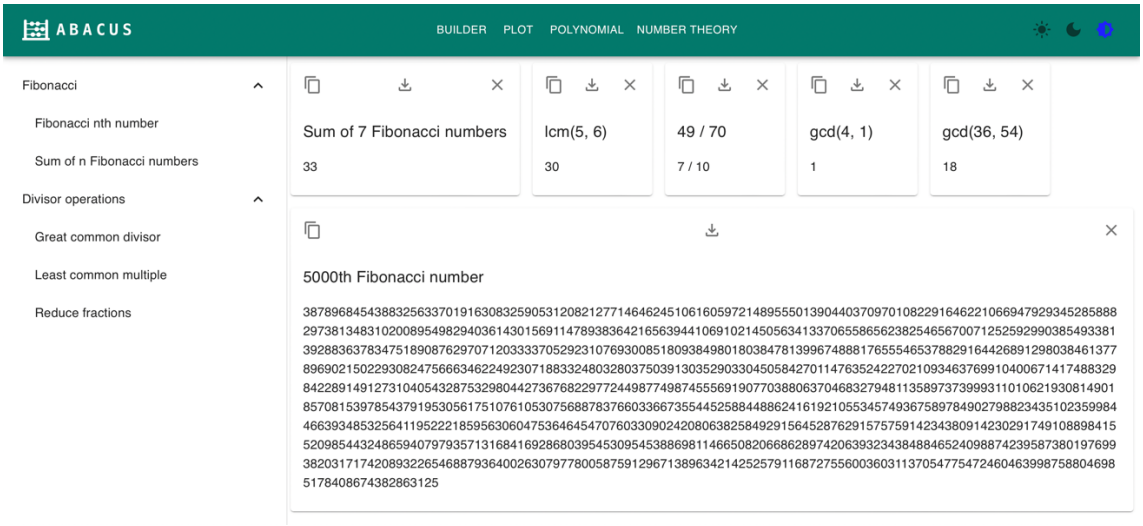

Рисунок 3.24 – Сторінка *number-theory*

Спершу в карточці з’являється форма для вводу чисел, кожне з полів валідується (рис. 3.25).

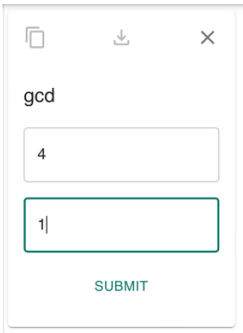

Рисунок 3.25 – Карточка з валідною формою

При невалідному вводі кнопка підтвердження є неактивною (рис. 3.26).

Рисунок 3.25 – Карточка з невалідною формою

При натисканні на кнопку форма замінюється на результат виконання відповідної операції (рис. 3.27).

Рисунок 3.27 – Карточка з результатом обчислення

Карточка з вже обчисленим результатом теж має кнопки копіювання виразу та його завантаження, а також кнопку, яка прибирає картку з набору.

Окремо слід зазначити, що більшість кнопок представлені значками, але при наведенні на них поруч з’являються текстові підказки (рис. 3.28).

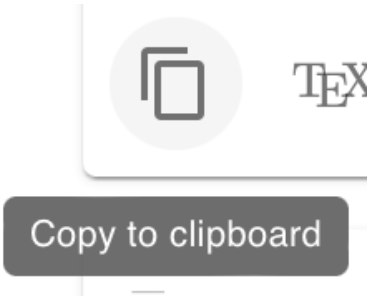

Рисунок 3.28 – Текстові підказки на кнопках

Як бачимо, усі функції веб-додатку, заявлені в постановці задачі, були реалізовані в повному обсязі.

ВИСНОВКИ

За результатами виконання кваліфікаційної роботи на основі описаних у ній математичних моделей створено програмне забезпечення, яке суттєво зменшує обчислювальну складність математичних операцій.

У ході підготовки кваліфікаційної роботи було виконано такі завдання:

1. Здійснено огляд науково-практичних досліджень з використання математичного апарату в різних сферах життєдіяльності людини.

2. Проаналізовано схожі застосунки, які розв’язують подібні математичні задачі, визначено переваги та недоліки цих програм. На основі отриманої інформації сформульовано вимоги до додатку, які необхідно було врахувати при його розробці.

3. Запропоновано методику вирішення поставлених задач, яка відповідає логіці проєкту. Серед існуючих підходів до створення веб-додатків було обрано SPA підхід, який дозволяє розробляти швидкі, динамічні веб-системи. З метою прискорення математичних обчислень було обрано алгоритми оптимізації, які необхідно застосувати при розробці відповідного програмного забезпечення. Зокрема, проаналізовано можливість використання алгоритму обчислення швидкого перетворення Фур’є для вирішення поставленої в роботі задачі.

4. Створено веб-додаток із використанням алгоритмів оптимізації, що дозволяє суттєво зменшити обчислювальну складність операцій. На основі JSON розроблено свій формат, який можна зручно використовувати в JavaScript для опису математичних виразів, їх зберігання, маніпулювання, обміну і навіть створення файлів вручну. Показано, що розроблений формат файлу має містити таку інформацію про математичний об’єкт: тип математичного виразу; операнди, на які поширюється дія оператора, або інша інформація, яка відповідає типу математичного виразу.

Результати проведеного дослідження підтверджують ефективність використання алгоритмів оптимізації у веб-додатку для досягнення швидкості розв’язання нетривіальних математичних задач. Основними типами задач, що

підтримуються програмою, є операції над поліномами (сума, різниця, множення, ділення, ділення з остачею), операції теорії чисел (пошук n-го числа Фібоначчі, пошук найбільшого спільного дільника, найменшого спільного кратного, скорочення дробів), побудова графіку функцій (декартових або полярних координат, функцій однієї змінної або рівняння двох змінних) тощо.

Розроблене програмне забезпечення є надійним, безпечним та простим у використанні, має широкий функціонал та зручний користувацький інтерфейс. Створений веб-додаток відповідає всім вимогам, сформульованим на етапі постановки задачі.

Подальше вдосконалення програмного продукту пов’язане з розширенням його функціональності за рахунок використання інших алгоритмічних рішень та підтримки нових типів математичних задач.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бобик О. І., Берегова Г. І., Копитко Б. І. Теорія ймовірності та математична статистика : підручник. Київ : Професіонал, 2007. 560 с.
2. Богданов О. М., Зінченко Я. В. Методи зменшення часу реалізації операції множення надвеликих чисел для системи захисту інформації. *Реєстрація, зберігання і обробка даних*. 2004. Т. 6. № 4. С. 94–106.
3. Василенко Н. М. Обернені числа Фібоначчі: властивості та застосування. *Науковий часопис НПУ імені М. П. Драгоманова. Серія 1. Фізико-математичні науки.* Київ : НПУ імені М. П. Драгоманова. 2012. № 13(2). С. 9–22.
4. Гаврилюк О. Д. Використання хмаро орієнтованих технологій навчання для формування професійних компетентностей майбутніх бакалаврів статистики: понятійно-термінологічний апарат дослідження. *Науковий часопис НПУ імені М. П. Драгоманова. Серія 5. Педагогічні науки: реалії та перспективи*. 2019. Вип. 66. С. 30–35.
5. Герасименко В. І. Математична фізика. *Енциклопедія сучасної України*: онлайн-версія / редкол.: І. М. Дзюба та ін.; НАН України, НТШ. Київ : Інститут енциклопедичних досліджень НАН України, 2018. URL: https://esu.com.ua/article-66934
6. Григорків В. С., Григорків М. В. Оптимізаційні методи та моделі : підручник. Чернівці : Чернівецький нац. ун-т, 2016. 400 с.
7. Гуляницький Л. Ф. Мулеса О. Ю. Прикладні методи комбінаторної оптимізації : навч. посіб. Київ : ВПЦ «Київський університет». 2016. 133 с.
8. Дослідження операцій та методи оптимізації: методичні рекомендації до практичних завдань для студентів усіх спеціальностей першого (бакалаврського) рівня / уклад. С. В. Прокопович, О. В. Панасенко, Л. О. Чаговець. Харків: ХНЕУ ім. С. Кузнеця, 2019. 64 с.
9. Квєтний Р. Н., Богач І. В., Софина О. Ю., Шушура О. М. Комп’ютерне моделювання систем та процесів. Методи обчислень : навч. посібник / за заг.

ред. Р. Н. Квєтного. Ч. 1. Вінниця : ВНТУ, 2012. 193 с. URL: http://kist.ntu.edu.ua/textPhD/kmsp.pdf

10. Креневич А. П., Бородін В. А. Видавнича система LaTeX : методичні вказівки до лабораторних занять з дисципліни «Практикум на ЕОМ». Київ : Київський університет. 2007. 49 с.
11. Махней О. В. Практикум з LaTeX : методичні рекомендації. Івано-Франківськ : Голіней, 2018. 36 с.
12. Мудрик І. Я. Автоматизовані системи діагностування стану пацієнтів, хворих на есенціальний тремор : дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 121. Тернопіль, 2021. 198 с.
13. Осадча К. П., Хромишев О. В. Розв'язання математичних задач засобами мови програмування Python. *Інформаційні технології в освіті та науці*. 2016. № 5. С. 216–220.
14. Рум'янцева К. Є., Вільчинська О. М. Використання економіко-математичних моделей під час вивчення дисциплін циклу «Математика для економістів». *Наукові записки. Серія: Проблеми методики фізико-математичної і технологічної освіти.* 2014. Вип. 5. Ч. 2. С. 49–53.
15. Рябуха Т. В. Математична біологія. *Енциклопедія сучасної України*: онлайн-версія / редкол.: І. М. Дзюба та ін.; НАН України, НТШ. Київ: Інститут енциклопедичних досліджень НАН України, 2018. URL: https://esu.com.ua/article-66927
16. Семеріков С. О. Теплицький І. О. Застосування системи комп'ютерної алгебри Maxima для генерування математичних текстів в системі дистанційного навчання. *Тези доповідей науково-практичної конференції «Нові технології навчання: психологічні аспекти»* / за ред. С. Д. Максименка, М. Л. Смульсон. Житомир, 2007. С. 39–40.
17. SPA в програмуванні: розбираємося з одностранічніками. *FoxmindEd* : веб-сайт. URL: https://foxminded.ua/spa-u-prohramuvanni/ (дата звернення: 24.05.2023).

18. Тєлєтов Д. О. Можливості застосування швидкого перетворення Фур'є для розв'язання практичних задач у computer science. *Інформатика. Математика. Автоматика*. Суми, 2023. С. 83–84.
19. Чим веб-додаток відрізняється від сайту? *Sunsoft* : веб-сайт. URL: http://sunsoft.com.ua/uk/Article/38 (дата звернення: 18.05.2023).
20. Шаповалов С. П., Шаповалов І. М. Числа Фібоначчі в сучасній картині світу. *Сучасна картина світу: інтеграція наукового та позанаукового знання* : збірник наукових праць. Суми, 2004. Вип. 3. С. 60–63.
21. Шкарупа А. О. Стрельцов О. А. Аналіз особливостей сучасних веб-додатків. *Автоматизація та приладобудування («Automation and Development of Electronic Devices» ADED-2022)* : збірник студентських наукових статей / редкол.: І.Ш. Невлюдов та ін., Харків : ХНУРЕ, 2022. Вип. 2. С. 102–108.
22. Ashraf S. Avoiding Vulnerabilities and Attacks with a Proactive Strategy for Web Applications. *Advances in Robotics and Mechanical Engineering*. Vol. 3. Issue 2. 2021. P. 263–271. URL: https://www.researchgate.net/publication/354904796_Avoiding_Vulnerabilities_and_Attacks_with_a_Proactive_Strategy_for_Web_Applications
23. Cooley J. W., Tukey J. W. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation.* 1965. Vol. 19. P. 297–301.
24. Desmos : веб-сайт. URL: https://www.desmos.com/ (дата звернення: 08.05.2023).
25. Difference between Website and Web Application (Web App). *Guru99* : веб-сайт. URL: https://www.guru99.com/difference-web-application-website.html (дата звернення: 18.05.2023).
26. GeoGebra : веб-сайт. URL: https://www.geogebra.org (дата звернення: 08.05.2023).
27. Herron D. Node.js Web Development: Create real-time server-side applications with this practical, step-by-step guide, 3rd Edition. Birmingham: Packt, 2016. 376 p.

28. Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. 3rd ed. 2009. London: The MIT Press. 1292 p.
29. KaTeX : веб-сайт. URL: https://katex.org (дата звернення: 03.04.2023).
30. Mathcad : веб-сайт. URL: https://www.mathcad.com/ (дата звернення: 10.05.2023).
31. MathML. *MDN Web Docs* : веб-сайт. URL: https://developer.mozilla.org/en-US/docs/Web/MathML (дата звернення: 19.05.2023).
32. MUI : веб-сайт. URL: https://mui.com (дата звернення: 10.05.2023).
33. Nearly Optimal Sparse Fourier Transform / H. Hassanieh, P. Indyk, D. Katabi, E. Price. *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*. 2012. P. 563–578. URL : https://arxiv.org/pdf/1201.2501.pdf
34. On the numerical performance of finite-difference-based methods for derivative-free optimization / H.-J. M. Shi, M. Q. Xuan, F. Oztoprak, J. Nocedal. *Optimization Methods and Software*. 2023. Vol. 38, no. 2. P. 289–311. URL: https://doi.org/10.1080/10556788.2022.2121832.
35. Pascal G, Bruno G., Daniel. R. Fast in-place algorithms for polynomial operations: division, evaluation, interpolation. *ISSAC '20*: Proceedings of the 45th International Symposium on Symbolic and Algebraic Computation 2020. P. 210–217. URL: https://doi.org/10.1145/3373207.3404061.
36. React : веб-сайт. URL: https://react.dev (дата звернення: 05.04.2023).
37. Shallit J. Origins of the Analysis of the Euclidean Algorithm. *Historia Mathematica*. 1994. Vol. 21. P. 401–419.
38. Symbolab : веб-сайт. URL: https://www.symbolab.com/ (дата звернення: 08.05.2023).
39. Tamam B., Dasari D. The use of Geogebra software in teaching mathematics. *Journal of Physics: Conference Series*. 2021. Vol. 1882. URL: https://doi.org/10.1088/1742-6596/1882/1/012042 (дата звернення: 07.05.2023).

40. Wolfram|Alpha : веб-сайт. URL: https://www.wolframalpha.com/ (дата звернення: 08.05.2023).
41. Wolfram S. A New Kind of Science. Champaign : Wolfram Media, 2002. 1280 p.

ДОДАТОК А

Використані алгоритми:

Швидке перетворення Фур'є – *FFT.js*

```
import Complex from 'complex.js';

const w = (N, positive) => {
  const phi = (2 * Math.PI) / N;
  return new Complex({ arg: positive ? phi : -phi, abs: 1 });
};

const getBitReverse = (bits, numBits) => {
  let rev = 0;
  for (let j = 0; j < numBits; ++j) {
    if (bits & (1 << j)) {
      rev |= 1 << (numBits - 1 - j);
    }
  }
  return rev;
};

const FFT = (buffer_, isForwardDirection = true) => {
  if (buffer_.length < 2) return buffer_;

  let numBits = 0;
  while (1 << numBits < buffer_.length) ++numBits;

  for (let i = 0; i < buffer_.length; ++i) {
    const reversed = getBitReverse(i, numBits);
    if (i < reversed) {
      [buffer_[i], buffer_[reversed]] = [buffer_[reversed], buffer_[i]];
    }
  }

  for (let step = 1; step < buffer_.length; step <<= 1) {
    const w1 = w(step << 1, isForwardDirection);
    let wk = new Complex(1);
    for (let j = 0; j < step; ++j) {
      for (let i = j; i < buffer_.length; i += step << 1) {
        const a1 = buffer_[i].clone();
        const a2 = buffer_[i + step].mul(wk);
        buffer_[i] = a1.add(a2);
        buffer_[i + step] = a1.sub(a2);
        if (!isForwardDirection) {
          buffer_[i] = buffer_[i].div(2);
          buffer_[i + step] = buffer_[i + step].div(2);
        }
      }
      wk = wk.mul(w1);
    }
  }

  return buffer_;
};

export default FFT;
```

Операції над поліномами (сума, різниця, множення, ділення, ділення з остачею) – *polynomialOperations.js*

```
import FFT from './FFT';
import Complex from 'complex.js';

const fill = (array, size) => {
  const newArray = array.map((item) => new Complex(item));
  newArray.push(...Array(size - newArray.length).fill(new Complex(0)));
  return newArray;
};

const removeLeadingZeros = (polynom) => {
  while (polynom.at(-1) === 0) {
    polynom.pop();
  }
  return polynom;
};

const elementWise = (left, right, action) => {
  const size = Math.max(left.length, right.length);
  const result = [];
  for (let i = 0; i < size; i++) {
    result.push(action(left[i] ?? 0, right[i] ?? 0));
  }
  return removeLeadingZeros(result);
};

const fullDivide = (left, right) => {
  const divisor =
    right.at(-1) !== 0 ? right : removeLeadingZeros(right.slice());

  if (!divisor.length) {
    throw new Error('Division by zero');
  }

  const remainder = left.slice();
  const quotient = [];
  while (remainder.length >= divisor.length) {
    const shift = remainder.length - divisor.length;
    const factor = remainder.pop() / divisor.at(-1);
    quotient.unshift(factor);
    for (let i = 0; i < divisor.length - 1; i++) {
      remainder[shift + i] -= divisor[i] * factor;
    }
  }
  return { quotient, remainder };
};

export const add = (left, right) =>
  elementWise(left, right, (lhs, rhs) => lhs + rhs);

export const substract = (left, right) =>
  elementWise(left, right, (lhs, rhs) => lhs - rhs);

export const multiply = (left, right) => {
  let size = 1;
  const multipliedSize = left.length + right.length - 1;
  while (size < multipliedSize) size <<= 1;

  const leftFFT = FFT(fill(left, size));
```

```
  const rightFFT = FFT(fill(right, size));

  for (let index = 0; index < leftFFT.length; index++) {
    leftFFT[index] = leftFFT[index].mul(rightFFT[index]);
  }

  return FFT(leftFFT, false)
    .slice(0, multipliedSize)
    .map((item) => +item.re.toFixed(6));
};

export const divide = (left, right) => fullDivide(left, right).quotient;

export const modulo = (left, right) => fullDivide(left, right).remainder;
```

Операції над матрицями (множення) – *matrixOperations.js*

```
export const multiply = (
  left,
  right,
  scalarMultiply = (lhs, rhs) => lhs * rhs,
  scalarAdd = (lhs, rhs) => lhs + rhs,
  zero = 0
) => {
  if (left[0].length !== right.length) {
    throw new Error('Invalid matrix dimensions for multiplication');
  }

  const result = new Array(left.length);
  for (let i = 0; i < left.length; i++) {
    result[i] = new Array(right[0].length).fill(zero);
    for (let j = 0; j < right[0].length; j++) {
      for (let k = 0; k < left[0].length; k++) {
        result[i][j] = scalarAdd(
          result[i][j],
          scalarMultiply(left[i][k], right[k][j])
        );
      }
    }
  }
  return result;
};
```

Алгоритм швидкого піднесення до степеню – *fastPower.js*

```
const fastPower = (
  base,
  power,
  multiply = (lhs, rhs) => lhs * rhs,
  one = 1
) => {
  let result = one;
  while (power > 0) {
    if (power % 2 === 1) {
      result = multiply(result, base);
    }
    base = multiply(base, base);
    power = Math.floor(power / 2);
  }
  return result;
};
export default fastPower;
```

Алгоритм швидкого обчислення *n*-го числа Фібоначчі – *fibonacci.js*

```
import fastPower from './fastPower';
import { multiply } from './matrixOperations';

const fibonacci = (n) => {
  if (n <= 0) return 0n;

  const baseMatrix = [
    [1n, 1n],
    [1n, 0n],
  ];

  const resultMatrix = fastPower(
    baseMatrix,
    n - 1,
    (lhs, rhs) => multiply(lhs, rhs, undefined, undefined, 0n),
    [
      [1n, 0n],
      [0n, 1n],
    ]
  );

  return resultMatrix[0][0];
};

const sqrt5 = Math.sqrt(5);
const goldenRation = (1 + sqrt5) / 2;

export const fibonacciApproximate = (n) =>
  (goldenRation ** n - (1 - goldenRation) ** n) / sqrt5;

export const fibonacciSum = (n) => fibonacci(n + 2) - 1n;

export const fibonacciSumApproximate = (n) => fibonacciApproximate(n + 2) - 1;

export default fibonacci;
```

Алгоритми пошуку найбільшого спільного дільника, найменшого спільного кратного, скорочення дробів – *divisor.js*

```
export const gcd = (a, b) => {
  while (b) {
    [a, b] = [b, a % b];
  }
  return a;
};

export const lcm = (a, b) => (a / gcd(a, b)) * b;

export const reduceFractions = (numerator, denominator) => {
  const factor = gcd(numerator, denominator);
  return { numerator: numerator / factor, denominator: denominator / factor };
};
```

ДОДАТОК Б

Класи математичних виразів:

Клас Operator – *Operator.js*

```
import operators from '../operators';

export default class Operator {
  static getOperator(expression) {
    return operators[expression.name];
  }

  static zero = () => ({ name: 'number', number: 0 });

  static one = () => ({ name: 'number', number: 1 });

  static isNumber(expression) {
    return expression.name === 'number';
  }

  static _emptiness() {
    return '\\textcolor{transparent}{x}';
  }

  static _wrapTexBrackets(texText, condition = true) {
    if (!condition) {
      return texText;
    }
    return `\\left(${texText}\\right)`;
  }

  static _highlight(tex, operands, options) {
    const color = options.highlight?.(operands) && options.highlightColor;
    return color ? `\\textcolor{${color}}{${tex}}` : tex;
  }
}
```

Клас BinaryOperator – *BinaryOperator.js*

```
import Operator from './Operator';

export default class BinaryOperator extends Operator {
  constructor(left, right) {
    super();
    this.left = left;
    this.right = right;
  }

  static areEqual(lhs, rhs) {
    if (lhs.left.name !== rhs.left.name) return false;
    if (lhs.right.name !== rhs.right.name) return false;

    return (
      this.getOperator(lhs.left).areEqual(lhs.left, rhs.left) &&
      this.getOperator(lhs.right).areEqual(lhs.right, rhs.right)
    );
  }

  static isBinary() {
    return true;
```

```
  }

  static preOrder(operands, action) {
    if (action(operands)) return false;
    this.getOperator(operands.left).preOrder(operands.left, action);
    this.getOperator(operands.right).preOrder(operands.right, action);
  }

  static inOrder(operands, action) {
    this.getOperator(operands.left).inOrder(operands.left, action);
    if (action(operands)) return false;
    this.getOperator(operands.right).inOrder(operands.right, action);
  }

  static postOrder(operands, action) {
    this.getOperator(operands.left).postOrder(operands.left, action);
    this.getOperator(operands.right).postOrder(operands.right, action);
    if (action(operands)) return false;
  }

  static plot(operands, options) {
    const left = this.getOperator(operands.left).plot(operands.left, options);
    const right = this.getOperator(operands.right).plot(
      operands.right,
      options
    );
    return `${operands.name}(${left}, ${right})`;
  }

  static swap(operands) {
    [operands.left, operands.right] = [operands.right, operands.left];
  }
}
```

Клас UnaryOperator – *UnaryOperator.js*

```
import Operator from './Operator';

export default class UnaryOperator extends Operator {
  constructor(value) {
    super();
    this.value = value;
  }

  static isUnary() {
    return true;
  }

  static areEqual(lhs, rhs) {
    if (lhs.value.name !== rhs.value.name) return false;
    return this.getOperator(lhs).areEqual(lhs.value, rhs.value);
  }

  static preOrder(operands, action) {
    if (action(operands)) return false;
    this.getOperator(operands.value).preOrder(operands.value, action);
  }

  static inOrder(operands, action) {
    if (action(operands)) return false;
    this.getOperator(operands.value).inOrder(operands.value, action);
  }
```

```
  static postOrder(operands, action) {
    this.getOperator(operands.value).postOrder(operands.value, action);
    if (action(operands)) return false;
  }

  static plot(operands, options) {
    const value = this.getOperator(operands.value).plot(
      operands.value,
      options
    );
    return `${operands.name}(${value})`;
  }
}
```

Клас NullaryOperator – *NullaryOperator.js*

```
import Operator from './Operator';

export default class NullaryOperator extends Operator {
  constructor(left, right) {
    super();
    this.left = left;
    this.right = right;
  }

  static isNullary() {
    return true;
  }

  static preOrder(operands, action) {
    return action(operands);
  }

  static inOrder(operands, action) {
    return action(operands);
  }

  static postOrder(operands, action) {
    return action(operands);
  }
}
```

Клас ComparisonOperator – *ComparisonOperator.js*

```
import BinaryOperator from './BinaryOperator';

export default class ComparisonOperator extends BinaryOperator {
  static symbol = () => '\\not =';

  static isComparison() {
    return true;
  }

  static plot(operands, options) {
    return super.plot({ ...operands, name: 'sub' }, options);
  }

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left} = ${right}`, operands, options);
```

```
  }
}
```

Клас NumberOperator – *NumberOperator.js*

```
import operators from '../operators';
import NullaryOpertor from './NullaryOperator';

export default class NumberOperator extends NullaryOpertor {
  static tex(operands, options) {
    return this._highlight(String(operands.number), operands, options);
  }

  static plot(operands, options) {
    return `${operands.number}`;
  }

  static areEqual(lhs, rhs) {
    return lhs.number === rhs.number;
  }
}

operators.number = NumberOperator;
```

Клас VariableOperator – *VariableOperator.js*

```
import operators from '../operators';
import NullaryOpertor from './NullaryOperator';

export default class VariableOperator extends NullaryOpertor {
  static tex(operands, options) {
    let variable = operands.variable;
    if (operands.accent && this.getAccents().includes(operands.accent)) {
      variable = `\\${operands.accent}{${variable}}`;
    }
    if (operands.index || operands.index === 0) {
      variable = `${variable}_{${operands.index}}`;
    }
    return this._highlight(variable, operands, options);
  }

  static plot(operands, options) {
    return options.replace ? options.replace(operands) : 'x';
  }

  static areEqual(lhs, rhs) {
    return lhs.variable === rhs.variable && lhs.accent === rhs.accent;
  }

  static getAccents() {
    return [
      'tilde',
      'mathring',
      'widetilde',
      'acute',
      'bar',
      'breve',
      'check',
      'dot',
      'ddot',
      'grave',
      'hat',
```

```
    ];
  }

  static getPattern() {
    return /^[a-zA-Z]*$/;
  }
}

operators.variable = VariableOperator;
```

Клас AddOperator – *AddOperator.js*

```
import operators from '../operators';
import BinaryOperator from './BinaryOperator';

export default class AddOperator extends BinaryOperator {
  static symbol = () => '+';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left}+${right}`, operands, options);
  }
}

operators.add = AddOperator;
```

Клас SubstractOperator – *SubstractOperator.js*

```
import operators from '../operators';
import BinaryOperator from './BinaryOperator';

export default class SubstractOperator extends BinaryOperator {
  static symbol = () => '-';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left}-${right}`, operands, options);
  }
}

operators.sub = SubstractOperator;
```

Клас MultiplyOperator – *MultiplyOperator.js*

```
import operators from '../operators';
import BinaryOperator from './BinaryOperator';

export default class MultiplyOperator extends BinaryOperator {
  static symbol = () => '\\times';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    let result = '';
    const leftType = operands.left.name;
    const rightType = operands.right.name;
    if (
      (leftType === 'number' || leftType === 'variable') &&
      rightType === 'variable'
    ) {
```

```
      result = `${left}${right}`;
    } else {
      let lowerPrecednce = ['add', 'sub'];
      result = `${this._wrapTexBrackets(
        left,
        lowerPrecednce.includes(leftType)
      )}\\cdot${this._wrapTexBrackets(
        right,
        lowerPrecednce.includes(rightType)
      )}`;
    }
    return this._highlight(result, operands, options);
  }
}

operators.mul = MultiplyOperator;
```

Клас DivideOperator – *DivideOperator.js*

```
import operators from '../operators';
import BinaryOperator from './BinaryOperator';

export default class DivideOperator extends BinaryOperator {
  static symbol = () => '\\div';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`\\frac{${left}}{${right}}`, operands, options);
  }
}

operators.div = DivideOperator;
```

Клас PowOperator – *PowOperator.js*

```
import operators from '../operators';
import BinaryOperator from './BinaryOperator';
import { number, object } from 'yup';

export default class PowOperator extends BinaryOperator {
  static symbol = () => 'x^y';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);

    let lowerPrecednce = ['add', 'sub', 'mul', 'div', 'pow'];
    let result = `{${this._wrapTexBrackets(
      left,
      lowerPrecednce.includes(operands.left.name)
    )}}^{${right}}`;

    return this._highlight(result, operands, options);
  }

  static plot(operands, options) {
    if (operands.right.name === 'number') {
      if (object({ number: number().integer() }).validateSync(operands.right)) {
```

```
        return super.plot(operands, options);
      }
    }
    const baseChange = {
      name: 'exp',
      value: {
        name: 'mul',
        left: operands.right,
        right: { name: 'ln', value: operands.left },
      },
    };
    return this.getOperator(baseChange).plot(baseChange, options);
  }
}

operators.pow = PowOperator;
```

Клас AbsOperator – *AbsOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class AbsOperator extends UnaryOperator {
  static symbol = () => `|${this._emptiness()}|`;

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(`|{${value}}|`, operands, options);
  }
}

operators.abs = AbsOperator;
```

Клас ExpOperator – *ExpOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class ExpOperator extends UnaryOperator {
  static symbol = () => '\\exp';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(`e^{${value}}`, operands, options);
  }
}

operators.exp = ExpOperator;
```

Клас SqrtOperator – *SqrtOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class SqrtOperator extends UnaryOperator {
  static symbol = () => `\\sqrt {${this._emptiness()}}`;

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(`\\sqrt {${value}}`, operands, options);
  }
}
```

```
operators.sqrt = SqrtOperator;
```

Клас NrtOperator – *NrtOperator.js*

```
import operators from '../operators';
import BinaryOperator from './BinaryOperator';

export default class NrtOperator extends BinaryOperator {
  static symbol = () => `\\sqrt[n]{${this._emptiness()}}`;

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`\\sqrt[${left}]{${right}}`, operands, options);
  }

  static plot(operands, options) {
    return super.plot({ ...operands, name: 'nthRoot' }, options);
  }
}

operators.nrt = NrtOperator;
```

Клас LogOperator – *LogOperator.js*

```
import operators from '../operators';
import BinaryOperator from './BinaryOperator';

export default class LogOperator extends BinaryOperator {
  static symbol = () => '\\log_x';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(
      `\\log_${left} {${this._wrapTexBrackets(right)}}`,
      operands,
      options
    );
  }

  static plot(operands, options) {
    const baseChange = {
      name: 'div',
      left: { name: 'ln', value: operands.right },
      right: { name: 'ln', value: operands.left },
    };
    return this.getOperator(baseChange).plot(baseChange, options);
  }
}

operators.log = LogOperator;
```

Клас LnOperator – *LnOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class LnOperator extends UnaryOperator {
  static symbol = () => '\\ln';
```

```
  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\ln {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }

  static plot(operands, options) {
    const value = this.getOperator(operands.value).plot(
      operands.value,
      options
    );
    return `log(${value})`;
  }
}

operators.ln = LnOperator;
```

Клас LgOperator – *LgOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class LgOperator extends UnaryOperator {
  static symbol = () => '\\lg';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\lg {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }

  static plot(operands, options) {
    const generalized = {
      name: 'log',
      left: { name: 'number', number: 10 },
      right: operands.value,
    };
    return this.getOperator(generalized).plot(generalized, options);
  }
}

operators.lg = LgOperator;
```

Клас LbOperator – *LbOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class LbOperator extends UnaryOperator {
  static symbol = () => '\\operatorname{lb}';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\operatorname{lb} {${this._wrapTexBrackets(value)}}`,
      operands,
```

```
      options
    );
  }

  static plot(operands, options) {
    const generalized = {
      name: 'log',
      left: { name: 'number', number: 2 },
      right: operands.value,
    };
    return this.getOperator(generalized).plot(generalized, options);
  }
}

operators.lb = LbOperator;
```

Клас SinOperator – *SinOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class SinOperator extends UnaryOperator {
  static symbol = () => '\\sin';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\sin {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }
}

operators.sin = SinOperator;
```

Клас CosOperator – *CosOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class CosOperator extends UnaryOperator {
  static symbol = () => '\\cos';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\cos {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }
}

operators.cos = CosOperator;
```

Клас TanOperator – *TanOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class TanOperator extends UnaryOperator {
```

```
  static symbol = () => '\\tan';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\tan {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }
}

operators.tan = TanOperator;
```

Клас CotOperator – *CotOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class CotOperator extends UnaryOperator {
  static symbol = () => '\\cot';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\cot {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }

  static plot(operands, options) {
    const opposite = {
      name: 'div',
      left: { name: 'number', number: 1 },
      right: { name: 'tan', value: operands.value },
    };
    return this.getOperator(opposite).plot(opposite, options);
  }
}

operators.cot = CotOperator;
```

Клас ArcsinOperator – *ArcsinOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class ArcsinOperator extends UnaryOperator {
  static symbol = () => '\\arcsin';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\arcsin {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }
}
```

```
operators.asin = ArcsinOperator;
```

Клас ArccosOperator – *ArccosOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class ArccosOperator extends UnaryOperator {
  static symbol = () => '\\arccos';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\arccos {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }
}

operators.acos = ArccosOperator;
```

Клас ArctanOperator – *ArctanOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class ArctanOperator extends UnaryOperator {
  static symbol = () => '\\arctan';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\arctan {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }
}

operators.atan = ArctanOperator;
```

Клас ArccotOperator – *ArccotOperator.js*

```
import operators from '../operators';
import UnaryOperator from './UnaryOperator';

export default class ArccotOperator extends UnaryOperator {
  static symbol = () => '\\operatorname{arccot}';

  static tex(operands, options) {
    const value = this.getOperator(operands.value).tex(operands.value, options);
    return this._highlight(
      `\\operatorname{arccot} {${this._wrapTexBrackets(value)}}`,
      operands,
      options
    );
  }

  static plot(operands, options) {
```

```
    const opposite = {
      name: 'atan',
      value: {
        name: 'div',
        left: { name: 'number', number: 1 },
        right: operands.value,
      },
    };
    return this.getOperator(opposite).plot(opposite, options);
  }
}

operators.acot = ArccotOperator;
```

Клас ModuloOperator – *ModuloOperator.js*

```
import operators from '../operators';
import BinaryOperator from './BinaryOperator';

export default class ModuloOperator extends BinaryOperator {
  static symbol = () => '\\bmod';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(` ${left}\\pmod ${right} `, operands, options);
  }
}

operators.mod = ModuloOperator;
```

Клас PolynomialOperator – *PolynomialOperator.js*

```
import operators from '../operators';
import Operator from './Operator';

export default class PolynomialOperator extends Operator {
  static symbol = () => 'sum_{i=0}^n a_{i}x^{i}';

  static tex(operands, options) {
    const texValue = (
      operands.vector.length === 0 ? [0] : operands.vector
    ).reduceRight((accumulator, element, index) => {
      if (isNaN(element)) {
        accumulator += '\\space\\textcolor{red}{!}\\space';
      } else if (element !== 0) {
        let ai = ``;

        if (accumulator) {
          ai += element > 0 ? `+ ` : `- `;
        } else if (element < 0) {
          ai += `-`;
        }
        if (Math.abs(element) !== 1 || index === 0) {
          ai += Math.abs(element);
        }

        let xi = ``;
        if (index !== 0) {
          xi = index === 1 ? `x` : `x^{${index}}`;
        }
        accumulator += `${ai}${xi}`;
```

```
      } else if (!accumulator && index === 0) {
        accumulator += '0';
      }
      return accumulator;
    }, '');

    return this._highlight(texValue, operands, options);
  }
}

operators.polynomial = PolynomialOperator;
```

Клас EmptyOperator – *EmptyOperator.js*

```
import operators from '../operators';
import NullaryOpertor from './NullaryOperator';

export default class EmptyOperator extends NullaryOpertor {
  static symbol = () => `\\boxed{${this._emptiness()}}`;

  static tex(operands, options) {
    return this._highlight(this.symbol(), operands, options);
  }

  static plot(operands, options) {
    return '';
  }

  static areEqual(lhs, rhs) {
    return true;
  }
}

operators.empty = EmptyOperator;
```

Клас EqualityOperator – *EqualityOperator.js*

```
import operators from '../operators';
import ComparisonOperator from './ComparisonOperator';

export default class EqualityOperator extends ComparisonOperator {
  static symbol = () => '=';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left} = ${right}`, operands, options);
  }
}

operators.equality = EqualityOperator;
```

Клас InequalityOperator – *InequalityOperator.js*

```
import operators from '../operators';
import ComparisonOperator from './ComparisonOperator';

export default class InequalityOperator extends ComparisonOperator {
  static symbol = () => '\\not =';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
```

```
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left} \\not = ${right}`, operands, options);
  }
}

operators.inequality = InequalityOperator;
```

Клас LessOperator – *LessOperator.js*

```
import operators from '../operators';
import ComparisonOperator from './ComparisonOperator';

export default class LessOperator extends ComparisonOperator {
  static symbol = () => '<';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left} < ${right}`, operands, options);
  }
}

operators.less = LessOperator;
```

Клас GreaterOperator – *GreaterOperator.js*

```
import operators from '../operators';
import ComparisonOperator from './ComparisonOperator';

export default class GreaterOperator extends ComparisonOperator {
  static symbol = () => '>';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left} > ${right}`, operands, options);
  }
}

operators.greater = GreaterOperator;
```

Клас LeqOperator – *LeqOperator.js*

```
import operators from '../operators';
import ComparisonOperator from './ComparisonOperator';

export default class LeqOperator extends ComparisonOperator {
  static symbol = () => '\\leq';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left} \\leq ${right}`, operands, options);
  }
}

operators.leq = LeqOperator;
```

Клас GeqOperator – *GeqOperator.js*

```
import operators from '../operators';
```

```
import ComparisonOperator from './ComparisonOperator';

export default class GeqOperator extends ComparisonOperator {
  static symbol = () => '\\geq';

  static tex(operands, options) {
    const left = this.getOperator(operands.left).tex(operands.left, options);
    const right = this.getOperator(operands.right).tex(operands.right, options);
    return this._highlight(`${left} \\geq ${right}`, operands, options);
  }
}

operators.geq = GeqOperator;
```

Клас ExpressionOperator – *ExpressionOperator.js*

```
import operators from '../operators';
import Operator from './Operator';
import UnaryOperator from './UnaryOperator';

export default class ExpressionOperator extends UnaryOperator {
  static tex(operands, options) {
    return this.getOperator(operands.value).tex(operands.value, options);
  }

  static plot(operands, options) {
    return this.getOperator(operands.value).plot(operands.value, options);
  }

  static swap(operands) {
    return Operator.getOperator(operands.value).swap?.(operands.value);
  }
}

operators.expression = ExpressionOperator;
```

ДОДАТОК В

Елементи React:

App.jsx

```
import { Box, CssBaseline } from '@mui/material';
import ThemeManager from './components/ThemeManager';
import ExportProvider from './components/ExportProvider';
import Header from './components/Header';
import { DndProvider } from 'react-dnd';
import { HTML5Backend } from 'react-dnd-html5-backend';
import Router from './components/Router';
import ExpressionStorage from './components/ExpressionStorage';

const App = () => (
  <>
    <ThemeManager>
      <CssBaseline />
      <Header />
      <Box
        sx={{
          display: 'flex',
          width: '100%',
          alignItems: 'center',
          justifyContent: 'center',
          p: 1,
        }}
      >
        <ExportProvider>
          <DndProvider backend={HTML5Backend}>
            <ExpressionStorage>
              <Router />
            </ExpressionStorage>
          </DndProvider>
        </ExportProvider>
      </Box>
    </ThemeManager>
  </>
);

export default App;
```

Роутінг:

routes.js

```
import MainPage from './components/pages/MainPage';
import NotFoundPage from './components/pages/NotFoundPage';
import PolynomialPage from './components/pages/PolynomialPage';
import NumberTheoryPage from './components/pages/NumberTheoryPage';
import ExpressionPage from './components/pages/ExpressionPage';
import PlotPage from './components/pages/PlotPage/PlotPage';

const routes = [
  {
    path: '/',
    children: [
      {
        index: true,
```

```
        element: <MainPage />,
      },
      {
        path: 'builder/',
        element: <ExpressionPage />,
      },
      {
        path: 'polynomial/',
        element: <PolynomialPage />,
      },
      {
        path: 'number-theory/',
        element: <NumberTheoryPage />,
      },
      {
        path: 'plot/',
        element: <PlotPage />,
      },
    ],
  },
  {
    path: '*',
    element: <NotFoundPage />,
  },
];

export default routes;
```

Router.jsx

```
import { useRoutes } from 'react-router-dom';
import routes from '../routes';

const Router = () => {
  let element = useRoutes(routes);
  return element;
};

export default Router;
```

Компоненти:

Header.jsx

```
import { useContext, useMemo, useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import {
  AppBar,
  Box,
  Button,
  Container,
  IconButton,
  Toolbar,
  Menu,
  MenuItem,
  Tooltip,
  Typography,
} from '@mui/material';
import { Menu as MenuIcon } from '@mui/icons-material';
```

```
import LogoIcon from './shared/LogoIcon';
import ThemeModeContext from './shared/contexts/ThemeModeContext';

const LogoComplex = ({ display }) => {
  return (
    <>
      <LogoIcon fontSize="large" sx={{ display, mr: 1 }} />
      <Typography
        variant="h5"
        noWrap
        component={Link}
        to="/"
        sx={{
          mr: 2,
          display,
          flexGrow: 1,
          fontFamily: 'monospace',
          fontWeight: 700,
          letterSpacing: '.3rem',
          color: 'inherit',
          textDecoration: 'none',
        }}
      >
        ABACUS
      </Typography>
    </>
  );
};

const Header = () => {
  const [anchorElNav, setAnchorElNav] = useState(null);
  const { modes, activeMode, activateMode } = useContext(ThemeModeContext);
  const navigate = useNavigate();

  const pages = useMemo(
    () => [
      {
        path: 'builder',
        title: 'Builder',
      },
      {
        path: 'plot',
        title: 'Plot',
      },
      {
        path: 'polynomial',
        title: 'Polynomial',
      },
      {
        path: 'number-theory',
        title: 'Number theory',
      },
    ],
    []
  );

  const handleOpenNavMenu = (event) => {
    setAnchorElNav(event.currentTarget);
  };

  const handleCloseNavMenu = () => {
    setAnchorElNav(null);
  };
```

```
const getHandleButtonClick = (page) => () => {
  handleCloseNavMenu();
  navigate('/' + page);
};

return (
  <AppBar
    position="sticky"
    sx={{ zIndex: (theme) => theme.zIndex.drawer + 1 }}
  >
    <Container maxWidth="xl">
      <Toolbar disableGutters>
        <LogoComplex display={{ xs: 'none', md: 'flex' }} />
        <Box sx={{ flexGrow: 1, display: { xs: 'flex', md: 'none' } }}>
          <IconButton
            size="large"
            aria-label="account of current user"
            aria-controls="menu-appbar"
            aria-haspopup="true"
            onClick={handleOpenNavMenu}
            color="inherit"
          >
            <MenuIcon />
          </IconButton>
          <Menu
            id="menu-appbar"
            anchorEl={anchorElNav}
            anchorOrigin={{
              vertical: 'bottom',
              horizontal: 'left',
            }}
            keepMounted
            transformOrigin={{
              vertical: 'top',
              horizontal: 'left',
            }}
            open={Boolean(anchorElNav)}
            onClose={handleCloseNavMenu}
            sx={{
              display: { xs: 'block', md: 'none' },
            }}
          >
            {pages.map((page) => (
              <MenuItem
                key={page.path}
                onClick={getHandleButtonClick(page.path)}
              >
                <Typography textAlign="center">{page.title}</Typography>
              </MenuItem>
            ))}
          </Menu>
        </Box>

        <LogoComplex display={{ xs: 'flex', md: 'none' }} />

        <Box sx={{ flexGrow: 1, display: { xs: 'none', md: 'flex' } }}>
          {pages.map((page) => (
            <Button
              key={page.path}
              onClick={getHandleButtonClick(page.path)}
              sx={{ my: 2, color: 'white', display: 'block' }}
            >
```

```
                {page.title}
              </Button>
            ))}
          </Box>

          <Box sx={{ flexGrow: 0 }}>
            {modes.map((mode) => (
              <Tooltip key={mode.name} title={`${mode.label} mode`}>
                <IconButton
                  sx={{
                    color: mode.name === activeMode ? mode.color : undefined,
                  }}
                  onClick={() => activateMode(mode.name)}
                >
                  {mode.icon()}
                </IconButton>
              </Tooltip>
            ))}
          </Box>
        </Toolbar>
      </Container>
    </AppBar>
  );
};
export default Header;
```

ThemeManager.jsx

```
import { ThemeProvider, createTheme, useMediaQuery } from '@mui/material';
import { useCallback, useMemo, useState } from 'react';
import ThemeModeContext from './shared/contexts/ThemeModeContext';
import {
  LightMode as LightModeIcon,
  DarkMode as DarkModeIcon,
  BrightnessMedium as SystemModeIcon,
} from '@mui/icons-material';

const ThemeManager = ({ children }) => {
  const [mode, setMode] = useState(localStorage.getItem('theme') || 'system');
  const modes = useMemo(
    () => [
      {
        name: 'light',
        label: 'Light',
        color: '#ffe548',
        icon: () => <LightModeIcon />,
      },
      {
        name: 'dark',
        label: 'Dark',
        color: '#ffe548',
        icon: () => <DarkModeIcon />,
      },
      {
        name: 'system',
        label: 'System',
        color: '#2020ff',
        icon: () => <SystemModeIcon />,
      },
    ],
    []
  );
```

```
  const systemMode = useMediaQuery('(prefers-color-scheme: dark)')
    ? 'dark'
    : 'light';

  const theme = useMemo(
    () =>
      createTheme({
        palette: {
          mode: mode === 'system' ? systemMode : mode,
          primary: {
            main: '#00796b',
          },
          secondary: {
            main: '#e91e63',
          },
        },
      }),
    [mode, systemMode]
  );

  const activateMode = useCallback((newMode) => {
    localStorage.setItem('theme', newMode);
    setMode(newMode);
  }, []);

  return (
    <ThemeModeContext.Provider
      value={{ modes, activeMode: mode, activateMode }}
    >
      <ThemeProvider theme={theme}>{children}</ThemeProvider>
    </ThemeModeContext.Provider>
  );
};

export default ThemeManager;
```

ExportProvider.jsx

```
import { useCallback, useRef, useState } from 'react';
import * as htmlExport from 'html-to-image';
import fileSaver from 'file-saver';
import { Box, Button, Dialog, DialogActions, Tab, Tabs } from '@mui/material';
import { createTheme, ThemeProvider } from '@mui/material/styles';
import MathExpression from './expression/MathExpression';
import ExportExpressionContext from './shared/contexts/ExportExpressionContext';

const ExportProvider = ({ children }) => {
  const [expression, setExpression] = useState(null);
  const [mode, setMode] = useState('transparent');
  const [file, setFile] = useState('Png');
  const ref = useRef(null);

  const filetypes = ['Png', 'Jpeg', 'Svg'];
  const openExportModal = useCallback(
    (expression) => setExpression(expression),
    []
  );
  const handleClose = () => setExpression(null);
  const onModeChangeHandler = (e, newValue) => setMode(() => newValue);
  const onFileChangeHandler = (e, newValue) => setFile(() => newValue);

  const getButtonClickHandler = useCallback(
```

```
    (type) => () => {
      if (ref.current === null) {
        return;
      }

      htmlExport[`to${type}`](ref.current)
        .then((dataUrl) => {
          fileSaver.saveAs(dataUrl);
        })
        .catch((err) => {
          console.log(err);
        });
    },
    [ref]
  );

  return (
    <>
      <ExportExpressionContext.Provider value={{ openExportModal }}>
        {children}
      </ExportExpressionContext.Provider>
      <Dialog open={!!expression} onClose={handleClose} maxWidth={false}>
        <Tabs value={mode} onChange={onModeChangeHandler} centered>
          <Tab value="transparent" label="Transparent" />
          <Tab value="light" label="Light" />
          <Tab value="dark" label="Dark" />
        </Tabs>
        <Tabs value={file} onChange={onFileChangeHandler} centered>
          {filetypes.map((type) => (
            <Tab key={type} value={type} label={type} />
          ))}
        </Tabs>
        <ThemeProvider
          theme={createTheme({
            palette: {
              mode: mode === 'transparent' ? 'light' : mode,
            },
          })}
        >
          <Box>
            <Box
              ref={ref}
              sx={{
                width: 'auto',
                bgcolor:
                  mode === 'transparent' ? undefined : 'background.default',
                color: 'text.primary',
              }}
            >
              {expression && <MathExpression>{expression}</MathExpression>}
            </Box>
          </Box>
        </ThemeProvider>
        <DialogActions>
          <Button onClick={getButtonClickHandler(file)}>{'Save'}</Button>
          <Button onClick={handleClose}>{'Close'}</Button>
        </DialogActions>
      </Dialog>
    </>
  );
};

export default ExportProvider;
```

ExpressionStorage.jsx

```
import { useState } from 'react';
import ExpressionStorageContext from
'./shared/contexts/ExpressionStorageContext';

const ExpressionStorage = ({ children }) => {
  const [expressions, setExpressions] = useState([]);

  const edit = (index) => (editCallback) =>
    setExpressions((prev) => {
      let newArray = prev.slice();
      newArray.splice(index, 1, ...editCallback(newArray.at(index)));
      return newArray;
    });

  return (
    <ExpressionStorageContext.Provider
      value={{ expressions, editExpression: edit }}
    >
      {children}
    </ExpressionStorageContext.Provider>
  );
};

export default ExpressionStorage;
```

MainPage.jsx

```
import { useCallback, useMemo } from 'react';
import { useNavigate } from 'react-router-dom';
import {
  Box,
  Card,
  CardActionArea,
  CardContent,
  Stack,
  Typography,
  useTheme,
} from '@mui/material';
import Tex from '../tex/Tex';
import FunctionPlot from '../FunctionPlot';
import builderImage from '../../static/Builder.png';
import numberTheoryImage from '../../static/NumberTheory.png';

const MainPage = () => {
  const navigate = useNavigate();

  const width = 300;
  const height = (width * 3) / 4;

  const theme = useTheme();

  const pages = useMemo(
    () => [
      {
        path: 'builder',
        title: 'Builder',
        element: (
```

```
          <Box
            sx={{
              display: 'flex',
              justifyContent: 'center',
              alignItems: 'center',
            }}
          >
            <img src={builderImage} width={width} alt="builder" />
          </Box>
        ),
      },
      {
        path: 'plot',
        title: 'Plot',
        element: (
          <FunctionPlot
            settings={{ disableZoom: true, width, height }}
            lines={[{ fn: 'sin(x)', color: theme.palette.secondary.main }]}
          />
        ),
      },
      {
        path: 'polynomial',
        title: 'Polynomial',
        element: <Tex>{'\\Huge \\sum_{i=0}^n a_ix^i'}</Tex>,
      },
      {
        path: 'number-theory',
        title: 'Number theory',
        element: (
          <img src={numberTheoryImage} width={width} alt="number theory" />
        ),
      },
    ],
    [height, theme.palette.secondary.main]
  );

  const splitArrayInTwoHalves = useCallback((arrray) => {
    const middleIndex = Math.floor(arrray.length / 2);
    return [arrray.slice(0, middleIndex), arrray.slice(middleIndex)];
  }, []);

  return (
    <Box
      sx={{
        p: 5,
        width: '100%',
        height: '100%',
        justifyContent: 'center',
        alignItems: 'center',
      }}
    >
      <Stack
        direction={{ xs: 'column', md: 'row' }}
        gap={5}
        sx={{
          justifyContent: 'center',
          alignItems: { xs: 'center', md: 'start' },
          height: { xs: 2 / 3, md: 1 / 3 },
        }}
      >
        {splitArrayInTwoHalves(pages).map((half) => (
          <Stack gap={5} width={'100%'}>
```

```
            {half.map((page) => (
              <Card key={page.name} onClick={() => navigate(page.path + '/')}>
                <CardActionArea>
                  <CardContent sx={{ pointerEvents: 'none' }}>
                    <Stack
                      sx={{
                        display: 'flex',
                        justifyContent: 'center',
                        alignItems: 'center',
                      }}
                    >
                      <Typography variant="h4" sx={{ fontWeight: 'bold' }}>
                        {page.title.toUpperCase()}
                      </Typography>
                      {page.element}
                    </Stack>
                  </CardContent>
                </CardActionArea>
              </Card>
            ))}
          </Stack>
        ))}
      </Stack>
    </Box>
  );
};

export default MainPage;
```

ExpressionPage.jsx

```
import { Divider, Fab, IconButton, Stack } from '@mui/material';
import {
  AddOutlined as AddIcon,
  Close as CloseIcon,
} from '@mui/icons-material';
import ExpressionList from '../expression/ExpressionList';
import NumberBar from '../bars/NumberBar';
import SymbolBar from '../bars/SymbolBar';
import OperatorBar from '../bars/OperatorBar';
import ClippedDrawer from '../shared/ClippedDrawer';
import { useCallback, useContext, useState } from 'react';
import { apply, operators, replace, tex } from '../../mathkernel';
import Tex from '../tex/Tex';
import                          ExpressionStorageContext                          from
'../shared/contexts/ExpressionStorageContext';

const ExpressionPage = () => {
  const [selectionMode, setSelectionMode] = useState(null);
  const { expressions, editExpression } = useContext(ExpressionStorageContext);

  const startSelectionMode = (item) => {
    setSelectionMode({
      item,
      selectIsForbidden: (under) => apply(under, item).cause,
      add: (under) => {
        const applingInfo = apply(under, item);
        if (applingInfo.cause) {
          return under;
        } else {
          return {
```

```
            ...replace(under, applingInfo.replacer, applingInfo.condition),
          };
        }
      },
      stop: () => setSelectionMode(() => null),
    });
  };

  const renderCardHeader = useCallback(
    ({ expression, onEdit }) => (
      <IconButton
        disabled={!!selectionMode?.selectIsForbidden(expression)}
        onClick={() => onEdit((prev) => [selectionMode.add(prev)])}
      >
        <AddIcon />
      </IconButton>
    ),
    [selectionMode]
  );

  return (
    <Stack
      direction="row"
      gap={3}
      sx={{ width: '100%', height: '100%', justifyContent: 'center' }}
    >
      <Stack
        gap={2}
        sx={{
          display: 'flex',
          justifyContent: 'center',
          maxWidth: 1 / 2,
        }}
      >
        <ExpressionList
          expressions={expressions}
          onEdit={editExpression}
          selectionMode={!!selectionMode}
          renderCardHeader={selectionMode && renderCardHeader}
        />
      </Stack>
      <ClippedDrawer drawerWidth={5 / 12} anchor={'right'}>
        {selectionMode && (
          <>
            <Tex>
              {selectionMode.item.operator
                ? operators[selectionMode.item.operator].symbol()
                : tex(selectionMode.item)}
            </Tex>
            <Divider />
          </>
        )}
        <Stack gap={3} sx={{ maxWidth: '100%', overflow: 'auto', p: 2 }}>
          <NumberBar onClick={startSelectionMode} />
          <SymbolBar onClick={startSelectionMode} />
          <OperatorBar onClick={startSelectionMode} />
          {selectionMode && (
            <Fab color="error" onClick={selectionMode?.stop}>
              <CloseIcon />
            </Fab>
          )}
        </Stack>
      </ClippedDrawer>
```

```
    </Stack>
  );
};

export default ExpressionPage;
```

PolynomialPage.jsx

```
import { useMemo, useState } from 'react';
import {
  ButtonGroup,
  Stack,
  Checkbox as CheckBox,
  Tooltip,
} from '@mui/material';
import { getEmpty, operators, polynomial } from '../../../mathkernel';
import {
  add,
  substract,
  multiply,
  divide,
  modulo,
} from '../../../algorithms/polynomialOperations';
import TexButton from '../../tex/TexButton';
import PolynomialColumn from './PolynomialColumn';
import {
  UploadFileOutlined as UploadFileIcon,
  KeyboardAltOutlined as KeyboardIcon,
  VisibilityOutlined as VisibilityOnOutlinedIcon,
  VisibilityOffOutlined as VisibilityOffOutlinedIcon,
} from '@mui/icons-material/';

const PolynomialPage = () => {
  const [left, setLeft] = useState(() => getEmpty());
  const [right, setRight] = useState(() => getEmpty());
  const [result, setResult] = useState(() => getEmpty());
  const [fileInput, setFileInput] = useState(false);
  const [hide, setHide] = useState(false);

  const polynomialOperations = useMemo(
    () => ({
      add: add,
      sub: substract,
      mul: multiply,
      div: divide,
      mod: modulo,
    }),
    []
  );

  return (
    <Stack
      direction={'column'}
      gap={2}
      sx={{ width: '100%', height: '100%', alignItems: 'center' }}
    >
      <Stack
        direction={'row'}
        gap={2}
        sx={{ justifyContent: 'center', alignItems: 'center' }}
      >
        <Tooltip title={'Load polynomials from file'} placement="bottom">
```

```
      <CheckBox
        icon={<KeyboardIcon />}
        checkedIcon={<UploadFileIcon />}
        checked={fileInput}
        onChange={(event) => setFileInput(event.target.checked)}
      />
    </Tooltip>

    <Tooltip title={'Hide polynomials representation'} placement="bottom">
      <CheckBox
        icon={<VisibilityOnOutlinedIcon />}
        checkedIcon={<VisibilityOffOutlinedIcon />}
        checked={hide}
        onChange={(event) => setHide(event.target.checked)}
      />
    </Tooltip>
  </Stack>
  <Stack
    direction={'row'}
    gap={2}
    sx={{ width: 1, justifyContent: 'center', height: '100%' }}
  >
    <PolynomialColumn
      value={left}
      onChange={setLeft}
      fileInput={fileInput}
      hide={hide}
      width={1 / 3}
    />

    <ButtonGroup orientation="vertical">
      {polynomial.map((key) => (
        <TexButton
          key={key}
          onClick={() =>
            setResult(() => {
              try {
                if (!left.vector || !right.vector) {
                  throw new Error('Empty polynomial recieved');
                }
                if (
                  (key === 'div' || key === 'mod') &&
                  right.vector.every((item) => item === 0)
                ) {
                  throw new Error('Division by zero polynomial');
                }
                return {
                  name: 'polynomial',
                  vector: polynomialOperations[key](
                    left.vector,
                    right.vector
                  ),
                };
              } catch (error) {
                return {
                  name: 'error',
                  message: error.message,
                };
              }
            })
          }
        >
          {operators[key].symbol()}
```

```
            </TexButton>
          ))}
        </ButtonGroup>

        <PolynomialColumn
          value={right}
          onChange={setRight}
          fileInput={fileInput}
          hide={hide}
          width={1 / 3}
        />
      </Stack>
      <PolynomialColumn value={result} hide={hide} width={2 / 3} />
    </Stack>
  );
};

export default PolynomialPage;
```

PolynomialColumn.jsx

```
import { Stack } from '@mui/material';
import ExpressionCard from '../../expression/ExpressionCard';
import PolynomialInput from './PolynomialInput';

const PolynomialColumn = ({ value, onChange, fileInput, hide, width }) => (
  <Stack direction={'column'} gap={2} sx={{ width }}>
    {onChange && (
      <PolynomialInput
        inputType={fileInput ? 'file' : 'string'}
        value={value}
        onChange={(newValue) => onChange(() => newValue)}
      />
    )}
    <ExpressionCard expression={value} hide={hide} disableDrag />
  </Stack>
);

export default PolynomialColumn;
```

PolynomialInput.jsx

```
import { useMemo } from 'react';
import { TextField } from '@mui/material';
import { readFileAsJSON } from '../../../readFile';

const PolynomialInput = ({ inputType, value, onChange }) => {
  const processValue = useMemo(
    () => ({
      string: (target) => {
        const vector = target.value
          .trim()
          .split(' ')
          .reverse()
          .map((item) => +item);

        onChange(
          vector.some((item) => isNaN(item))
            ? { name: 'error', message: 'Only numbers accept' }
            : { name: 'polynomial', vector }
```

```
        );
      },

      file: (target) => {
        onChange({ name: 'loading' });
        readFileAsJSON(target.files[0])
          .then((jsonData) => {
            if (Array.isArray(jsonData)) {
              onChange({ name: 'polynomial', vector: jsonData });
            } else if (Array.isArray(jsonData.vector)) {
              onChange(jsonData);
            } else throw new Error();
          })
          .catch((error) => {
            onChange({ name: 'error', message: error.message });
          });
      },
    }),
    [onChange]
  );

  return (
    <>
      {inputType && (
        <TextField
          fullWidth
          type={inputType}
          onChange={(event) => processValue[inputType](event.target)}
        />
      )}
    </>
  );
};

export default PolynomialInput;
```

NumberTheoryPage.jsx

```
import { useCallback, useState } from 'react';
import { Box, Divider, Grid, List } from '@mui/material';
import fibonacci, { fibonacciSum } from '../../../algorithms/fibonacci';
import NestedListItem from '../../shared/NestedListItem';
import NumberCard from './NumberCard';
import ClippedDrawer from '../../shared/ClippedDrawer';
import { gcd, lcm, reduceFractions } from '../../../algorithms/divisor';

const NumberTheoryPage = () => {
  const [cards, setCards] = useState([]);

  const addCard = useCallback((card) => {
    setCards((prev) => prev.concat(card));
  }, []);

  const editCard = (index) => (editCallback) =>
    setCards((prev) =>
      prev.map((item, i) => (index !== i ? item : editCallback(item)))
    );

  const removeCard = (index) => () =>
    setCards((prev) => prev.filter((item, i) => index !== i));
```

```
const sections = [
  {
    name: 'fibonacci',
    label: 'Fibonacci',
    children: [
      {
        name: 'nth',
        label: 'Fibonacci nth number',
        onClickHandler: () => {
          addCard({
            calculate: fibonacci,
            onResult: (result) => `${result}`,
            onTitleRender: (numbers) =>
              `${numbers ? numbers[0] : 'n'}th Fibonacci number`,
          });
        },
      },
      {
        name: 'nthSum',
        label: 'Sum of n Fibonacci numbers',
        onClickHandler: () => {
          addCard({
            calculate: fibonacciSum,
            onResult: (result) => `${result}`,
            onTitleRender: (numbers) =>
              `Sum of ${numbers ? numbers[0] : 'n'} Fibonacci numbers`,
          });
        },
      },
    ],
  },
  {
    name: 'divisor',
    label: 'Divisor operations',
    children: [
      {
        name: 'gcd',
        label: 'Great common divisor',
        onClickHandler: () => {
          addCard({
            calculate: gcd,
            onResult: (result) => `${result}`,
            onTitleRender: (numbers) =>
              'gcd' + (numbers ? `(${numbers[0]}, ${numbers[1]})` : ''),
          });
        },
      },
      {
        name: 'lcm',
        label: 'Least common multiple',
        onClickHandler: () => {
          addCard({
            calculate: lcm,
            onResult: (result) => `${result}`,
            onTitleRender: (numbers) =>
              'lcm' + (numbers ? `(${numbers[0]}, ${numbers[1]})` : ''),
          });
        },
      },
      {
        name: 'reduceFractions',
        label: 'Reduce fractions',
        onClickHandler: () => {
```

```
            addCard({
              calculate: reduceFractions,
              onResult: (result) =>
                `${result.numerator} / ${result.denominator}`,
              onTitleRender: (numbers) =>
                numbers ? `${numbers[0]} / ${numbers[1]}` : 'Reduce fractions',
            });
          },
        },
      ],
    },
  ];

  return (
    <>
      <ClippedDrawer>
        <Divider />
        <Box sx={{ overflow: 'auto', p: 1 }}>
          <List sx={{ width: '100%' }}>
            {sections.map((section) => (
              <NestedListItem key={section.name} section={section} />
            ))}
          </List>
        </Box>
      </ClippedDrawer>

      <Grid container spacing={2}>
        {cards
          .map((card, index) => (
            <Grid key={index} item zeroMinWidth>
              <NumberCard
                key={index}
                {...card}
                onEdit={editCard(index)}
                onClose={removeCard(index)}
              />
            </Grid>
          ))
          .reverse()}
      </Grid>
    </>
  );
};

export default NumberTheoryPage;
```

NumberCard.jsx

```
import {
  Card,
  CardActions,
  CardContent,
  IconButton,
  LinearProgress,
  Stack,
  Tooltip,
  Typography,
} from '@mui/material';
import {
  ContentCopy as CopyIcon,
  Close as CloseIcon,
  FileDownloadOutlined as FileDownloadIcon,
```

```
} from '@mui/icons-material';
import copyToClipboard from 'copy-to-clipboard';
import fileSaver from 'file-saver';
import asynchronize from '../../../asynchronize';
import NumberForm from './NumberForm';

const NumberCard = ({
  numbers,
  result,
  error,
  calculate,
  onEdit,
  onTitleRender,
  onResult,
  onClose,
}) => {
  const title = onTitleRender?.(numbers);

  let content = null;
  if (!numbers && !result && !error) {
    content = (
      <NumberForm
        amount={calculate.length}
        onSubmit={(values) => {
          onEdit((prev) => ({ ...prev, numbers: values }));
          asynchronize(calculate)(...values).then(
            (result) =>
              onEdit((prev) => ({ ...prev, result: onResult(result) })),
            (error) => onEdit((prev) => ({ ...prev, error: error.message }))
          );
        }}
      />
    );
  } else if (numbers && !result && !error) {
    content = <LinearProgress />;
  } else {
    content = result || error;
  }

  const contentIsString = typeof content === 'string';

  const actions = [
    {
      name: 'copy',
      title: 'Copy to clipboard',
      disabled: !contentIsString,
      onClickHandler: () => copyToClipboard(content),
      onIconRender: () => <CopyIcon />,
    },
    {
      name: 'download',
      title: 'Download',
      disabled: !contentIsString,
      onClickHandler: () =>
        fileSaver.saveAs(
          new Blob([content], { type: 'text/plain;charset=utf-8' }),
          title || undefined
        ),
      onIconRender: () => <FileDownloadIcon />,
    },
    {
      name: 'close',
      title: 'Close',
```

```
      onClickHandler: onClose,
      onIconRender: () => <CloseIcon />,
    },
  ];

  return (
    <Card>
      <CardActions sx={{ justifyContent: 'space-between' }}>
        {actions.map((action) => (
          <Tooltip key={action.name} title={action.title} placement="bottom">
            <span>
              <IconButton
                onClick={action.onClickHandler}
                disabled={action.disabled}
              >
                {action.onIconRender()}
              </IconButton>
            </span>
          </Tooltip>
        ))}
      </CardActions>
      <CardContent>
        <Stack gap={2}>
          <Typography variant="h6" noWrap>
            {title}
          </Typography>
          {contentIsString ? (
            <Typography sx={{ wordBreak: 'break-word' }}>{content}</Typography>
          ) : (
            content
          )}
        </Stack>
      </CardContent>
    </Card>
  );
};

export default NumberCard;
```

NumberForm.jsx

```
import { useMemo } from 'react';
import { object, number } from 'yup';
import { Button, Stack } from '@mui/material';
import NumberTextField from '../../shared/NumberTextField';
import { useState } from 'react';

const NumberForm = ({
  amount,
  onSubmit = () => {},
  submitLabel = 'Submit',
}) => {
  const [values, setValues] = useState(new Array(amount).fill(''));

  const schema = useMemo(() => object({ number: number().required() }), []);

  const isValid = useMemo(
    () =>
      values.every((value) =>
        schema.isValidSync({ name: 'number', number: value })
      ),
    [values, schema]
```

```
  );

  const edit = (index, editCallback) =>
    setValues((prev) =>
      prev.map((item, i) => (index !== i ? item : editCallback(item)))
    );

  const onSubmitHandler = (event) => {
    event.preventDefault();
    if (isValid) {
      onSubmit(
        values.map(
          (value) =>
            schema.validateSync({ name: 'number', number: value }).number
        )
      );
    }
  };

  return (
    <form onSubmit={onSubmitHandler}>
      <Stack gap={2} sx={{ display: 'flex', justifyContent: 'center' }}>
        {values.map((item, index) => {
          const itemIsValid =
            !item || schema.isValidSync({ name: 'number', number: item });
          return (
            <NumberTextField
              key={index}
              value={item}
              onChange={(event) => edit(index, () => event.target.value)}
              error={!itemIsValid}
              helperText={!itemIsValid && 'Invalid input'}
            />
          );
        })}
        <Button type="submit" disabled={!isValid}>
          {submitLabel}
        </Button>
      </Stack>
    </form>
  );
};

export default NumberForm;
```

PlotPage.jsx

```
import { useCallback, useContext, useMemo, useState } from 'react';
import FunctionPlot from '../../FunctionPlot';
import { Checkbox, Stack, Tooltip } from '@mui/material';
import {
  LineAxis as PlotOnIcon,
  ShowChart as PlotOffIcon,
} from '@mui/icons-material';
import { areEqual, getVariables, hasEmpty, plot } from '../../../mathkernel';
import ExpressionList from '../../expression/ExpressionList';
import ClippedDrawer from '../../shared/ClippedDrawer';
import                        ExpressionStorageContext                        from
'../../shared/contexts/ExpressionStorageContext';
import LineSettings from './LineSettings';
import PlotSettings from './PlotSettings';
```

```
const PlotPage = () => {
  const { expressions, editExpression } = useContext(ExpressionStorageContext);
  const [settings, setSettings] = useState({ tip: {}, xAxis: {}, yAxis: {} });

  const onErrorHandler = useCallback((error) => console.error(error), []);

  const plots = useMemo(
    () =>
      expressions
        .filter((expression) => !!expression.plot)
        .map((expression) => {
          if (expression.plot.variables.length === 2) {
            return {
              fn: plot(expression, {
                replace: (operands) => {
                  if (areEqual(operands, expression.plot.variables[0])) {
                    return 'x';
                  }
                  if (areEqual(operands, expression.plot.variables[1])) {
                    return 'y';
                  }
                  throw new Error('Expect this as unreacheble');
                },
              }),
              ...expression.plot,
              fnType: 'implicit',
            };
          }

          const graphData = { ...expression.plot };
          if (expression.plot.fnType === 'polar') {
            graphData.r = plot(expression, { replace: () => 'theta' });
            if (graphData.graphType !== 'scatter') {
              graphData.graphType = 'polyline';
            }
          } else {
            graphData.fn = plot(expression, { replace: () => 'x' });
            if (graphData.graphType !== 'scatter') {
              graphData.graphType = 'interval';
            }
          }
          return graphData;
        }),
    [expressions]
  );

  return (
    <Stack
      direction="row"
      gap={3}
      sx={{ width: '100%', height: '100%', justifyContent: 'center' }}
    >
      <ClippedDrawer drawerWidth={7 / 12}>
        <Stack
          direction="row"
          sx={{
            display: 'flex',
            justifyContent: 'center',
            alignItems: 'center',
            overflow: 'auto',
            p: 2,
          }}
        >
```

```
      <FunctionPlot
        settings={settings}
        lines={plots}
        onError={onErrorHandler}
      />
      <PlotSettings
        direction={'column'}
        settings={settings}
        onEdit={setSettings}
      />
    </Stack>
  </ClippedDrawer>

  <Stack
    gap={3}
    sx={{
      display: 'flex',
      justifyContent: 'center',
      maxWidth: 1 / 3,
    }}
  >
    <ExpressionList
      expressions={expressions}
      onEdit={editExpression}
      renderCardHeader={({ expression, onEdit }) => {
        const variables = hasEmpty(expression)
          ? null
          : getVariables(expression);
        const canBuildPlot = !!variables && variables.length <= 2;

        return (
          <Tooltip
            title={
              !canBuildPlot
                ? 'Cannot build plot by this expression'
                : expression.plot
                ? 'Remove from the chart'
                : 'Add to the chart'
            }
            placement="left"
          >
            <span>
              <Checkbox
                icon={<PlotOffIcon />}
                checkedIcon={<PlotOnIcon />}
                checked={!!expression.plot}
                disabled={!canBuildPlot}
                onChange={(event) => {
                  onEdit(({ plot, ...prev }) => {
                    if (event.target.checked) {
                      prev.plot = { variables };
                    }
                    return [prev];
                  });
                }}
              />
            </span>
          </Tooltip>
        );
      }}
      renderAddition={({ expression, index, onEdit }) => (
        <LineSettings
          key={index}
```

```
              direction="column"
              expression={expression}
              onEdit={onEdit}
            />
          )}
          additionPosition="left"
        />
      </Stack>
    </Stack>
  );
};

export default PlotPage;
```

PlotSettings.jsx

```
import { Checkbox, Stack, Tooltip } from '@mui/material';
import {
  GridOffOutlined as GridOffIcon,
  GridOnOutlined as GridOnIcon,
  ZoomInMapOutlined as ZoomIcon,
  BrowserNotSupportedOutlined as ZoomOffIcon,
  MultipleStopOutlined as TipIcon,
} from '@mui/icons-material';

const PlotSettings = ({ direction, settings, onEdit }) => {
  const settingsActions = [
    {
      name: 'grid',
      title: 'Show grid',
      checkedTitle: 'Hide grid',
      icon: <GridOffIcon />,
      checkedIcon: <GridOnIcon />,
    },
    {
      name: 'disableZoom',
      title: 'Disable zoom',
      checkedTitle: 'Enable zoom',
      icon: <ZoomIcon />,
      checkedIcon: <ZoomOffIcon />,
    },
    {
      name: 'xLine',
      title: 'Show x-axis tip',
      checkedTitle: 'Hide x-axis tip',
      icon: <TipIcon sx={{ transform: 'rotate(90deg)' }} />,
      checkedIcon: <TipIcon sx={{ transform: 'rotate(90deg)' }} />,
      additionalKey: 'tip',
    },
    {
      name: 'yLine',
      title: 'Show y-axis tip',
      checkedTitle: 'Hide y-axis tip',
      icon: <TipIcon />,
      checkedIcon: <TipIcon />,
      additionalKey: 'tip',
    },
  ];

  const getOnSettingChangeHandler = (setting, additionalKey) => (event) => {
    onEdit((prev) => {
      let containing = additionalKey ? prev[additionalKey] : prev;
```

```
      containing[setting] = event.target.checked;
      prev[setting] = event.target.checked;
      return { ...prev };
    });
  };

  return (
    <Stack
      direction={direction}
      gap={1}
      sx={{ display: 'flex', alignItems: 'center' }}
    >
      {settingsActions.map((action) => {
        let containing = action.additionalKey
          ? settings[action.additionalKey]
          : settings;

        return (
          <Tooltip
            key={action.name}
            title={containing[action.name] ? action.checkedTitle : action.title}
            placement="right"
          >
            <Checkbox
              color="secondary"
              icon={action.icon}
              checkedIcon={action.checkedIcon}
              checked={!!containing[action.name]}
              onChange={getOnSettingChangeHandler(
                action.name,
                action.additionalKey
              )}
            />
          </Tooltip>
        );
      })}
    </Stack>
  );
};

export default PlotSettings;
```

LineSettings.jsx

```
import { Checkbox, Stack, Tooltip } from '@mui/material';
import {
  FormatPaintRounded as FillOn,
  FormatPaintOutlined as FillOff,
  LocationSearchingOutlined as SkipTipOffIcon,
  LocationDisabledOutlined as SkipTipOnIcon,
  ScatterPlot as ScatterIcon,
  PolylineRounded as PolylineIcon,
  Radar as PolarIcon,
  GridGoldenratio as CartesianIcon,
} from '@mui/icons-material';
import ColorPicker from '../../shared/ColorPicker';

const LineSettings = ({ direction, expression, onEdit }) => {
  const settingsActions = [
    {
```

```
    name: 'closed',
    title: 'Fill area under graph',
    checkedTitle: 'Do not fill area under graph',
    icon: <FillOff />,
    checkedIcon: <FillOn />,
    isAvailable: (plot) => plot.variables.length <= 2,
    isChecked: (item) => !!item,
    setter: (checked) => checked,
  },
  {
    name: 'graphType',
    title: 'Scatter graph type',
    checkedTitle: 'Polyline graph type',
    icon: <PolylineIcon />,
    checkedIcon: <ScatterIcon />,
    isAvailable: (plot) => plot.variables.length <= 1,
    isChecked: (item) => item === 'scatter',
    setter: (checked) => (checked ? 'scatter' : 'interval'),
  },
  {
    name: 'fnType',
    title: 'Polar coordinates',
    checkedTitle: 'Cartesian coordinates',
    icon: <CartesianIcon />,
    checkedIcon: <PolarIcon />,
    isAvailable: (plot) => plot.variables.length <= 1,
    isChecked: (item) => item === 'polar',
    setter: (checked) => (checked ? 'polar' : 'linear'),
  },
  {
    name: 'skipTip',
    title: 'Skip tip for this graph',
    checkedTitle: 'Do not skip tip for this graph',
    icon: <SkipTipOffIcon />,
    checkedIcon: <SkipTipOnIcon />,
    isAvailable: (plot) => plot.variables.length <= 1,
    isChecked: (item) => !!item,
    setter: (checked) => checked,
  },
];

const editPlotSettings = (editCallback) =>
  onEdit((prev) => [{ ...prev, plot: editCallback(prev.plot) }]);

return (
  <Stack
    direction={'row'}
    gap={1}
    sx={{ display: 'flex', alignItems: 'center' }}
  >
    {!!expression.plot && (
      <Tooltip title={'Choose graph color'} placement="left">
        <span>
          <ColorPicker
            color={expression.plot?.color}
            onChange={(color) =>
              editPlotSettings((prev) => ({ ...prev, color }))
            }
          />
        </span>
      </Tooltip>
    )}
    <Stack
```

```
          direction={direction}
          gap={1}
          sx={{ display: 'flex', alignItems: 'center' }}
        >
          {!!expression.plot && (
            <>
              {settingsActions
                .filter((action) => action.isAvailable(expression.plot))
                .map((action) => {
                  return (
                    <Tooltip
                      key={action.name}
                      title={
                        action.isChecked(expression.plot?.[action.name])
                          ? action.checkedTitle
                          : action.title
                      }
                      placement="left"
                    >
                      <Checkbox
                        color="secondary"
                        icon={action.icon}
                        checkedIcon={action.checkedIcon}
                        checked={action.isChecked(expression.plot?.[action.name])}
                        onChange={(event) =>
                          editPlotSettings((prev) => ({
                            ...prev,
                            [action.name]: action.setter(event.target.checked),
                          }))
                        }
                      />
                    </Tooltip>
                  );
                })}
            </>
          )}
        </Stack>
      </Stack>
    );
};

export default LineSettings;
```

NotFoundPage.jsx

```
import { Stack, Typography } from '@mui/material';

const NotFoundPage = () => {
  return (
    <Stack>
      <Typography variant="h2" component="h1" gutterBottom>
        {'Oops! Page Not Found'}
      </Typography>
      <Typography variant="h4">
        {'The page you are looking for does not exist.'}
      </Typography>
    </Stack>
  );
};

export default NotFoundPage;
```

BaseFormBar.jsx

```
import { useMemo } from 'react';
import { ValidationError } from 'yup';
import TexButton from '../tex/TexButton';
import { ButtonGroup, Stack } from '@mui/material';
import { tex } from '../../mathkernel';

const BaseFormBar = ({
  values,
  schema,
  onClick,
  mainValue = true,
  children,
  direction = 'row',
}) => {
  const validated = useMemo(() => {
    try {
      return schema.validateSync(values);
    } catch (error) {
      if (error instanceof ValidationError) {
        return null;
      }
      throw error;
    }
  }, [schema, values]);

  return (
    <form
      onSubmit={(event) => {
        event.preventDefault();
        if (validated) {
          onClick({ name: 'expression', value: validated });
        }
      }}
    >
      <Stack
        gap={1}
        direction={direction}
        sx={{ display: 'flex', alignItems: 'center' }}
      >
        {children}
        <ButtonGroup>
          <TexButton
            type="submit"
            color={!validated ? 'error' : undefined}
            disabled={!mainValue}
            draggable={
              validated && {
                name: 'expression',
                value: validated,
              }
            }
          >
            {`${validated ? tex(validated) : '!'}`}
          </TexButton>
        </ButtonGroup>
      </Stack>
    </form>
  );
};
export default BaseFormBar;
```

NumberBar.jsx

```
import { useMemo, useState } from 'react';
import { object, number } from 'yup';
import NumberTextField from '../shared/NumberTextField';
import BaseFormBar from './BaseFormBar';

const NumberBar = ({ onClick }) => {
  const [value, setValue] = useState('');

  const schema = useMemo(() => object({ number: number().required() }), []);

  const values = useMemo(() => ({ name: 'number', number: value }), [value]);

  return (
    <BaseFormBar
      values={values}
      schema={schema}
      onClick={onClick}
      mainValue={value}
    >
      <NumberTextField
        size="small"
        label={'Number'}
        value={value}
        onChange={(event) => setValue(() => event.target.value)}
      />
    </BaseFormBar>
  );
};

export default NumberBar;
```

SymbolBar.jsx

```
import { useMemo, useState } from 'react';
import {
  Box,
  Stack,
  TextField,
  ToggleButton,
  ToggleButtonGroup,
} from '@mui/material';
import { number, object, string } from 'yup';
import { operators, tex } from '../../mathkernel';
import Tex from '../tex/Tex';
import NumberTextField from '../shared/NumberTextField';
import BaseFormBar from './BaseFormBar';

const SymbolBar = ({ onClick }) => {
  const [variable, setVariable] = useState('');
  const [accent, setAccent] = useState('none');
  const [index, setIndex] = useState('');

  const accents = useMemo(() => operators.variable.getAccents(), []);
  const variablePattern = useMemo(() => operators.variable.getPattern(), []);

  const schema = useMemo(
    () =>
      object({
        variable: string().required().matches(variablePattern),
```

```
        index: number().integer().positive().nullable(true),
      }),
    [variablePattern]
  );

  const values = useMemo(
    () => ({
      name: 'variable',
      variable,
      index: index || undefined,
      accent: accent !== 'none' ? accent : undefined,
    }),
    [variable, index, accent]
  );

  return (
    <BaseFormBar
      values={values}
      schema={schema}
      onClick={onClick}
      mainValue={variable}
    >
      <Stack gap={1}>
        {/* <Box sx={{ display: 'flex', width: '100%', overflow: 'auto',
justifyContent: '' }}> */}
        <ToggleButtonGroup
          value={accent}
          color="primary"
          exclusive
          onChange={(event, newAccent) => setAccent(newAccent)}
        >
          {['none'].concat(accents).map((item) => (
            <ToggleButton
              key={item}
              value={item}
              sx={{ textTransform: 'none' }}
            >
              <Tex displayMode={false}>
                {tex({
                  name: 'variable',
                  variable: 'x',
                  accent: item !== 'none' ? item : undefined,
                })}
              </Tex>
            </ToggleButton>
          ))}
        </ToggleButtonGroup>

        <Stack
          gap={1}
          direction={'row'}
          sx={{ display: 'flex', alignItems: 'center' }}
        >
          <TextField
            label={'Variable'}
            value={variable}
            onChange={(event) => setVariable(() => event.target.value)}
          />
          <Box sx={{ display: 'flex', height: '100%', alignItems: 'flex-end' }}>
            <NumberTextField
              size="small"
              label={'Index'}
              value={index}
```

```
              onChange={(event) => setIndex(() => event.target.value)}
            />
          </Box>
        </Stack>
      </Stack>
    </BaseFormBar>
  );
};
export default SymbolBar;
```

OperatorBar.jsx

```
import { ButtonGroup, Stack } from '@mui/material';
import TexButton from '../tex/TexButton';
import {
  operators,
  arithmetic,
  comparison,
  inverseTrigonometric,
  power,
  trigonometric,
} from '../../mathkernel';

const OperatorBar = ({ onClick }) => {
  return (
    <Stack gap={1}>
      {[arithmetic, power, trigonometric, inverseTrigonometric, comparison].map(
        (set, index) => (
          <ButtonGroup key={index}>
            {set.map((key) => {
              const operator = { operator: key };
              return (
                <TexButton
                  key={key}
                  draggable={operator}
                  onClick={() => onClick(operator)}
                >
                  {operators[key].symbol()}
                </TexButton>
              );
            })}
          </ButtonGroup>
        )
      )}
    </Stack>
  );
};

export default OperatorBar;
```

Tex.jsx

```
import 'katex/dist/katex.min.css';
import { renderToString } from 'katex';

const Tex = ({ children, displayMode = true }) => {
  return (
    <span
      dangerouslySetInnerHTML={{
        __html: renderToString(children, {
          displayMode,
          trust: true,
```

```
          strict: (errorCode) =>
            errorCode === 'htmlExtension' ? 'ignore' : 'error',
        }),
      }}
    />
  );
};

export default Tex;
```

TexButton.jsx

```
import { Button } from '@mui/material';
import { useDrag } from 'react-dnd';
import Tex from './Tex';

const TexButton = ({ children, draggable, sx = {}, ...props }) => {
  const [collectedDrag, drag, dragPreview] = useDrag(
    () => ({
      type: 'operator',
      item: () => draggable,
      collect: (monitor) => ({
        isDragging: monitor.isDragging(),
      }),
    }),
    [children]
  );

  sx.textTransform = 'none';
  return (
    <Button sx={sx} {...props}>
      {draggable ? (
        <div ref={collectedDrag.isDragging ? dragPreview : drag}>
          <Tex displayMode={false}>{children}</Tex>
        </div>
      ) : (
        <Tex displayMode={false}>{children}</Tex>
      )}
    </Button>
  );
};

export default TexButton;
```

TexIcon.jsx

```
import { Box } from '@mui/material';
import Tex from './Tex';

const TexIcon = ({ tex }) => {
  return (
    <Box
      sx={{
        display: 'flex',
```

```
        alignItems: 'end',
        justifyContent: 'center',
        width: '24px',
        height: '24px',
      }}
    >
      <Tex displayMode={false}>{`\\tiny ${tex}`}</Tex>
    </Box>
  );
};

export default TexIcon;
```

MathExpression.jsx

```
import { tex } from '../../mathkernel';
import Tex from '../tex/Tex';

const MathExpression = ({ children, ...props }) => (
  <Tex>{tex(children, props)}</Tex>
);
export default MathExpression;
```

DraggableMathExpression.jsx

```
import { useTheme } from '@mui/material';
import { useDrag, useDrop } from 'react-dnd';
import { apply, replace } from '../../mathkernel';
import MathExpression from './MathExpression';

const DraggableMathExpression = ({ children: expression, onEdit }) => {
  const [collectedDrag, drag, dragPreview] = useDrag(
    () => ({
      type: 'math',
      item: () => expression,
      collect: (monitor) => ({
        isDragging: monitor.isDragging(),
      }),
    }),
    [expression]
  );

  const [collectedDrop, drop] = useDrop(() => ({
    accept: ['math', 'operator'],
    drop: (item) => {
      onEdit((prev) => {
        const applingInfo = apply(prev, item);
        if (applingInfo.cause) {
          return [prev];
        } else {
          return [
            {
              ...replace(prev, applingInfo.replacer, applingInfo.condition),
            },
          ];
        }
      });
    },
    collect: (monitor) => ({
      isOver: monitor.isOver(),
```

```
      overingItem: monitor.getItem(),
    }),
  }));

  const theme = useTheme();
  const secondaryColor = theme.palette.secondary.main;

  const onHangHandler = collectedDrop.isOver
    ? apply(expression, collectedDrop.overingItem).condition
    : undefined;

  return (
    <div ref={drop}>
      <div ref={collectedDrag.isDragging ? dragPreview : drag}>
        <MathExpression
          highlight={onHangHandler}
          highlightColor={secondaryColor}
        >
          {expression}
        </MathExpression>
      </div>
    </div>
  );
};

export default DraggableMathExpression;
```

ExpressionAlert.jsx

```
import { useMemo } from 'react';
import { Alert, CircularProgress } from '@mui/material';

const ExpressionAlert = ({ type, message }) => {
  const alertInfos = useMemo(
    () => ({
      loading: {
        severity: 'warning',
        message: 'Loading is in progress',
        icon: <CircularProgress color="warning" size={20} sx={{ m: '1px' }} />,
      },
      error: {
        severity: 'error',
        message: 'Error occured',
      },
      empty: {
        severity: 'info',
        message: 'Expression is empty',
      },
      default: {
        severity: 'success',
        message: 'Expression is ready',
      },
    }),
    []
  );

  const currentInfo = useMemo(
    () => alertInfos[type] || alertInfos.default,
    [type, alertInfos]
  );

  return (
```

```
    <Alert severity={currentInfo.severity} icon={currentInfo.icon}>
      {message || currentInfo.message}
    </Alert>
  );
};

export default ExpressionAlert;
```

Expression.jsx

```
import { isValidOperator } from '../../mathkernel';
import DraggableMathExpression from './DraggableMathExpression';
import ExpressionAlert from './ExpressionAlert';
import MathExpression from './MathExpression';

const Expression = ({ expression, onEdit, disableDrag, hide }) => (
  <>
    {isValidOperator(expression.name) && !hide ? (
      disableDrag ? (
        <MathExpression>{expression}</MathExpression>
      ) : (
        <DraggableMathExpression onEdit={onEdit}>
          {expression}
        </DraggableMathExpression>
      )
    ) : (
      <ExpressionAlert type={expression.name} message={expression.message} />
    )}
  </>
);

export default Expression;
```

ExpressionTitle.jsx

```
import { Dialog, TextField, Typography } from '@mui/material';

const ExpressionTitle = ({ title, isEditing, onChange, onClose }) => (
  <>
    <Typography variant="h5" noWrap>
      {title}
    </Typography>
    <Dialog open={isEditing} onClose={onClose} maxWidth={false}>
      <form
        onSubmit={(event) => {
          event.preventDefault();
          event.stopPropagation();
          onClose();
        }}
      >
        <TextField
          value={title}
          onChange={(event) => onChange(() => event.target.value)}
          label="Title"
          autoFocus
          sx={{ m: 1 }}
        />
      </form>
    </Dialog>
  </>
);
```

```
export default ExpressionTitle;
```

ExpressionCard.jsx

```
import { useContext, useMemo, useState } from 'react';
import {
  Box,
  Card,
  CardActions,
  CardContent,
  IconButton,
  Stack,
  Tooltip,
  Typography,
} from '@mui/material';
import {
  ContentCopy as CopyIcon,
  Delete as DeleteIcon,
  ImportExport as ExportIcon,
  ChangeCircle as ChangeCircleIcon,
  FileDownloadOutlined as FileDownloadIcon,
  Title as TitleIcon,
  Calculate as CalculateIcon,
  Filter2 as DoubleIcon,
} from '@mui/icons-material';
import copyToClipboard from 'copy-to-clipboard';
import fileSaver from 'file-saver';
import { tex, swap, isValidOperator, operators } from '../../mathkernel';
import ExportExpressionContext from '../shared/contexts/ExportExpressionContext';
import TextEditingDialog from '../shared/TextEditingDialog';
import TexIcon from '../tex/TexIcon';
import Expression from './Expression';

const ExpressionCard = ({
  expression,
  onEdit,
  hide,
  editable,
  removable,
  disableDrag,
  selectionMode,
  renderHeader,
}) => {
  const { openExportModal } = useContext(ExportExpressionContext);
  const [isTitleEditing, setIsTitleEditing] = useState(false);

  const actions = [
    {
      name: 'copy',
      title: 'Copy to clipboard',
      available: () => isValidOperator(expression.name),
      onClickHandler: () =>
        copyToClipboard(JSON.stringify(expression, null, 2)),
      onIconRender: () => <CopyIcon />,
    },
    {
      name: 'tex',
      title: 'Copy tex to clipboard',
      available: () => !hide && isValidOperator(expression.name),
      onClickHandler: () => copyToClipboard(tex(expression)),
      onIconRender: () => <TexIcon tex="\TeX" />,
    },
```

```
  {
    name: 'download',
    title: 'Download',
    available: () => isValidOperator(expression.name),
    onClickHandler: () =>
      fileSaver.saveAs(
        new Blob([JSON.stringify(expression, null, 2)], {
          type: 'application/json',
        }),
        expression.title || undefined
      ),
    onIconRender: () => <FileDownloadIcon />,
  },
  {
    name: 'export',
    title: 'Export as',
    available: () => !hide && isValidOperator(expression.name),
    onClickHandler: () => openExportModal(expression),
    onIconRender: () => <ExportIcon />,
  },
  {
    name: 'title',
    title: 'Add or edit title',
    available: () => isValidOperator(expression.name) && !!onEdit,
    onClickHandler: () => setIsTitleEditing(() => true),
    onIconRender: () => <TitleIcon />,
  },
  {
    name: 'delete',
    title: 'Remove',
    available: () => removable,
    onClickHandler: () => onEdit(() => []),
    onIconRender: () => <DeleteIcon />,
  },
  {
    name: 'swap',
    title: 'Swap',
    available: () => isValidOperator(expression.name) && editable,
    onClickHandler: () => onEdit((prev) => [swap({ ...prev })]),
    onIconRender: () => <ChangeCircleIcon />,
  },
  {
    name: 'double',
    title: 'Double',
    available: () => isValidOperator(expression.name) && !!onEdit,
    onClickHandler: () =>
      onEdit((prev) => [prev, JSON.parse(JSON.stringify(prev))]),
    onIconRender: () => <DoubleIcon />,
  },
];

const headerIcons = useMemo(
  () => ({
    expression: <CalculateIcon />,
    equality: <TexIcon tex={operators.equality.symbol()} />,
    inequality: <TexIcon tex={operators.inequality.symbol()} />,
    less: <TexIcon tex={operators.inequality.symbol()} />,
    greater: <TexIcon tex={operators.inequality.symbol()} />,
    leq: <TexIcon tex={operators.inequality.symbol()} />,
    geq: <TexIcon tex={operators.inequality.symbol()} />,
    empty: <TexIcon tex={operators.empty.symbol()} />,
    polynomial: <TexIcon tex={'\\sum a_ix^i'} />,
    loading: <TexIcon tex={'?'} />,
```

```
    default: <TexIcon tex={'!'} />,
  }),
  []
);

const onTitleChangeHandler = (changeCallback) => {
  onEdit((prev) => {
    const newTitle = changeCallback(prev.title);
    if (newTitle) {
      prev.title = newTitle;
    } else {
      delete prev.title;
    }
    return [{ ...prev }];
  });
};

return (
  <Card sx={{ height: 'fit-content' }}>
    <Box sx={{ display: 'flex', justifyContent: 'space-between' }}>
      <IconButton disabled>
        {headerIcons[expression.name] || headerIcons.default}
      </IconButton>
      {renderHeader?.({ expression, onEdit })}
    </Box>
    <Box
      sx={{
        opacity: selectionMode ? 1 / 3 : 1,
        pointerEvents: selectionMode ? 'none' : 'initial',
      }}
    >
      <CardContent>
        <Stack gap={2} sx={{ overflow: 'auto' }}>
          <Expression
            expression={expression}
            onEdit={onEdit}
            disableDrag={disableDrag}
            hide={hide}
          />
          {expression.title && (
            <Typography variant="h5" noWrap>
              {expression.title}
            </Typography>
          )}
          <TextEditingDialog
            open={isTitleEditing}
            onClose={() => setIsTitleEditing(() => false)}
            text={expression.title || ''}
            onChange={onTitleChangeHandler}
          />
        </Stack>
      </CardContent>
      <CardActions sx={{ overflow: 'auto' }}>
        {actions
          .filter((item) => item.available())
          .map((item, index) => (
            <Tooltip key={index} title={item.title} placement="bottom">
              <IconButton onClick={item.onClickHandler}>
                {item.onIconRender()}
              </IconButton>
            </Tooltip>
          ))}
      </CardActions>
```

```
      </Box>
    </Card>
  );
};

export default ExpressionCard;
```

ExpressionList.jsx

```
import { Fab, Stack } from '@mui/material';
import {
  Add as AddIcon,
  FileDownloadOutlined as FileDownloadOutlinedIcon,
  FileUploadOutlined as FileUploadOutlinedIcon,
} from '@mui/icons-material';
import fileSaver from 'file-saver';
import { getEmpty } from '../../mathkernel';
import ExpressionCard from './ExpressionCard';
import FileUploadFab from '../shared/FileUploadFab';

const ExpressionList = ({
  selectionMode,
  expressions,
  onEdit,
  renderCardHeader,
  renderAddition,
  additionPosition,
}) => {
  const push = (data) => onEdit(expressions.length)(() => [data].flat());

  const mapAdditionPosition = {
    left: 'row-reverse',
    right: 'row',
    top: 'column-reverse',
    bottom: 'column',
  };

  return (
    <>
      {expressions.map((expression, index) => (
        <Stack key={index} direction={mapAdditionPosition?.[additionPosition]}>
          <ExpressionCard
            expression={expression}
            onEdit={onEdit(index)}
            editable
            removable
            selectionMode={selectionMode}
            renderHeader={renderCardHeader}
          />
          {renderAddition?.({ expression, index, onEdit: onEdit(index) })}
        </Stack>
      ))}
      <Stack gap={2} direction={'row'} sx={{ justifyContent: 'center' }}>
        <Fab color="primary" onClick={() => push(getEmpty())}>
          <AddIcon />
        </Fab>
        <FileUploadFab onLoad={(data) => push(data)}>
          <FileUploadOutlinedIcon />
        </FileUploadFab>
        <Fab
          color="primary"
          onClick={() =>
```

```
            fileSaver.saveAs(
              new Blob([JSON.stringify(expressions, null, 2)], {
                type: 'application/json',
              })
            )
          }
        >
          <FileDownloadOutlinedIcon />
        </Fab>
      </Stack>
    </>
  );
};

export default ExpressionList;
```

FunctionPlot.jsx

```
import { useEffect, useRef } from 'react';
import functionPlot from 'function-plot';

const FunctionPlot = ({ onError, settings, lines }) => {
  const plotRef = useRef(null);

  useEffect(() => {
    try {
      functionPlot({
        ...settings,
        target: plotRef.current,
        data: lines,
      });
    } catch (error) {
      onError?.(error);
    }

    const old = plotRef.current;
    return () => {
      old.innerHTML = null;
    };
  }, [lines, onError, settings]);

  return <div ref={plotRef} />;
};

export default FunctionPlot;
```

ClippedDrawer.jsx

```
import { Drawer, Toolbar } from '@mui/material';

const ClippedDrawer = ({ children, anchor, drawerWidth = 1 / 4 }) => (
  <Drawer
    variant="permanent"
    anchor={anchor}
    sx={{
      width: drawerWidth,
      flexShrink: 0,
      [`& .MuiDrawer-paper`]: {
        width: drawerWidth,
        boxSizing: 'border-box',
      },
    }}
```

```
  >
    <Toolbar />
    {children}
  </Drawer>
);

export default ClippedDrawer;
```

ColorPicker.jsx

```
import { useState } from 'react';
import { SketchPicker } from 'react-color';
import { IconButton, Popover } from '@mui/material';
import {
  SquareRounded as SquareIcon,
  IndeterminateCheckBoxTwoTone as EmptySquareIcon,
} from '@mui/icons-material';

const ColorPicker = ({ color, onChange }) => {
  const [anchorColorPicker, setAnchorColorPicker] = useState(null);

  const handleColorPickerClick = (event) => {
    setAnchorColorPicker(event.currentTarget);
  };

  const handleColorPickerClose = () => {
    setAnchorColorPicker(null);
  };

  const handleColorChange = (color) => onChange(color.hex);

  return (
    <>
      <IconButton onClick={handleColorPickerClick}>
        {color ? <SquareIcon sx={{ color }} /> : <EmptySquareIcon />}
      </IconButton>
      <Popover
        open={!!anchorColorPicker}
        anchorEl={anchorColorPicker}
        onClose={handleColorPickerClose}
        anchorOrigin={{
          vertical: 'bottom',
          horizontal: 'left',
        }}
      >
        <SketchPicker
          disableAlpha
          color={color}
          onChangeComplete={handleColorChange}
        />
      </Popover>
    </>
  );
};

export default ColorPicker;
```

FileUploadFab.jsx

```
import { Fab } from '@mui/material';
import { readFileAsJSON } from '../../readFile';
```

```
const FileUploadFab = ({ onLoad, children }) => {
  const handleFileRead = (e) => {
    readFileAsJSON(e.target.files[0]).then((data) => {
      onLoad(data);
    });
  };

  return (
    <>
      <input
        type="file"
        id="file-input"
        onChange={handleFileRead}
        style={{ display: 'none' }}
      />
      <label htmlFor="file-input">
        <Fab component="span" color="primary" aria-label="Read File">
          {children}
        </Fab>
      </label>
    </>
  );
};

export default FileUploadFab;
```

LogoIcon.jsx

```
import { SvgIcon } from '@mui/material';
import { ReactComponent as AbacusIcon } from '../../static/abacus-icon.svg';

const LogoIcon = (props) => (
  <SvgIcon component={AbacusIcon} inheritViewBox {...props} />
);
export default LogoIcon;
```

NestedListItem.jsx

```
import { useCallback, useState } from 'react';
import {
  Collapse,
  List,
  ListItemButton,
  ListItemIcon,
  ListItemText,
} from '@mui/material';
import { ExpandLess, ExpandMore } from '@mui/icons-material';

const NestedListItem = ({ section }) => {
  const [open, setOpen] = useState(false);
  const onExpandHandler = useCallback(() => setOpen((prev) => !prev), []);

  return (
    <>
      <ListItemButton
        onClick={section.children ? onExpandHandler : section.onClickHandler}
      >
        {section.icon && <ListItemIcon>{section.icon()}</ListItemIcon>}
        <ListItemText primary={section.label || ''} />
        {section.children && (open ? <ExpandLess /> : <ExpandMore />)}
      </ListItemButton>
      {section.children && (
```

```
        <Collapse in={open} timeout="auto" unmountOnExit>
          <List component="div" disablePadding sx={{ pl: 2 }}>
            {section.children.map((item) => (
              <NestedListItem key={item.name} section={item} />
            ))}
          </List>
        </Collapse>
      )}
    </>
  );
};

export default NestedListItem;
```

NumberTextField.jsx

```
import { TextField } from '@mui/material';

const NumberTextField = ({ inputProps, ...props }) => (
  <TextField {...props} inputProps={{ ...inputProps, inputMode: 'numeric' }} />
);

export default NumberTextField;
```

TextEditingDialog.jsx

```
import { Dialog, TextField } from '@mui/material';

const TextEditingDialog = ({ open, onClose, text, onChange }) => (
  <Dialog open={open} onClose={onClose} maxWidth={false}>
    <form
      onSubmit={(event) => {
        event.preventDefault();
        event.stopPropagation();
        onClose();
      }}
    >
      <TextField
        value={text}
        onChange={(event) => onChange(() => event.target.value)}
        autoFocus
      />
    </form>
  </Dialog>
);

export default TextEditingDialog;
```