**Міністерство освіти і науки України**

**Львівський державний коледж харчової і переробної промисловості**

**Національного університету харчових технологій**

**Механіко-технологічне відділення**

***Циклова комісія інформатики***

|  |  |
| --- | --- |
|  | **Допущено до захисту**  Завідувач механіко-технологічного відділення  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. Є. Сагайдак  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 |

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

**до дипломного проекту на тему:**

|  |
| --- |
|  |
|  |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Студента** IV курсу групи ПК-4 | | |
|  | **спеціальності 5.05010101** | | |
|  | **Обслуговування програмних систем** | | |
|  | **і комплексів** |  |  |

(підпис) (ініціали, прізвище студента)

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Керівник** |  |  |
|  |  | (підпис) | (ініціали, прізвище) |
|  | **Консультанти:** |  |  |
|  |  | (підпис) | (ініціали, прізвище) |
|  |  |  |  |
|  |  | (підпис) | (ініціали, прізвище) |
|  | **Рецензент:** |  |  |
|  |  | (підпис) | (ініціали, прізвище) |

**РЕФЕРАТ**

Звіт про дипломний проект: 85 с, 9 рис., 4 табл., 2 додатки, 12 джерел.

Об'єктом дослідження є можливість розробки і створення програми.

Мета роботи - розробка програми «Апроксимація»

При створенні програми були вивчені та розглянуті технології .Net Framework, Windows forms.

Результатом даної роботи є розробка інформаційної структури, дизайну та інформаційне наповнення програми, який надає необхідну інформацію викладачам і студентам а також абітурієнтам.

Створена програма дозволить переглядати графіку апроксимаційних функцій на основі введених точок.

**ЗМІСТ**

Арк.

**ВСТУП 4**

1. **ТЕОРЕТИЧНА ЧАСТИНА**
   1. Поняття апроксимації
   2. Роль предметної області в сферах технологій
   3. Типи функцій апроксимації
   4. Проектування програми
   5. Засоби розробки
      1. .NET Framework
      2. Windows forms
      3. Мова програмування C#
      4. Бібліотека OxyPlot
   6. Розробка класів
   7. Діаграма класів
   8. Діаграма варіантів використання
2. **ПРОЕКТНА ЧАСТИНА**
   1. Етапи розробки проекту
      1. Вибір програмних засобів для розробки програми
      2. Постановка завдання при проектуванні
      3. Алгоритм функціонування інформаційної системи
      4. Опис та реалізація модулів системи
   2. Програмна реалізація
      1. Призначення програми
      2. Вхідні дані
      3. Результуючі дані
3. **ЕКОНОМІЧНА ЧАСТИНА**
   1. Аналіз ринку
   2. Техніко-економічне обґрунтування
   3. Стрічковий графік
   4. Кошторис витрат
4. **ОХОРОНА ПРАЦІ**
   1. Характеристика негативних факторів проектованого об'єкта
   2. Профілактичні заходи з охорони праці

**ВИСНОВКИ**

**ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

Додаток А Дизайн програми

Додаток Б Код програми

**Вступ**

Спочатку для складання програм було розглянуто математичний опис використовуваних методів. Так само додатково були вивчені основи мови програмування C #.

 Приділено особливу увагу стандартним операторам і функцій, алгоритмічної мови С #, які були використані, в написанні кожного класу..

В системі підготовки інженерів технічних спеціальностей чисельні методи є важливою складовою.

Метою роботи є розробка алгоритмів вирішення конкретних математичних задач, запрограмованих на мові С # і виконаних в Microsoft Visual Studio 2017.

Практичне виконання курсової роботи передбачає вирішення типових інженерних задач обробки даних з використанням методів матричної алгебри, рішення систем лінійних алгебраїчних рівнянь.

Мета дипломного проекту - практичне освоєння методів обчислення прикладної математики, вдосконалення навичок розробки алгоритмів і побудови програм на мові високого рівня; використання принципів модульного та об’єктно-орієнтованого програмування та вдосконалення техніки використання підпрограм закріплення знань з програмування на мові С#.

Крім зазначеного, дипломна робота призначена для засвоєння навичок з оформлення документації на програмні середовища.

**1 ТЕОРЕТИЧНА ЧАСТИНА**

**1.1 Поняття апроксимації**

Апроксимація (від латинського "approximate" - "наближатися") - наближений опис емпіричних даних за допомогою рівнянь, необхідна для проведення інтер і екстраполяції. Завдання - знайти таку функцію, виражену аналітичною формулою, щоб вона найкращим чином описувала емпіричні дані. Найбільш часто використовується метод найменших квадратів, який дозволяє знайти функцію з такими параметрами, що сума квадратів відхилень знайденою функції від заданих значень функції буде мінімальною.

Між величинами може існувати точний (функціональний) зв'язок, коли одному значенню аргументу відповідає одне певне значення, і менш точний (кореляційний) зв'язок, коли одному конкретному значенню аргумента відповідає наближене значення або деяка безліч значень функції, в тій чи іншій мірі близьких один до одного . При веденні наукових досліджень, обробці результатів спостереження або експерименту зазвичай доводиться стикатися з другим варіантом.

При виконанні будь-якої науково-дослідної роботи виникає проблема виявлення справжнього характеру залежності досліджуваних показників. Для цього і застосовується апроксимація - наближений опис кореляційної залежності змінних відповідним рівнянням функціональної залежності, що передає основну тенденцію залежності (або її "тренд").

При виборі апроксимації слід виходити з конкретного завдання дослідження. Важливо враховувати, наскільки істотні і чим зумовлені відхилення конкретних значень від одержуваного тренда. При описі залежності емпірично певних значень можна досягти і набагато більшої точності, використовуючи якесь складніше, багато параметричне рівняння.

Таким чином, вибираючи метод апроксимації, дослідник завжди йде на компроміс: вирішує, якою мірою в даному випадку доцільно і доречно «пожертвувати» деталями і, відповідно, наскільки узагальнено слід висловити залежність зіставлених змінних.

Метод найменших квадратів - один з методів теорії помилок для оцінки невідомих величин за результатами вимірів, що містить випадкові помилки.

Метод найменших квадратів застосовується також для наближення представлення заданої функції іншими (простішими) функціями і часто виявляється корисна при обробці різного роду вимірів.

Метод найменших квадратів був запропонований К.Ф. Гауссом і А. Лежандром. Нині спосіб являє собою один з найважливіших розділів математичної статистики і широко використовується для статистичних висновків в різних областях науки і техніки.

У більшості експериментальних даних, що задаються за допомогою табличній функції, є досить великий розкид точок. При цьому використання безперервної інтерполяції не завжди виправдано, оскільки ставиться завдання дослідити загальну тенденцію зміни фізичної величини.

В цьому випадку апроксимації шукана крива не обов'язково повинна проходити через задані точки. Передбачається використовувати криву, сума квадратів відхилень у вузлових точках мінімальна. Саме в таких випадку використовується метод найменших квадратів.

Фахівці в області автоматизації технологічних процесів і виробництв мають справу з великим об'ємом експериментальних даних, для обробки яких використовується комп'ютер.

При розгляді різних завдань в цій області виникає, зокрема, необхідність виявлення деяких емпіричних закономірностей, рішення систем рівнянь, первинної статистичної обробки експериментальних даних.

**1.2. Роль предметної області в сферах технологій**

Апроксимація функцій застосовується в багатьох сферах науки та технологій.

Деякі проблеми фізики бувають занадто складні для вирішення шляхом прямого аналізу, або прогрес може бути обмежений наявними аналітичними інструментами. Таким чином, навіть коли точне представлення відомо, апроксимація може дати достатньо точне рішення, при цьому значно знизити складність проблеми. Фізики часто наближують форму Землі до сфери, хоча можливі більш точні терміни, оскільки багато фізичних характеристик (наприклад, гравітації) набагато простіше розрахувати для сфери, ніж для інших форм.

Апроксимація також використовується для аналізу руху декількох планет, що обертаються навколо зірки. Це надзвичайно важко через складні взаємодії гравітаційних впливів планет один на одного. Наближене рішення здійснюється шляхом виконання ітерацій.

Використання ускладнень для виправлення помилок може дати більш точні рішення. Моделювання рухів планет і зірок також дає більш точні розв'язки.

Найбільш поширені версії філософії науки визнають, що емпіричні вимірювання завжди є наближеннями - вони не ідеально відображають те, що вимірюється.

Властиві допуску помилок декількох додатків (наприклад, графічних додатків) дозволяють використовувати апроксимацію (наприклад, зниження точності числових обчислень) для підвищення продуктивності та енергоефективності. [ Цей підхід використання навмисної, контрольованої апроксимації для досягнення різних оптимізацій називається наближеним обчисленням.

**1.3. Типи апроксимації**

Якщо задана функція y (x), то це означає, що будь-якому допустимому значенню x відповідає значення y. Але нерідко виявляється, що знаходження цього значення дуже трудомістке. Наприклад, y (x) може бути визначено як рішення складного завдання, в якій x грає роль параметра, або y (x) вимірюється в дорогому експерименті. При цьому можна вичислити невелику таблицю значень функції, але пряме знаходження функції при великому числі значень аргументу буде практично неможливо. Функція y (x) може брати участь в будь-яких фізико-технічних або чисто математичних розрахунках, де її доводиться багаторазово обчислювати. У цьому вигідно замінити функцію y (x) наближеною формулою, тобто підібрати деяку функцію φ (x). Яка близька в деякому розумінні до y (x) і просто обчислюється. Потім при всіх значення аргументу вважають y (x) ≈ φ (x). Близькість отримують введенням в апроксимуючу функію вільних параметрів a = {a1, a2, ... an} і відповідним їх вибором.

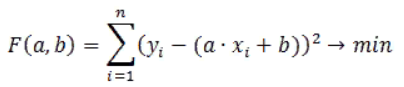
**Лінійна апроксимація**

Будь-яка лінійна функція може бути записана рівнянням:

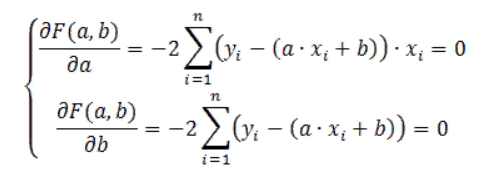


Апроксимація полягає в знаходженні коефіцієнтів a і b рівняння таких, щоб все експериментальні точки лежали найближче до апроксимуючої прямої.

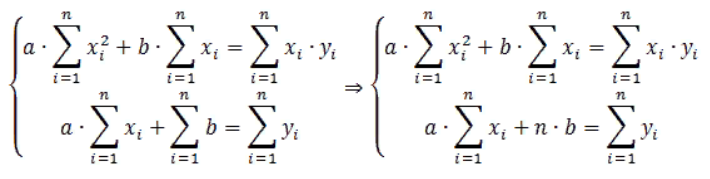
З цією метою найчастіше використовується метод найменших квадратів (МНК), суть якого полягає в наступному: сума квадратів відхилень значення точки від апроксимуючої точки приймає мінімальне значення:



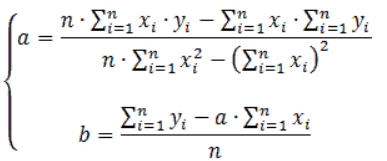
Рішення поставленого завдання зводиться до знаходження екстремуму зазначеної функції двох змінних. З цією метою знаходимо частинні похідні функції функції за коефіцієнтами a і b і прирівнюємо їх до нуля.



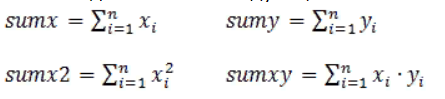
Вирішуємо отриману систему рівнянь



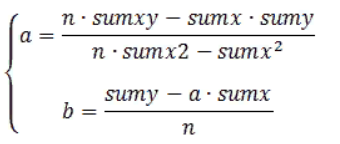
Визначаємо значення коефіцієнтів



Для обчислення коефіцієнтів необхідно знайти такі складові:

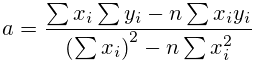


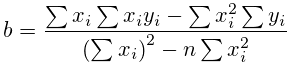
Тоді значення коефіцієнтів будуть визначені як



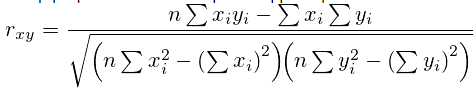
Для зручності формули були спрощені для кращого розуміння. Програма обчислює через наступні формули:

Коефіцієнти a і b:





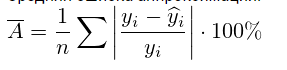
Коефіцієнт кореляції:



Коефіцієнт детермінації:



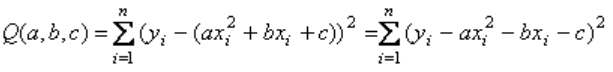
Середня помилка апроксимації:



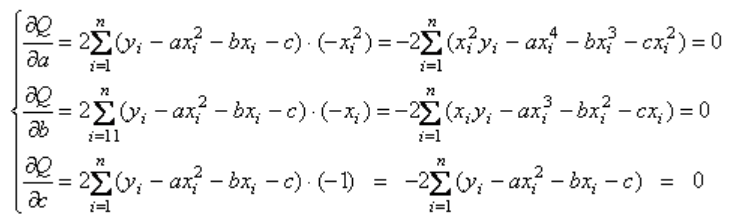
**Квадратична апроксимація**

Якщо апроксимуючою функцією є квадратична залежність, то її функція це ,

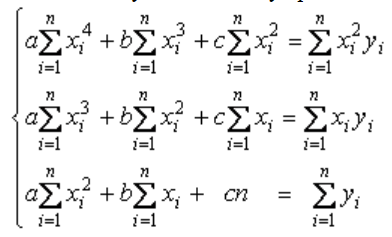
А параметри a, b, c знаходять з умови мінімуму функції:



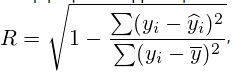
Умови мінімуму функції зводяться до системи рівнянь:



Після перетворень отримуємо систему трьох лінійних рівнянь з трьома невідомими:

,

Коефіцієнт кореляції:

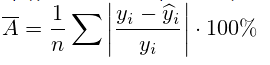


Де 

Коефіцієнт детермінації:

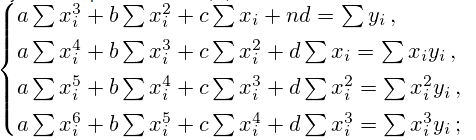


Середня помилка апроксимації:



**Кубічна апроксимація**

Кубічна апроксимація схожа за формулою до квадратної



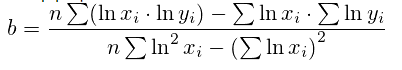
Коефіцієнт кореляції, коефіцієнт детермінації, середня помилка апроксимації - використовуються ті ж формули, що і для квадратичної апроксимації.

**Степенева апроксимація**

Функція:



Коефіцієнт b:



Коефіцієнт a:



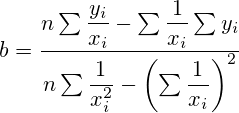
Коефіцієнт кореляції, коефіцієнт детермінації, середня помилка апроксимації - використовуються ті ж формули, що і для квадратичної регресії.

**Гіперболічна регресія**

Рівняння регресії:

\widehat{y}=a + \frac{b}{x}

Коефіцієнт b:



Коефіцієнт a:

a=\dfrac{1}{n}\sum y_i-\dfrac{b}{n}\sum\dfrac{1}{x_i}

Коефіцієнт кореляції, коефіцієнт детермінації, середня помилка апроксимації - використовуються ті ж формули, що і для квадратичної регресії.

**Логарифмічна регресія**

Рівняння регресії:

\widehat{y}=a + b\ln x

Коефіцієнт b:

b=\dfrac{n\sum(y_i\ln x_i)-\sum\ln x_i\cdot \sum y_i }{n\sum\ln^2x_i-\left(\sum\ln x_i\right)^2 }

Коефіцієнт a:

a=\dfrac{1}{n}\sum y_i-\dfrac{b}{n}\sum\ln x_i

Коефіцієнт кореляції, коефіцієнт детермінації, середня помилка апроксимації - використовуються ті ж формули, що і для квадратичної регресії.

**Експоненціальна регресія**

Рівняння регресії:

\widehat{y}=e^{a+bx}

Коефіцієнт b:

b=\dfrac{n\sum x_i\ln y_i-\sum x_i\cdot\sum\ln y_i }{n\sum x_i^2-\left(\sum x_i\right)^2 }

Коефіцієнт a:

a=\dfrac{1}{n}\sum\ln y_i-\dfrac{b}{n}\sum x_i

Коефіцієнт кореляції, коефіцієнт детермінації, середня помилка апроксимації - використовуються ті ж формули, що і для квадратичної регресії.

**1.4. Проектування програми**

**Проектування ПЗ**- це процес визначення архітектури, набору компонентів, їх інтерфейсів, інших характеристик системи і кінцевого складу програмного продукту.

## Проектування програмного забезпечення - це процес вирішення задач та планування для створення програмного рішення. Після того як мета і специфікація програми описані, розробник створює дизайн проекту. В дизайн включаються як описи низькорівневих компонентів, алгоритмів, так і огляд архітектури. Проектування передбачає вироблення властивостей системи на основі аналізу постановки задачі, а саме: моделей предметної області, вимог до ПЗ, а також досвіду проектувальника. Визначенню внутрішніх властивостей системи та деталізації її зовнішніх властивостей власне і присвячене проектування.

## Залежно від класу створюваного ПЗ, процес проектування може забезпечуватися як «ручним» проектуванням, так і різними засобами його автоматизації. У процесі проектування ПЗ для вираження його характеристик використовуються різні нотації - блок-схеми, ER-діаграми, UML-діаграми, DFD-діаграми, а також макети.

## Наведемо опис системи, що моделює предметну область, в термінах об'єктно-орієнтованого підходу:

## Об'єкт (object) - це обмежена сутність, що характеризується своїм станом і поведінкою. Об'єкт є екземпляром класу, який описує усю множину суб'єктів даного типу, тобто об'єктів з таким же станом і поведінкою.

## Тип class (клас) описує множину однотипних об'єктів з одними й тими ж даними і функціями, тобто мають загальний стан і поведінку. За допомогою class програміст описує спільно дані і дії над цими даними.

## Конструктор (constructor) - це функція класу, основне призначення якої є в ініціалізації значень даних новостворюваного об'єкту. Ім'я конструктора завжди співпадає з іменем класу. Конструктор викликається кожного разу при створенні об'єкту. Клас може мати декілька перевантажених конструкторів. При створенні об'єкту застосовується той конструктор, параметри якого забезпечують встановлення потрібних навчальних значень даних цього об'єкту. Конструктор не має значення, яке повертається.

## Інкапсуляція (encapsulation) - це процес розмежування елементів об'єкту, що визначають його влаштування та поведінку. Інкапсуляція забезпечується через закриття інформації про внутрішню структуру об'єкту, маскуванням внутрішніх деталей які явно не впливають на зовнішню поведінку.

## Спадкування (inheritance) класів - це засоби отримання нових класів на базі існуючих. При цьому новий клас спадкує дані і функції з одного базового класу.

## Поліморфізм (polymorphism) - це підхід, який реалізує ідею: один інтерфейс - багато методів. Тобто, різні об'єкти можуть мати методи з однаковим інтерфейсом, проте різним функціонуванням.

**1.5. Засоби розробки**

Microsoft Visual Studio - серія продуктів фірми Microsoft, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

Visual Studio включає один або декілька з наступних компонентів:

Visual Basic .NET, а до його появи — Visual Basic

Visual C++

Visual C#

Visual F# (входить до складу Visual Studio 2010);

Visual Studio Debugger

Багато варіантів постачання також включають:

Microsoft SQL Server або

MSDE Visual Source Safe — файл-серверна система управління версіями

У минулому, до складу Visual Studio також входили продукти:

Visual InterDev

Visual J++

Visual J#

Visual FoxPro

Visual Source Safe – файл-серверна система управління версіями.

**1.5.1 .NET Framework**

Microsoft .NET (читається дот-нет) програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Одною з ідей .NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість.

Кожна бібліотека (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET — крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD (від Microsoft) і варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell), DotGNU [1].

Захист авторських прав відноситься до створення середовищ виконання (CLR — Common Language Runtime) для програм .NET. Компілятори для .NET випускаються багатьма фірмами для різних мов вільно.

.NET поділяється на дві основні частини — середовище виконання (по суті віртуальна машина) та інструментарій розробки.

Середовища розробки .NET-програм: Visual Studio .NET (C++, C#, J#), SharpDevelop, Borland Developer Studio (Delphi, C#) тощо. Середовище Eclipse має додаток для розробки .NET-програм. Застосовні програми також можна розроблювати в текстовому редакторі та використовувати консольний компілятор.

Як і технологія Java, середовище розробки .NET створює байт-код, призначений для виконання віртуальною машиною. Вхідна мова цієї машини в .NET називається CIL (Common Intermediate Language), також відома як MSIL (Microsoft Intermediate Language), або просто IL. Застосування байт-коду дозволяє отримати крос-платформність на рівні скомпільованого проекту (в термінах .NET: збірка), а не на рівні сирцевого тексту, як, наприклад, в С. Перед запуском збірки в середовищі виконання (CLR) байт-код перетворюється вбудованим в середовище JIT-компілятором (just in time, компіляція на льоту) в машинні коди цільового процесора.

Слід зазначити, що один з перших JIT-компіляторів для Java був також розроблений фірмою Microsoft (тепер в Java використовується досконаліша багаторівнева компіляція — Sun HotSpot). Сучасна технологія динамічної компіляції дозволяє досягнути аналогічного рівня швидкодії з традиційними «статичними» компіляторами (наприклад, С++) і питання швидкодії часто залежить від якості того чи іншого компілятора.

**1.5.2 Windows Forms**

Інтерфейс програмування додатків (API), відповідальний за графічний інтерфейс користувача і є частиною Microsoft .NET Framework. Даний інтерфейс спрощує доступ до елементів інтерфейсу Microsoft Windows за допомогою створення обгортки для Win32 API в керованому коді.

Як і Abstract Window Toolkit (AWT) (схожий API для мови Java), бібліотека Windows Forms була розроблена як частина .NET Framework для спрощення розробки компонентів графічного інтерфейсу користувача. Windows Forms побудована на основі застарілого Windows API.

Windows Forms надає можливість розробки кросплатформного графічного користувацького інтерфейсу. Але Windows Forms є лише обгорткою Windows API-компонентів.

**1.5.3 Мова програмування C#**

Мова програмування C#— об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до С++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов С++, Delphi, Модула і Smalltalk — С#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).

**1.5.4. Бібліотека OxyPlot**

OxyPlot - це бібліотека генерації графіку з відкритим кодом, яка ліцензована під ліцензією MIT. Ліцензія MIT доступна і дозволяє використовувати в фірмовому програмному забезпеченні.

Бібліотека базується на .NET і націлена на декілька платформ. Основна бібліотека - це портативна бібліотека, що дозволяє легко використовувати повторно використовуваний код на різних платформах.

Основна бібліотека - це бібліотека портативних класів, яка може використовуватися на різних платформах. Спеціальні елементи керування реалізовані для WPF, Windows 8, Windows Phone, Windows Phone Silverlight, Windows Forms, Silverlight, GTK #, Xwt, Xamarin.iOS, Xamarin.Android, Xamarin.Forms і Xamarin.Mac.

Мета полягала в тому, щоб надати модель побудови графіку, яка проста у використанні і відкрита для розширення. Є можливість змінювати зовнішній вигляд ділянок, змінюючи прості властивості. Додаткові налаштування можуть бути досягнуті шляхом отримання нових підкласів, які мають повний контроль як за поведінкою, так і за візуалізацією.

OxyPlot містить багато різних типів осей і рядів. Якщо потрібні деякі функціональні можливості, які не входять до бібліотеки, ви також можете створити похідний клас і перевизначити реалізацію візуалізації.

Ділянки можна експортувати у файлові формати, такі як png, pdf та svg.

OxyPlot надається як пакет NuGet. Щоб використовувати бібліотеку, необхідно додати посилання на пакет NuGet, який підтримує платформу вашої програми.

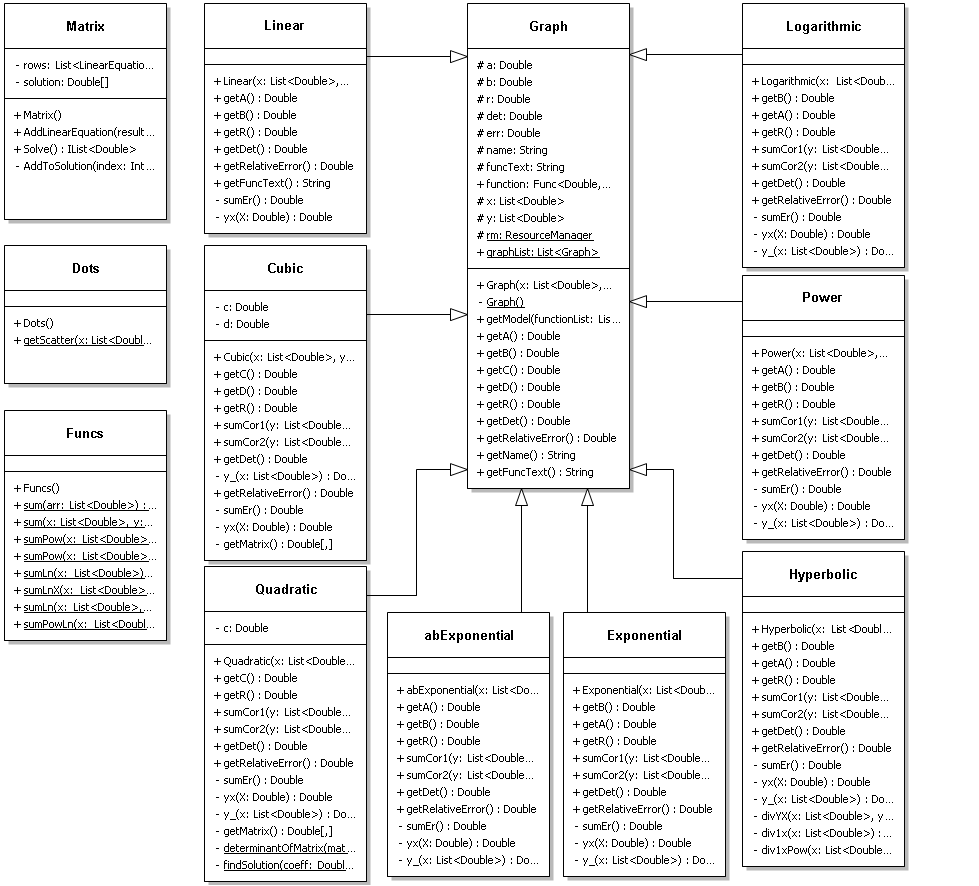
**1.6. Розробка класів**

Перед тим як почати програмувати проект, який буде мати кілька класів, слід створити список цих класів та їх короткий опис

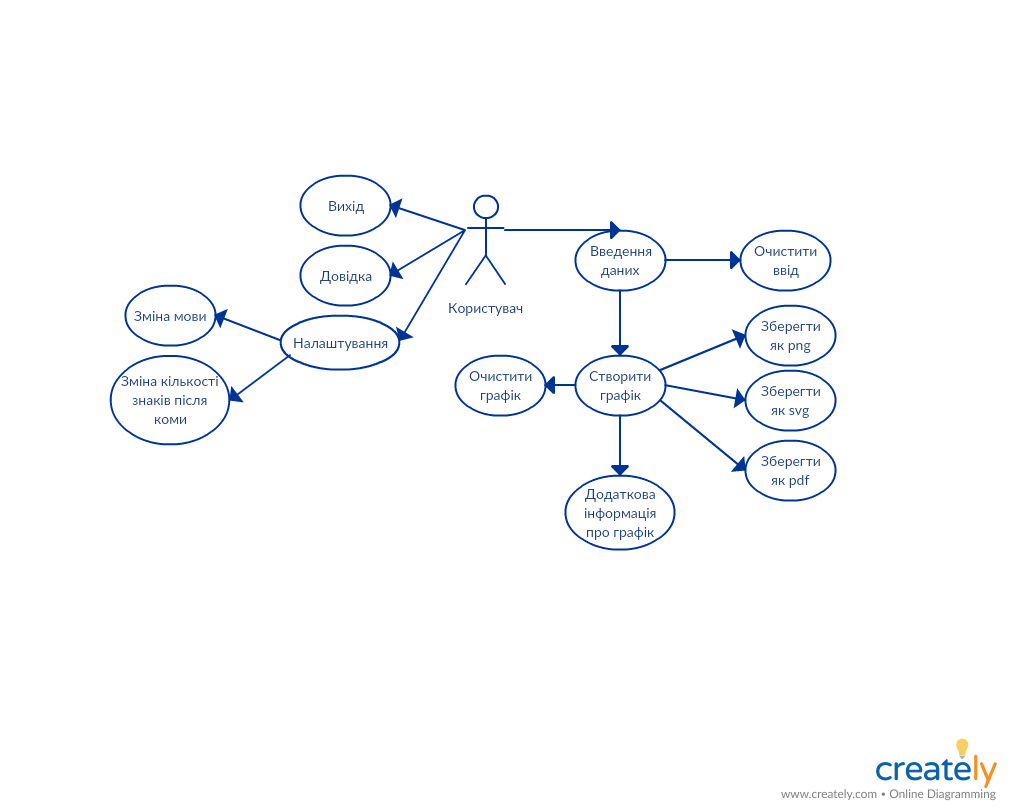
**Таблиця 1.6.** Відомості про класи, які було використано в програмі

|  |  |  |
| --- | --- | --- |
| Клас | Базовий клас | Опис |
| MainForm | Form | Головна форма |
| SettingsForm | Form | Форма налаштувань |
| InformForm | Form | Форма додаткової інформації |
| Dots | Немає | Клас для побудови точок |
| Funcs | Немає | Клас, що має основні функції для обчислювання даних |
| Matrix | Немає | Клас для обчислювання матриць |
| Graph | Немає | Основний клас для будування графіку |
| abExponentional | Graph | Клас для обчислення апроксимації показниковим методом |
| Cubic | Graph | Клас для обчислення апроксимації кубічним методом |
| Exponentional | Graph | Клас для обчислення апроксимації експоненційним методом |
| Hyperbolic | Graph | Клас для обчислення апроксимації гіперболічним методом |
| Linear | Graph | Клас для обчислення апроксимації лінійним методом |
| Logarithmic | Graph | Клас для обчислення апроксимації логарифмічним методом |
| Power | Graph | Клас для обчислення апроксимації степеневим методом |
| Quadratic | Graph | Клас для обчислення апроксимації квадратичним методом |

**1.7. Діаграма класів**



**1.8. Діаграма варіантів використання**



1. **ПРОЕКТНА ЧАСТИНА**

**2.1. Eтапи розробки проекту**

Водоспадна (каскадна, послідовна) модель.

Водоспадна модель життєвого циклу (англ. waterfall model) була запропонована в 1970 р. Вінстоном Ройсом. Вона передбачає послідовне виконання всіх етапів проекту в строго фіксованому порядку. Перехід на наступний етап означає повне завершення робіт на попередньому етапі. Вимоги, визначені на стадії формування вимог, суворо документуються у вигляді технічного завдання і фіксуються на весь час розробки проекту. Кожна стадія завершується випуском повного комплекту документації, достатньої для того, щоб розробка могла бути продовжена іншою командою розробників.

Етапи проекту у відповідності з каскадною моделлю:

* Формування вимог.
* Проектування.
* Реалізація.
* Тестування.
* Впровадження.
* Експлуатація та супровід.

Переваги

* Повна і погоджена документація на кожному етапі.
* Легко визначити терміни і витрати на проект.

Недоліки

У водоспадної моделі перехід від однієї фази проекту до іншого передбачає повну коректність результату (виходу) попередньої фази. Однак неточність будь-якої вимоги або некоректна його інтерпретація в результаті призводить до того, що доводиться «відкочуватися» до ранньої фазі проекту і необхідна переробка не просто вибиває проектну команду з графіка, але часто призводить до якісного зростання витрат і, не виключено, до припинення проекту в тій формі, в якій він спочатку замислювався. На думку сучасних фахівців, основна помилка авторів водоспадної моделі полягає у припущеннях, що проект проходить через весь процес один раз, спроектована архітектура хороша і проста у використанні, проект здійснення розумний, а помилки в реалізації легко усуваються в міру тестування. Ця модель виходить з того, що всі помилки будуть зосереджені на реалізації, а тому їх усунення відбувається рівномірно під час тестування компонентів і системи[1]. Таким чином, водоспадна модель для великих проектів мало реалістична і може бути ефективно використана тільки для створення невеликих систем.

**2.1.1 Вибір програмних засобів для розробки програми**

Microsoft Visual Studio-серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

**2.1.2. Постановка завдання при проектуванні**

Одним з найбільш важливих кроків програмування є постановка задачі. Вона виконує функції контракту між користувачем і програмістом (програмістами).Які юридично погано складений контракт, погана постановка задачі марна. При гарній постановці завдання як користувач, так і програміст ясно і недвозначно представляють завдання, яку необхідно виконати, тобто в цьому випадку враховуються інтереси як користувача, так і програміста.

Користувач може планувати використання ще нествореного програмного забезпечення, спираючись на знання того, що воно може. Гарна постановка завдання служить основою для формування її рішення.

Постановка завдання (специфікація програми);по суті, означає точне, повне і зрозуміле опис того, що відбувається при виконанні конкретної програми. Користувач зазвичай дивиться на комп'ютер, як на чорний ящик: для нього неважливо, як працює комп'ютер, а важливо, що може комп'ютер з того, що цікавить користувача. При цьому основна увага фокусується на взаємодії людини з машиною.

При створенні й розвитку ПЗ рекомендується застосовувати такі загальносистемні принципи:

* принцип включення, який передбачає, що вимоги до створення, функціонування та розвитку ПЗ визначаються з боку більш складної системи, що включає його в себе;
* принцип системної єдності, який полягає в тому, що на всіх стадіях створення, функціонування та розвитку ПЗ його цілісність буде забезпечуватися зв'язками між підсистемами, а також функціонуванням підсистеми управління;
* принцип розвитку, який передбачає в ПЗ можливість його нарощування та вдосконалення компонентів і зв'язків між ними;
* принцип комплексності, який полягає в тому, що ПЗ забезпечує зв'язаність обробки інформації, як окремих елементів, так і для всього обсягу даних в цілому на всіх стадіях обробки;
* принцип інформаційної єдності, тобто у всіх підсистемах, засобах забезпечення і компонентах ПЗ використовуються єдині терміни, символи, умовні позначення і способи подання;
* принцип сумісності полягає в тому, що мова, символи, коди та засоби програмного забезпечення узгоджені, забезпечують спільне функціонування всіх підсистем і зберігають відкритою структуру системи в цілому;
* принцип інваріантності визначає інваріантність підсистем і компонентів ПЗ до оброблюваної інформації, тобто вони є універсальними або типовими.

При розробці програми увага була направлена на такі функціональні вимоги:

* Будування точок на основі введених даних.
* Будування графіку апроксимації на основі обраних видів апроксимації.
* Відображення інформації, що використовується при обчисленні, такої як коефіцієнт детермінації, коефіцієнт кореляції, середня помилка апроксимації.
* Можливість зміни мови з української на англійську та навпаки.
* Можливість зміни кількості знаків після коми в результатах розрахунків.

Нефункціональні вимоги:

* Зручний користувацький інтерфейс
* Програма повинна запускатись на ОС Windows 7, Windows 8, Windows 10.
* Користувачі не повинні мати можливості вводити не цифрові дані

**2.1.3. Алгоритм функціонування інформаційної системи**

Структуру інформаційної системи складає сукупність окремих її частин - підсистем. Підсистема - це частина системи, яка виділена за певною ознакою. Тому структура будь-якої інформаційної системи може бути представлена як сукупність підсистем, що забезпечують інформаційне, технічне, математичне, програмне, організаційне і правове забезпечення

Програма працює як сукупність класів, які відповідають за свою задачу.

Після вводу даних, дані проходять перевірку на коректність введених даних, саме тому в програмі неможливо ввести будь-які інші символи крім цифр.

Після того як користувач натискає кнопку «Розрахунок», програма перевіряє чи данні були введені та робить перевірку чи всі клітинки таблиці були заповнені, якщо ні – то клітинки автоматично заповнюються нулями. Тільки тоді коли введені дані пройдуть всі перевірки на коректність, вони заносяться в два списка X і Y. Дані проходять обчислення і клас Dots будує точки на графіку. Якщо при натисканні кнопки «Розрахунок» були відмічені деякі типи апроксимації, то розраховуються саме ці вибрані типи, за допомогою відповідно названих класів. Коли всі дані будуть обчислені, графік з відповідними типами апроксимації буде побудовано.

Користувачу на вибір були надані такі типи апроксимації:

Лінійна апроксимація

Квадратична апроксимація

Кубічна апроксимація

Степенева апроксимація

Показова апроксимація

Логарифмічна апроксимація

Гіперболічна апроксимація

Експоненціальна апроксимація

Користувач може обрати як один з цих типів апроксимації, так і декілька видів.

**2.1.4. Опис та реалізація модулів системи**

Структура програми полягає в основному класі Graph, який описує поведінку інших типів апроксимації. В ньому є поля та методи, які використовуються у всіх наданих типах апроксимації. Ці типи успадковують його поля та методи, це робиться для того щоб код не повторювався і був дотриманий один з основних концепцій ООП – успадкування.

В таблицях буде описаний тільки клас Graph, тому що інші класи типів апроксимації мають точно такі самі поля та методи. Винятком є класи Quadratic та Cubic, що містять у собі змінні c і d, які потрібні для розрахунків.

**Таблиця 2.1.4.** Відомості про поля, які знаходяться в класі Graph

|  |  |  |  |
| --- | --- | --- | --- |
| **Поле** | **Тип** | **Доступ** | **Опис** |
| a | double | protected | Змінна, що містить значення a |
| b | double | protected | Змінна, що містить значення a |
| r | double | protected | Змінна, що містить коефіцієнт кореляції |
| err | double | protected | Змінна, що містить середню помилку апроксимації |
| det | double | protected | Змінна, що містить коефіцієнт детермінації |
| funcText | string | protected | Змінна, що містить текст функції |
| name | string | protected | Змінна, що містить назву типу апроксимації |
| function | Func<double, double> | public | Змінна, що містить функцію потрібну для будування графіка |
| x | List<double> | protected | Змінна, що містить набір значень x |
| y | List<double> | protected | Змінна, що містить набір значень y |
| graphList | List<Graph> | static public | Змінна, що містить список обраних типів апроксимації |
| rm | ResourceManager | protected | Змінна, що містить інформацію про поточну обрану мову |

**Таблиця 2.1.4.** Відомості про методи, які знаходяться в класі Graph

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Метод** | **Тип, який повертає** | **Аргумент** | **Доступ** | **Опис** |
| getA | double |  | public | Метод повертає значення змінної a |
| getB | double |  | public | Метод повертає значення змінної b |
| getC | double |  | public | Метод повертає значення змінної c |
| getD | double |  | public | Метод повертає значення змінної d |
| getR | double |  | public | Метод повертає значення коефіцієнту кореляції |
| getDet | double |  | public | Метод повертає значення коефіцієнту детермінації |
| getRelativeError | double |  | public | Метод повертає значення середньої помилки апроксимації |
| getName | string |  | public | Метод повертає назву типу апроксимації |
| getFuncText | string |  | public | Метод повертає текст функції апроксимації |
| getModel | PlotModel | List<FunctionSeries> functionList | public | Метод отримує список обраних типів апроксимації та на їх основі будує графік |

**Таблиця 2.1.4.** Відомості про методи, які знаходяться в класі Funcs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Метод** | **Тип, який повертає** | **Аргумент** | **Доступ** | **Опис** |
| sum | double | List<double> arr | static public | Метод повертає суму ряду |
| sum | double | List<double> x, List<double> y | static public | Метод повертає добуток двох рядів |
| sumPow | double | List<double> x, int power | static public | Метод повертає суму ряда в заданому степені |
| sumPow | double | List<double> x, List<double> y, int power | static public | Метод повертає суму рядів y і x в заданому степені |
| sumLn | double | List<double> x | static public | Метод повертає суму логарифма натурального від ряда |
| sumLn | double | List<double> x, List<double> y | static public | Метод повертає суму двох рядів від логарифма натурального |
| sumLnX | double | List<double> x, List<double> y | static public | Метод повертає суму x від логарифма натурального та y |
| sumPowLn | double | List<double> x, int power | static public | Метод повертає cумe логарифма натурального від масиву в заданому степені |

**2.2 Програмна реалізація**

**2.2.1 Призначення програми**

Програма розроблена для використання в будь яких експерементах та дослідах що включає в себе спостерігання за зміною об’єкта, аналіз позиції предмета і тд. Функції апроксимації використовують в багатьох сферах науки та технології. Будь який звичайний користувач може використовувати програму як в математичних цілях, так і для простого аналізу зміни властивостей об’єктів під різним кутом.

**2.2.2** **Вхідні дані**

Головна форма має таблицю, в яку можна вписувати розрахункові дані. Після цього можна побудувати графік, який відобразить точки та графік в залежності від обраного типу апроксимації. Користувач може обирати між декількома типами апроксимації (лінійна, квадратична, кубічна, степенева, показова, логарифмічна, гіперболічна, експоненціальна). Ввід в таблицю захищений від введення з клавіатури символів, що не є цифрами і захищенний від вставки небажаних символів.

**2.2.3 Результуючі дані**

Головна форма виводить графік на основі введених даних та обраного виду апроксимації. Дані на графіку підписані та при клацанні мишкою на графік можна дивитись координати точок та кривих.

Форма з додаткової інформацією містить динамічні компоненти, які автоматично створюються в залежності від вибору типів апроксимації. На цій формі відображають результати розрахунків. Програма розраховує a, b, коефіцієнт кореляції, коефіцієнт детермінації та середню помилку апроксимації. Користувач має можливість змінити кількість знаків після коми в результатах розрахунків через відповідний пункт в меню. Програма має дві мови інтерфейсу: англійська та українська. Користувач може змінити мову в налаштуваннях, але для цього потрібно буде перезапустити програму.

**3. ЕКОНОМІЧНА ЧАСТИНА**

**3.1 Аналіз ринку**

Ринок програмного забезпечення в даний час є найбільш бурхливо розвиваються. Одночасно з тим, він має ряд унікальних особливостей. По-перше, він сильно неоднорідний. Немає єдиного ринку програм, в ньому можна виділити кілька основних ринків:

• серверних операційних систем

• ПО для серверів і мереж

• операційних систем для настільних комп'ютерів

• систем забезпечення безпеки

• прикладного ПО

• баз даних

• коштів розробки

Всі ці ринки не є ізольованими, вони тісно взаємопов'язані на основі сумісності та критеріїв для користувача вибору.

По-друге, на економічні аспекти поширення програм дуже сильно впливають правові обмеження особливого роду - програмні ліцензії та пов'язані з ними побічні явища - програмне піратство. Конкурентна боротьба в даній сфері зазвичай здійснюється, як і чисто економічними методами, (цінова політика, монополізація ринку, поглинання партнерів конкурентів і т. Д.), Так і юридичними (судові позови, ліцензійна політика і т. П.).

Основною особливістю ринку ПО є проблема забезпечення сумісності програмного і апаратного забезпечення користувача. Практично це проявляється в наступному - вибір програмного набору починається ще при виборі самого комп'ютера, його типу і архітектури.

Програма «Апроксімація» була розроблена для виконання різного роду дослідів та експерементів, які включають в собі аналіз та спостерігання за змінами властивостей об'єкта. Програму можна використовувати в різних сферах науки. Вона відкрита для додавання модулів та подальшого вдосконалення.

**3.2 Техніко-економічне обґрунтування**

Щоб оцінити витрати на проведення розробки програми «Апроксимація» потрібно:

1) розрахувати час, необхідний для виконання усіх робіт;

2) кошторис витрат;

Для розробки програми був задіяний один програміст

**3.3 Стрічковий графік виконання робіт**

Стрічковий графік використовують для наглядного представлення співвідношення складових якогось параметра і одночасно для вираження зміни цих складових з часом. При побудові стрічкового графіка прямокутник графіка ділять на зони пропорційно складовим або відповідно до кількісних значень. По довжині стрічки розмічають ділянки відповідно до співвідношення складових по кожному чиннику. Систематизуючи стрічковий графік так, щоб стрічки розташовувалися у послідовному тимчасовому порядку, можна оцінити зміну складових з часом. Час для розробки електронного журналу групи наведений в таблиці 3.1.

Таблиця 3.1. Стрічковий графік

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1-й місяць | | | | 2-й місяць | | | | 3-ій місяць | | | |
| тижні | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Вибір та узгодження теми роботи |  |  |  |  |  |  |  |  |  |  |  |  |
| Збір інформації |  |  |  |  |  |  |  |  |  |  |  |  |
| Вибір засобів реалізації |  |  |  |  |  |  |  |  |  |  |  |  |
| Розробка програми |  |  |  |  |  |  |  |  |  |  |  |  |
| Тестування та налагодження |  |  |  |  |  |  |  |  |  |  |  |  |
| Оформлення результатів роботи |  |  |  |  |  |  |  |  |  |  |  |  |

**3.4 Кошторис витрат**

1) Розрахунок витрат на видаткові матеріали наведений в таблиці 3.2.

Таблиця 3.2. Витрати на видаткові матеріали

|  |  |
| --- | --- |
| Матеріал | Сума, грн. |
| Visual studio 2017 | 250 |
| Разом | 250 |

2) Заробітна плата – матеріальна винагорода, яка отримується працівником відповідно до витрат і результатів праці і витрачається ним на задоволення особистих потреб.

Таблиця 3.3. Розрахунок річного фонду заробітної плати працівників

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Назва професії | Кількість відпрацьованих людиною днів | Денна тарифна ставка | Тарифний фонд зарплати | Доплати | Основна зарплата | Додаткова зарплата | Загальний фонд зарплати |
| Програміст | 40 | 150 | 6000 | 3600 | 9600 | 960 | 10560 |
| Разом |  | - | - | - |  |  |  |

Визначаємо доплати до тарифного фонду:

Д = Тф.з. × 0,6

Д = 6000,00 × 0,6 = 3600,00 грн

де, Тф.з  - тарифний фонд зарплати, грн

Основна зарплата складає:

ЗПосн = Тф.з. + Д

ЗПосн = 6000,00 + 3600,00 = 9600,00 грн

де, ЗПосн – основна зарплата, грн

Д – доплати, грн

Додаткова зарплата становить:

ЗПдод = ЗПосн × 0,10

ЗПдод = 9600,00 × 0,10 = 960,00 грн

Де, ЗПдод – додаткова зарлата, грн

Визначаємо загальний фонд зарплати:

Z = ЗПосн + ЗПдод

Z = 9600,00 + 960,00 = 10560,00 грн

3) Від загального фонду заробітної плати здійснюємо відрахування на соціальні заходи (N) – 22% ЄСВ

N = 10560,00 × 0,22 = 2323,2 грн

4) Накладні витрати (Nv) – це додаткові витрати до основних витрат для забезпечення процесів виробництва (витрати підприємств, господарських організацій, будівництв, пов’язані з їх управлінням і обслуговуванням, вмістом і експлуатацією устаткування). До накладних витрат відносяться: витрати експлуатацію і поточний ремонт будівель, споруд і устаткування; відрахування на соціальне страхування і інші обов’язкові платежі; заробітну плату адміністративно-управлінського персоналу; витрати, пов’язані з втратами від браку і простоїв і ін.

Від загального фонду заробітної плати здійснюємо розрахунок накладних витрат – 20%.

Nv = 10560,00 × 0,2 = 2112,00 грн

5) Послуги сторонніх організацій (Р), наприклад, вартість доступу до інтернету провайдера “Київстар” – 120грн/місяць. Загальна вартість 360 грн.

6) Собівартість (S) — всі витрати, затрачені підприємством на виробництво і реалізацію продукції або послуги.

*S = Z + N+ Nv + P = S = 10560,00 + 2323,20 + 2112,00 + 360,00 = 15355,2 грн*

7) Податок на додану вартість (ПДВ) – непрямий податок, форма вилучення до бюджету частини доданої вартості, яка створюється на всіх стадіях процесу виробництва товарів, робіт і послуг і вноситься до бюджету по мірі реалізації.

ПДВ = S × 20% = 3071,04 грн.

Таблиця 3.4 Зведений кошторис витрат на програму «Апроксимація».

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Матеріали | Заробітна плата | Відрахування на ЄСВ | Накладні витрати | Послуги сторонніх організацій | Собівартість | ПДВ | Всього |
| 250 | 10560 | 2323,2 | 2112 | 360 | 15355,2 | 3071,04 | 18676,24 |

Випускна вартість програми «Апроксимація» складає – 18676,24грн.

**4. ОХОРОНА ПРАЦІ**

**4.1 Характеристика негативних факторів проектованого об'єкта**

Робота з охорони праці виконувалася на підприємстві в інженерному відділі. Характер виконуваної роботи: оформлення спеціальної документації, постійна робота з комп’ютером та оргтехнікою.

У приміщенні знаходиться: два джерела безперебійного живлення, модем, комп'ютери, принтер, копіювальний апарат, факс, телефон.

У приміщенні присутні наступні небезпечні та шкідливі фактори:

а) фізичні фактори:

1) Підвищений рівень електромагнітних випромінювань від електронних приладів та дисплеїв персональних комп’ютерів, який може спричинити погіршення роботи серцево-судинної системи та погіршити зір людини.

2) Підвищена напруга в електричному ланцюзі, замкнення якого може пройти крізь тіло людини.

3) Підвищений рівень шуму на робочому місці. Джерела – системний блок ЕОМ, принтер. Викликає швидку стомлюваність, роздратування, головний біль, безсоння, незручність спілкування, зниження продуктивності праці.

4) Недостатня освітленість робочих поверхонь у зоні відео терміналів викликає швидку втому, погіршує працездатність, може привести до професійних захворювань (робоча міопія (короткозорість), спазм акомодації).

5) Надмірна освітленість, яка сприяє швидкій втомі органів зору і може привести до осліплення працівників.

6) Підвищений рівень вібрації.

б) хімічні фактори:

1) Зниження кількості аеронів у повітрі, що призводить до виникнення захворювань дихальних шляхів.

2) У приміщеннях з дисплеями необхідно контролювати рівень аероіонізації. Варто враховувати, що м’яке рентгенівське випромінювання на струмопровідних ділянках схеми викликають іонізацію повітря з утворенням позитивних іонів, які є шкідливими для людини.

3) Особливу увагу необхідно звертати на дотримання гранично допустимих концентрацій озону і окислів азоту в повітрі під час роботи іонізаторів.

4) Пил (ГДК=10 ), що викликає подразнення дихального тракту та слизових оболонок.

в) психофізіологічні:

1) Перш за все відноситься вимушене положення тіла під час роботи, тривала напруга окремих груп м'язів. При тривалому стоячому положенні може виникнути плоскостопість, варикозне розширення вен, зміни хребта (кіфоз).

2) Гіподинамія (обмеження рухової активності) найбільш характерна для різних видів розумової праці. Обмеження рухової активності призводить до зниження функціональних можливостей м'язової системи: зменшення силових показників, витривалості, тонусу м'язів і т.д.

3) Перенапруга зору при роботі з екранними пристроями.

**4.2 Профілактичні заходи з охорони праці**

Дослідження зорових функцій у людей, протягом декількох років працюючих за екранами ПК, виявило зниження обсягу акомодації в порівнянні з віковою нормою й більшою частотою короткозорості в порівнянні з людьми того ж віку, не пов'язаних з комп'ютером. В осіб, що пред'являли вищеописані скарги, всі ці зміни були виражені більш різко. Дослідження впливу самої роботи з дисплеєм на зір показало, що за робочу зміну відбувається зменшення обсягу акомодації, і в деяких користувачів розвивається тимчасова (так називана помилкова) короткозорість.

У сучасному житті без комп'ютера вже не обійтися. Але як з «неминучого зла» перетворити його в дійсно корисного помічника?

Не зневажайте відвідуванням офтальмолога, і не займайтеся самолікуванням.

Використайте спеціальні краплі для очей.

Робіть обов'язкові паузи під час роботи на близькій відстані через кожні 20-30 хвилин.

Важливе значення має правильне організовування робочого місця й раціональний режим роботи.

Придбайте спеціальні окуляри із прогресивними лінзами, у яких зона ясного бачення відповідає переміщенню погляду при роботі на різних відстанях. Застосування таких окулярів в інтенсивних користувачів ПК дало зниження зорового стомлення й поліпшення показників акомодації в порівнянні зі звичайними окулярами в 85% працівників.

При дотриманні перерахованих рекомендацій зменшується кількість помилок оператора, особливо в другій половині дня зменшуються дратівливість і головні болі, поліпшується емоційний стан. В окулярах з комп'ютерним фільтром комфортно в приміщенні, освітленому штучними джерелами світла, (особливо люмінесцентними лампами), тому що окуляри поліпшують спектральний склад світла, що попадає в очі. У них комфортно на вулиці, у похмуру погоду - видно чіткіше й контрастніше, а в сонячний день вони не пропускають в очі дуже активну короткохвильову частину спектра. Таким чином, окуляри з комп'ютерним фільтром можуть бути рекомендовані для постійного носіння. А це дуже важливо, тому що більше 50% комп’ютерщиків - люди в окулярах.

Сухість очей досить розповсюджена проблема серед користувачів ПК. Вважають, що причин широкого поширення сухості око серед користувачів ПК трохи:

* більше рідке моргання користувача при роботі на комп'ютері (частота моргання становить приблизно третину від звичайної частоти);
* широко розкриті очі при розгляданні зображення на моніторі, це приводить до збільшення швидкості випару сльози з поверхні ока.

Для зменшення синдрому «сухого ока» при роботі на комп'ютері треба, насамперед, забезпечити правильні умови роботи:

* правильно встановити екран монітора (центр на 10-20см нижче око користувача);
* вибрати правильно відстань (не менш 50см) до монітора;
* крім того, користувачам комп'ютера варто нагадувати про необхідність більш часто моргати при перших ознаках сухості око;
* треба так само час від часу закрити очі й зробити трохи кругових (обертальних) рухів;
* корисно влаштовувати 2-3-ох хвилинні перерви при роботі за монітором, фокусуючи погляд в далині.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному та доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Неприпустимим є підключення ЕОМ, периферійних пристроїв ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ до .звичайної двопровідної електромережі, в тому числі — з використанням перехідних пристроїв.

При розташуванні в приміщенні за його периметром до 5 персональних ЕОМ, використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволяється прокладання їх без металевих труб та гнучких металевих рукавів.

Металеві труби та гнучкі металеві рукави повинні бути заземлені. Заземлення повинно відповідати вимогам ДНАОП 0.00-1.21-98 "Правила безпечної експлуатації електроустановок споживачів". Заземлені конструкції, що знаходяться у приміщеннях (батереї опалення, водопровідні труби, кабелі із заземленим відкритим екраном тощо), мають бути надійно захищені діелектричними щитками або сітками від випадкового дотику.

Є неприпустимими:

* експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізольованими провідниками;
* застосування саморобних продовжувачів, які не відповідають вимогам ПВЕ до переносних електропроводок;
* застосування для опалення приміщення нестандартного (саморобного) електронагрівального обладнання або ламп розжарювання;
* користування пошкодженими розетками, розгалужувальними та з’єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;
* підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканиною та іншими горючими матеріалами, експлуатація їх зі знятими ковпаками (розсіювачами);
* використання електроапаратури та приладів в умовах, що не відповідають вказівкам (рекомендаціям) підприємств-виготовлювачів.

Будь-які перепланування, зміни функціонального призначення приміщень здійснювати тільки за наявності проектної документації, яка пройшла попередню експертизу на відповідність нор­мативним актам з питань пожежної безпеки з позитивним результатом в органах державного пожежного нагляду.

Меблі та устаткування мають розміщатися так, щоб забезпечити вільний евакуаційний прохід до виходу з приміщення (завширшки не менше 1,0 м). Евакуаційні шляхи та виходи слід завжди утримувати вільними, нічим не захаращеними. В міру накопичення горючих відходів (використаного паперу тощо), а також після закінчення роботи їх слід прибирати у спеціально відведені сміттєзбірники.

Електромережі, електроприлади та апаратура повинні експлуатуватися тільки у справному стані з урахуванням вказівок і рекомендацій заводів-виробників. У разі пошкоджень електромереж, вимикачів, розеток та інших електроприладів слід негайне вимкнути їх і вжити необхідних заходів щодо приведення до пожежобезпечного стану.

Документи, папір та інші горючі матеріали слід зберігати на відстані, не менше: 1,0 м — від електрощитів, електрокабелів, проводів: 0,5 м — від світильників: 0,25 м — від приладів опалення.

Шляхи евакуації, що не мають природного освітлення, у разі наявності людей повинні постійно освітлюватись електричним світлом.

Електрощити, групові електрощитки повинні бути оснащені схемами підключення споживачів з пояснювальними написами і вказаним значенням номінального струму апарата захисту (плавкої вставки).

Встановлення на горючі основи (конструкції) електророзеток, вимикачів, перемикачів та інших подібних апаратів допускається тільки з підкладанням під них суцільного негорючого матеріалу, що виступає за габарити апарата не менш ніж на 0,01 м.

Засоби протипожежного захисту слід утримувати у справному стані. Всі працівники в офісі зобов'язані вміти користуватися наявними вогнегасниками, іншими первинними засобами пожежогасіння та внутрішніми пожежними кранами, знати місця їх розташування. Відстань від найвіддаленішого місця офісу до найближчого вогнегасника не повинна перевищувати 20 м.

Пожежні сповіщувачі повинні функціонувати цілодобово і постійно утримуватися в чистоті. До них має бути забезпечений вільний доступ. Відстань від складованих матеріалів і устаткування до сповіщувачів повинна бути не менше 0,6 м.

У всіх, незалежно від призначення, приміщеннях, які після закінчення роботи замикаються і не контролюються черговим персоналом, з усіх електроустановок та електроприладів, а також з мереж їх живлення повинна бути відключена напруга (за винятком чергового освітлення, протипожежних та охоронних установок, а також електроустановок, що за вимогами техно­логії працюють цілодобово).

У службових приміщеннях (офісах) забороняється:

* Улаштовувати тимчасові електромережі, застосовувати саморобні некалібровані плавкі вставки в запобіжниках і саморобні подовжувачі, які не відповідають вимогам Правил улаштування електроустановок, експлуатувати світильники зі знятими ковпаками (розсіювачами);
* Використовувати вимикачі та штепсельні розетки для розвішування на них одягу або інших предметів, обгортати електролампи й світильники папером, заклеювати ділянки електропроводки горючою тканиною, папером;
* Користуватися електрочайниками, мікрохвильовими печами та ін. (окрім спеціально відведених та обладнаних місць), залишати без нагляду ввімкнені в електромережу кондиціонери, комп'ютери, інше електроустаткування тощо, порушувати правила експлуатації електроприладів;
* Захаращувати підходи до засобів пожежогасіння, використовувати пожежні кранкомплекти і пожежний інвентар не за призначенням;
* Використовувати для зберігання документів, різних матеріалів, предметів та інвентарю шафи (ніші) інженерних комунікацій;
* Курити (крім спеціально відведених місць, позначених написом "Місце для куріння" та обладнаних урною або попільничками з негорючого матеріалу), проводити зварювальні та інші вогневі роботи без оформлення відповідного дозволу, застосовувати ЛЗР.

Відповідальний за пожежну безпеку після закінчення роботи зобов'язаний:

* Ретельно оглянути всі службові приміщення, пересвідчитись у тому, що нема порушень, які можуть призвести до виникнення пожежі;
* Вимкнути освітлення, знеструмити прилади та устаткування (за винятком електроустаткування, яке відповідно до вимог технології має працювати цілодобово).

**ВИСНОВКИ**

В результаті проведеної роботи були вивчені способи апроксимації функції методом найменших квадратів. В якості апроксимуючих кривих були обрані: пряма, квадратна, кубічна, експоненціальна, показова, степенева, логарифмічна та гіперболічна функції.

Був складений алгоритм вирішення задачі різними способами.

В результаті виконання курсової роботи були реалізовані наступні завдання:

• Складання блок-схеми алгоритму апроксимації залежності за допомогою методу найменших квадратів.

• Складання програми для апроксимації залежності за допомогою методу найменших квадратів на мові С # в Microsoft Visual C # 2017.

В ході виконання дипломної роботи були вирішені типові інженерні завдання обробки даних, використовуючи методи матричної алгебри, і вирішення системи лінійних алгебраїчних рівнянь. Навички, набуті в процесі виконання курсової роботи, є основою для використання обчислювальних методів прикладної математики і техніки програмування в процесі вивчення всіх наступних дисциплін, а також при виконанні курсових і дипломних проектів.

В ході даної роботи:

Освоєно типові обчислювальні методи прикладної математики;

Удосконалено навички розробки алгоритмів і побудови програм на мові C#.

Удосконалено принципи модульного та ООП програмування та техніки використання підпрограм.

Таким чином, поставлені цілі курсової роботи були виконані.

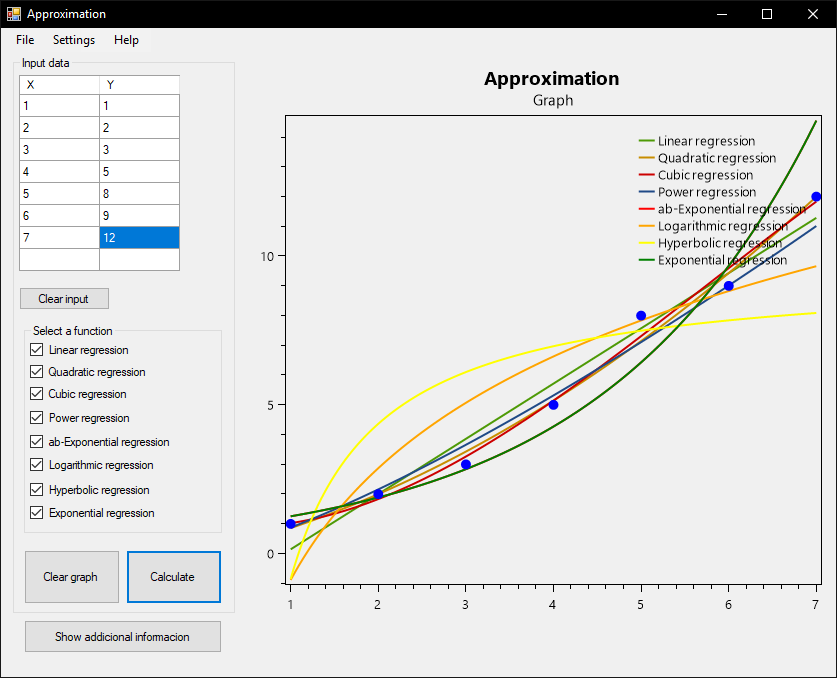
**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Чисельні методи Н.Н. Каліткін 2011 р.
2. Буч Г. Обьектно-ориентированное проектирование с примерами применения: пер. с англ. -М.: Конкорд, 1992. - 512 с.
3. Архангельський А.Я. Программирование в С++ Builder 6. – Москва: БИНОМ 2003. – 1140с.
4. Трофименко О.Г. Основи програмування. Теорія та практика – Одеса: Фенікс 2010. – 544с.
5. Грицюк Ю.І., Рак Т.Є. Об'єктно-орієнтоване програмування мовою С++ : навчальний посібник. – Львів : Вид-во Львівського ДУ БЖД, 2011. – 404 с
6. Подбельский В. В. Программирование на язьіке Си++ / В. В. Подбельский. - М.: Финансьі и статистика, 2007. - 560 с.
7. Лафоре Р. Обьектно-ориентированное программирование в C++/ Р. Лафоре. - 4-е изд. - Питер, 2004. - 923 с.
8. С# 4.0 и платформа .NET 4 для профессионалов : Пер. С англ. – М. : ООО “И.Д.Вильямс”, 2011. – 1440с. : ил. – Парал. Тит. Англ.
9. С# 5.0 и платформа .NET 4.5 6-е узд. : Пер. С англ. – М. : ООО “И.Д.Вильямс”, 2013. – 1312с. : ил. – Парал. Тит. Англ.
10. С#. Сборник рецептов. – СПБ. : БХВ-Петербург, 2007. – 432c. : ил. ISBN 5-494157-969-1 .
11. C# 2008 и платформа .NET 3.5 для профессионалов. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2009-1392c. : ил. – Парал. Тит. Англ.
12. Язык програмирования C# 2010 и платформа .NET 4.0, 5-е изд. : Пер. с англ. – М. : ООО “И.Д. Вильямс”, 2011. – 1392с. : ил. – Парал. Тит. Англ.
13. Електронна енциклопедія Вікіпедія: [Электронный ресурс] – режим доступа: <https://www.wikipedia.org/>

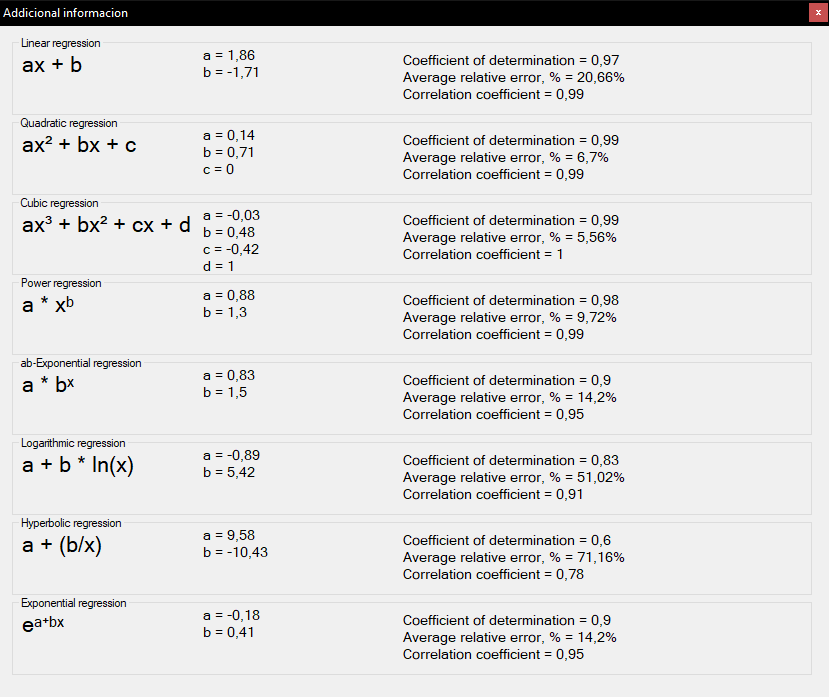
**ДОДАТОК А**

**Дизайн програми**

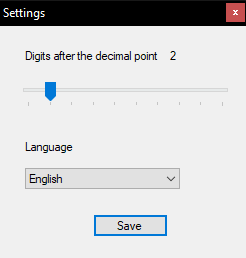
Головне меню з всіма обраними типами апроксимації.



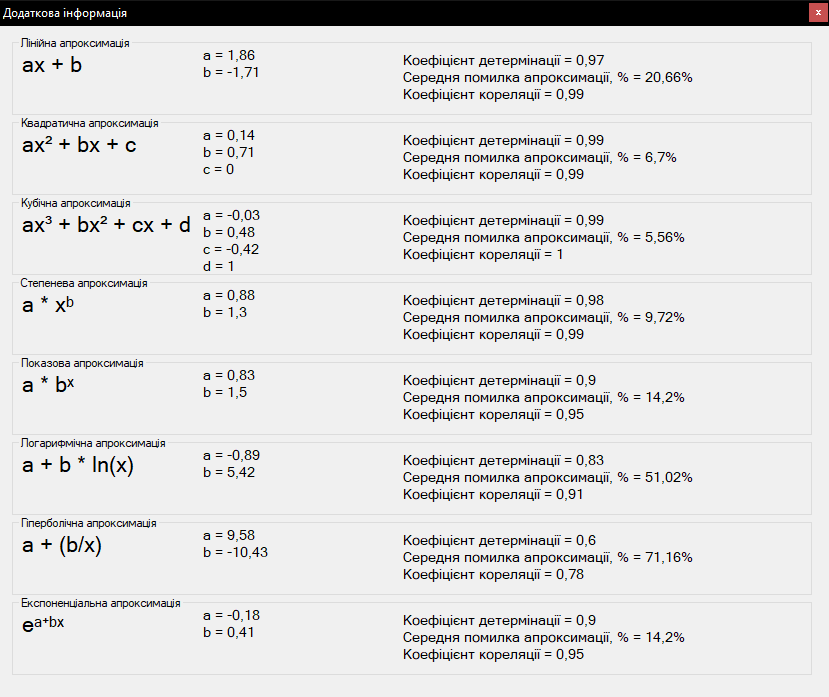
Форма додаткової інформації з усіма типами апроксимації на англійській мові.



Форма налаштувань.



Форма додаткової інформації з усіма типами апроксимації на українській мові.



**ДОДАТОК Б**

**Код програми**

**MainForm.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Resources;

using System.Windows.Forms;

using Approximation.Properties;

using Approximation.Regression;

using OxyPlot;

using OxyPlot.Series;

using OxyPlot.WindowsForms;

namespace Approximation

{

public partial class MainForm : Form

{

PlotModel myModel;

public static ResourceManager rm;

public static int rmDigits;

List<double> x = new List<double>();

List<double> y = new List<double>();

public MainForm()

{

this.InitializeComponent();

myModel = new PlotModel();

rm = new ResourceManager("Approximation.Properties." + Settings.Default["Language"], Assembly.GetExecutingAssembly());

rmDigits = Convert.ToInt32(Settings.Default["Digits"]);

checkBox1.Text = rm.GetString("Linear");

checkBox2.Text = rm.GetString("Quadratic");

checkBox3.Text = rm.GetString("Cubic");

checkBox4.Text = rm.GetString("Power");

checkBox5.Text = rm.GetString("abExponential");

checkBox6.Text = rm.GetString("Logarithmic");

checkBox7.Text = rm.GetString("Hyperbolic");

checkBox8.Text = rm.GetString("Exponential");

button1.Text = rm.GetString("Calculate");

button2.Text = rm.GetString("ClearGraph");

button3.Text = rm.GetString("ClearInput");

button4.Text = rm.GetString("ShowInf");

groupBox1.Text = rm.GetString("InputData");

groupBox2.Text = rm.GetString("SelectFunction");

fileToolStripMenuItem.Text = rm.GetString("File");

settingsToolStripMenuItem.Text = rm.GetString("Settings");

helpToolStripMenuItem.Text = rm.GetString("Help");

exportToolStripMenuItem.Text = rm.GetString("Export");

aboutToolStripMenuItem1.Text = rm.GetString("About");

exitToolStripMenuItem.Text = rm.GetString("Exit");

}

//Коли клітинка пуста, вона заповнюється нулем

private void WhenCellIsEmptySetToZero()

{

dataGridView1.AllowUserToAddRows = false; //видаляє останній рядок

foreach (DataGridViewRow row in dataGridView1.Rows)

{

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

if (string.IsNullOrEmpty(row.Cells[i].Value as string))

{

row.Cells[i].Value = 0;

}

}

}

}

//кнопка для малювання графіку

private void button1\_Click(object sender, EventArgs e)

{

Graph.graphList.Clear();

if ((dataGridView1.Rows.Count != 1))

{

WhenCellIsEmptySetToZero();

clearPlot();

myModel = new PlotModel();

//Заповнює дані з таблиці

x.Clear();

y.Clear();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

x.Add(Convert.ToInt32(row.Cells[0].Value));

y.Add(Convert.ToInt32(row.Cells[1].Value));

}

dataGridView1.AllowUserToAddRows = true;

Graph graph = new Graph(x, y);

myModel = graph.getModel(checkBoxes());

this.plot1.Model = myModel;

}

}

//Отримає вибрані типи апроксимації

List<FunctionSeries> checkBoxes()

{

List<FunctionSeries> functionList = new List<FunctionSeries>();

if (checkBox1.Checked)

{

functionList.RemoveAll(o => o.Title == rm.GetString("Linear"));

Linear linear = new Linear(x, y);

Graph.graphList.Add(linear);

functionList.Add(new FunctionSeries(linear.function, x.Min(), x.Max(), 0.0001, rm.GetString("Linear")));

}

else

{

functionList.RemoveAll(o => o.Title == rm.GetString("Linear"));

}

if (checkBox2.Checked)

{

functionList.RemoveAll(o => o.Title == rm.GetString("Quadratic"));

Quadratic quadratic = new Quadratic(x, y);

Graph.graphList.Add(quadratic);

functionList.Add(new FunctionSeries(quadratic.function, x.Min(), x.Max(), 0.0001, rm.GetString("Quadratic")));

}

else

{

functionList.RemoveAll(o => o.Title == rm.GetString("Quadratic"));

}

if (checkBox3.Checked)

{

functionList.RemoveAll(o => o.Title == rm.GetString("Cubic"));

Cubic cubic = new Cubic(x, y);

Graph.graphList.Add(cubic);

functionList.Add(new FunctionSeries(cubic.function, x.Min(), x.Max(), 0.0001, rm.GetString("Cubic")));

}

else

{

functionList.RemoveAll(o => o.Title == rm.GetString("Cubic"));

}

if (checkBox4.Checked)

{

functionList.RemoveAll(o => o.Title == rm.GetString("Power"));

Power power = new Power(x, y);

Graph.graphList.Add(power);

functionList.Add(new FunctionSeries(power.function, x.Min(), x.Max(), 0.0001, rm.GetString("Power")));

}

else

{

functionList.RemoveAll(o => o.Title == rm.GetString("Power"));

}

if (checkBox5.Checked)

{

functionList.RemoveAll(o => o.Title == rm.GetString("abExponential"));

abExponential abexpo = new abExponential(x, y);

Graph.graphList.Add(abexpo);

functionList.Add(new FunctionSeries(abexpo.function, x.Min(), x.Max(), 0.0001, rm.GetString("abExponential")));

}

else

{

functionList.RemoveAll(o => o.Title == rm.GetString("abExponential"));

}

if (checkBox6.Checked)

{

functionList.RemoveAll(o => o.Title == rm.GetString("Logarithmic"));

Logarithmic logarithmic = new Logarithmic(x, y);

Graph.graphList.Add(logarithmic);

functionList.Add(new FunctionSeries(logarithmic.function, x.Min(), x.Max(), 0.0001, rm.GetString("Logarithmic")));

}

else

{

functionList.RemoveAll(o => o.Title == rm.GetString("Logarithmic"));

}

if (checkBox7.Checked)

{

functionList.RemoveAll(o => o.Title == rm.GetString("Hyperbolic"));

Hyperbolic hyperbolic = new Hyperbolic(x, y);

Graph.graphList.Add(hyperbolic);

functionList.Add(new FunctionSeries(hyperbolic.function, x.Min(), x.Max(), 0.0001, rm.GetString("Hyperbolic")));

}

else

{

functionList.RemoveAll(o => o.Title == rm.GetString("Hyperbolic"));

}

if (checkBox8.Checked)

{

functionList.RemoveAll(o => o.Title == rm.GetString("Exponential"));

Exponential exponential = new Exponential(x, y);

Graph.graphList.Add(exponential);

functionList.Add(new FunctionSeries(exponential.function, x.Min(), x.Max(), 0.0001, rm.GetString("Exponential")));

}

else

{

functionList.RemoveAll(o => o.Title == rm.GetString("Exponential"));

}

return functionList;

}

//Кнопка для очищення графіка

private void button2\_Click(object sender, EventArgs e)

{

clearPlot();

}

//Очищення графіка

private void clearPlot()

{

myModel.Title = "";

myModel.Series.Clear();

foreach (var axis in myModel.Axes)

axis.Reset();

myModel.ResetAllAxes();

myModel.InvalidatePlot(true);

GC.Collect();

GC.WaitForPendingFinalizers();

}

//Зберігає графік в форматі png

private void pngToolStripMenuItem\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFile = new SaveFileDialog();

string desktop = Environment.GetFolderPath(Environment.SpecialFolder.Desktop); //desktop path

saveFile.InitialDirectory = desktop;

saveFile.Filter = "Image Files(\*.png) |\*.png;";

saveFile.Title = rm.GetString("SaveImg");

saveFile.FileName = "graph";

saveFile.AddExtension = true;

saveFile.DefaultExt = "png";

saveFile.FilterIndex = 2;

saveFile.RestoreDirectory = true;

if (saveFile.ShowDialog() == DialogResult.OK)

{

var pngExporter = new PngExporter { Width = 600, Height = 400, Background = OxyColors.White };

pngExporter.ExportToFile(myModel, saveFile.FileName);

}

}

//Зберігає графік в форматі pdf

private void pdfToolStripMenuItem\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFile = new SaveFileDialog();

string desktop = Environment.GetFolderPath(Environment.SpecialFolder.Desktop); //desktop path

saveFile.InitialDirectory = desktop;

saveFile.Filter = "Pdf Files(\*.pdf) |\*.pdf;";

saveFile.Title = rm.GetString("SaveImg");

saveFile.FileName = "graph";

saveFile.AddExtension = true;

saveFile.DefaultExt = "pdf";

saveFile.FilterIndex = 2;

saveFile.RestoreDirectory = true;

if (saveFile.ShowDialog() == DialogResult.OK)

{

var pdfExporter = new PdfExporter { Width = 600, Height = 400, Background = OxyColors.White };

pdfExporter.ExportToFile(myModel, saveFile.FileName);

}

}

//Зберігає графік в форматі svg

private void svgToolStripMenuItem\_Click(object sender, EventArgs e)

{

SaveFileDialog saveFile = new SaveFileDialog();

string desktop = Environment.GetFolderPath(Environment.SpecialFolder.Desktop); //desktop path

saveFile.InitialDirectory = desktop;

saveFile.Filter = "Svg Files(\*.svg) |\*.svg;";

saveFile.Title = rm.GetString("SaveImg");

saveFile.FileName = "graph";

saveFile.AddExtension = true;

saveFile.DefaultExt = "svg";

saveFile.FilterIndex = 2;

saveFile.RestoreDirectory = true;

if (saveFile.ShowDialog() == DialogResult.OK)

{

var svgExporter = new OxyPlot.WindowsForms.SvgExporter { Width = 600, Height = 400 };

svgExporter.ExportToFile(myModel, saveFile.FileName);

}

}

//Забороняє вводити в таблицю щось крім цифр

private void dataGridView1\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

try

{

e.Control.KeyPress +=

new KeyPressEventHandler(Control\_KeyPress);

}

catch (Exception)

{

}

}

private void Control\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar)

&& !char.IsDigit(e.KeyChar)

&& e.KeyChar != '.')

{

e.Handled = true;

}

if (e.KeyChar == '.'

&& (sender as TextBox).Text.IndexOf('.') > -1)

{

e.Handled = true;

}

}

//Кнопка виходу

private void exitToolStripMenuItem\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void button3\_Click(object sender, EventArgs e)

{

dataGridView1.Rows.Clear();

}

//Кнопка додаткової інформації

private void button4\_Click(object sender, EventArgs e)

{

//if there is no graph drawn show error message

if ((Graph.graphList != null) && (!Graph.graphList.Any()))

{

string message = rm.GetString("NoGraph");

string title = rm.GetString("Error");

MessageBox.Show(message, title);

}

else //create form

{

FormCollection fc = Application.OpenForms;

bool isShown = false;

//search for form named Form2

foreach (Form frm in fc)

{

if (frm.Name == "Form2")

isShown = true;

}

fc = null;

//if form is not already opened

if (isShown == false)

{

var myForm = new InfForm(Graph.graphList); //create form

myForm.Show();

}

}

}

//Settings button

private void settingsToolStripMenuItem\_Click(object sender, EventArgs e)

{

FormCollection fc = Application.OpenForms;

bool isShown = false;

//search for form named Form3

foreach (Form frm in fc)

{

if (frm.Name == "Form3")

isShown = true;

}

fc = null;

//if form is not already opened

if (isShown == false)

{

var myForm = new SettingsForm();

myForm.Show(); //create form

}

}

//Кнопка про програму

private void aboutToolStripMenuItem1\_Click(object sender, EventArgs e)

{

string message = rm.GetString("AboutText");

string title = rm.GetString("About");

MessageBox.Show(message, title);

}

}

}

**InfForm.cs**

using Approximation.Regression;

using System;

using System.Collections.Generic;

using System.Resources;

using System.Windows.Forms;

namespace Approximation

{

public partial class InfForm : Form

{

private List<Graph> approx;

private static ResourceManager rm;

public InfForm(List<Graph> approx)

{

InitializeComponent();

rm = MainForm.rm;

this.approx = approx;

createDetails(approx);

this.Text = rm.GetString("AddInf");

this.Size = new System.Drawing.Size(835, 60+80\*this.approx.Count);

}

void createDetails(List<Graph> approx)

{

int digits = MainForm.rmDigits;

for (int i = 0; i< approx.Count;i++)

{

Label label1 = new Label();

Label label2 = new Label();

Label label3 = new Label();

Label label4 = new Label();

Label labela = new Label();

Label labelb = new Label();

Label labelc = new Label();

Label labeld = new Label();

GroupBox groupBox1 = new GroupBox();

groupBox1.Controls.Add(label1);

groupBox1.Controls.Add(label2);

groupBox1.Controls.Add(label3);

groupBox1.Controls.Add(label4);

groupBox1.Location = new System.Drawing.Point(12, 80\*i+10);

groupBox1.Size = new System.Drawing.Size(800, 80);

groupBox1.Text = approx[i].getName();

label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 16F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

label1.Location = new System.Drawing.Point(17, 80 \* i + 25);

//label1.Size = new System.Drawing.Size(220, 55);

label1.Size = new System.Drawing.Size(185, 55);

label1.Text = approx[i].getFuncText().ToString();

label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

label2.Location = new System.Drawing.Point(400, 80 \* i + 25);

label2.Size = new System.Drawing.Size(400, 18);

label2.Text = rm.GetString("Determination") + " = " + Math.Round(approx[i].getDet(),digits).ToString();

label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

label3.Location = new System.Drawing.Point(400, 80 \* i + 42);

label3.Size = new System.Drawing.Size(400, 18);

label3.Text = rm.GetString("ErrorAp") + " = " + Math.Round(approx[i].getRelativeError(),digits).ToString() + "%";

label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

label4.Location = new System.Drawing.Point(400, 80 \* i + 59);

label4.Size = new System.Drawing.Size(400, 18);

label4.Text = rm.GetString("Correlation") + " = " + Math.Round(approx[i].getR(),digits).ToString();

labela.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

labela.Location = new System.Drawing.Point(200, 80 \* i + 20);

labela.Size = new System.Drawing.Size(400, 18);

labela.Text = "a = " + Math.Round(approx[i].getA(),digits).ToString();

labelb.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

labelb.Location = new System.Drawing.Point(200, 80 \* i + 37);

labelb.Size = new System.Drawing.Size(400, 18);

labelb.Text = "b = " + Math.Round(approx[i].getB(),digits).ToString();

Controls.Add(label1);

Controls.Add(label2);

Controls.Add(label3);

Controls.Add(label4);

Controls.Add(labela);

Controls.Add(labelb);

if (approx[i].getC() != 1/0F)

{

labelc.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

labelc.Location = new System.Drawing.Point(200, 80 \* i + 54);

labelc.Size = new System.Drawing.Size(400, 16);

labelc.Text = "c = " + Math.Round(approx[i].getC(),digits).ToString();

Controls.Add(labelc);

}

if (approx[i].getD() != 1 / 0F)

{

labeld.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

labeld.Location = new System.Drawing.Point(200, 80 \* i + 71);

labeld.Size = new System.Drawing.Size(400, 16);

labeld.Text = "d = " + Math.Round(approx[i].getD(),digits).ToString();

Controls.Add(labeld);

}

Controls.Add(groupBox1);

}

}

}

}

**SettingsForm.cs**

using Approximation.Properties;

using System;

using System.Resources;

using System.Windows.Forms;

namespace Approximation

{

public partial class SettingsForm : Form

{

public static ResourceManager rm;

public SettingsForm()

{

InitializeComponent();

}

private void Form3\_Load(object sender, EventArgs e)

{

rm = MainForm.rm;

label1.Text = rm.GetString("DigitsText");

label3.Text = rm.GetString("Language");

button1.Text = rm.GetString("Save");

this.Text = rm.GetString("Settings");

label2.Text = Settings.Default["Digits"].ToString();

trackBar1.Value = Convert.ToInt32(Settings.Default["Digits"]);

this.TopMost = true;

if (Settings.Default["Language"].ToString() == "ua")

comboBox1.SelectedIndex = 0;

if (Settings.Default["Language"].ToString() == "en")

comboBox1.SelectedIndex = 1;

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

label2.Text = trackBar1.Value.ToString();

}

private void button1\_Click(object sender, EventArgs e)

{

Settings.Default["Digits"] = trackBar1.Value;

if (comboBox1.SelectedIndex == 0)

Settings.Default["Language"] = "ua";

if (comboBox1.SelectedIndex == 1)

Settings.Default["Language"] = "en";

Settings.Default.Save();

string message = rm.GetString("RestartText");

string title = rm.GetString("Saved");

MessageBox.Show(message, title);

this.Close();

}

}

}

**Funcs.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation

{

class Funcs

{

//Сума масиву

static public double sum(List<double> arr)

{

return arr.Sum();

}

//Сума двох масивів

static public double sum(List<double> x, List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += x[i] \* y[i];

}

return sum;

}

//Сума масива в степені N

static public double sumPow(List<double> x, int power)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow(x[i], power);

}

return sum;

}

//Сума масива y і x в степені N

static public double sumPow(List<double> x, List<double> y, int power)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow(x[i], power) \* y[i];

}

return sum;

}

//Сума логарифма натурального від масиву

static public double sumLn(List<double> x)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Log(x[i]);

}

return sum;

}

static public double sumLnX(List<double> x, List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Log(x[i]) \* y[i];

}

return sum;

}

//Сума двох масивів від логарифма натурального

static public double sumLn(List<double> x, List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += (Math.Log(x[i]) \* Math.Log(y[i]));

}

return sum;

}

//Сума логарифма натурального від масиву в степені N

static public double sumPowLn(List<double> x, int power)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow(Math.Log(x[i]), power);

}

return sum;

}

}

}

**Dots.cs**

using OxyPlot;

using OxyPlot.Series;

using System.Collections.Generic;

namespace Approximation

{

class Dots

{

public static ScatterSeries getScatter(List<double> x, List<double> y)

{

var scatterSeries = new ScatterSeries { MarkerType = MarkerType.Circle };

for (int i = 0; i < x.Count; i++)

scatterSeries.Points.Add(new ScatterPoint(x[i], y[i]));

return scatterSeries;

}

}

}

**Matrix.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation

{

class Matrix

{

List<LinearEquation> rows = new List<LinearEquation>();

double[] solution;

public void AddLinearEquation(double result, params double[] coefficients)

{

rows.Add(new LinearEquation(result, coefficients));

}

public IList<double> Solve() //Returns a list of coefficients for the variables in the same order they were entered

{

solution = new double[rows[0].Coefficients.Count()];

for (int pivotM = 0; pivotM < rows.Count() - 1; pivotM++)

{

int pivotN = rows[pivotM].IndexOfFirstNonZero;

for (int i = pivotN + 1; i < rows.Count(); i++)

{

LinearEquation rowToReduce = rows[i];

double pivotFactor = rowToReduce[pivotN] / -rows[pivotM][pivotN];

rowToReduce.AddCoefficients(rows[pivotM], pivotFactor);

}

}

while (rows.Any(r => r.Result != 0))

{

LinearEquation row = rows.FirstOrDefault(r => r.NonZeroCount == 1);

if (row == null)

{

break;

}

int solvedIndex = row.IndexOfFirstNonZero;

double newSolution = row.Result / row[solvedIndex];

AddToSolution(solvedIndex, newSolution);

}

return solution;

}

private void AddToSolution(int index, double value)

{

foreach (LinearEquation row in rows)

{

double coefficient = row[index];

row[index] -= coefficient;

row.Result -= coefficient \* value;

}

solution[index] = value;

}

private class LinearEquation

{

public double[] Coefficients;

public double Result;

public LinearEquation(double result, params double[] coefficients)

{

this.Coefficients = coefficients;

this.Result = result;

}

public double this[int i]

{

get { return Coefficients[i]; }

set { Coefficients[i] = value; }

}

public void AddCoefficients(LinearEquation pivotEquation, double factor)

{

for (int i = 0; i < this.Coefficients.Count(); i++)

{

this[i] += pivotEquation[i] \* factor;

if (Math.Abs(this[i]) < 0.000000001) //Because sometimes rounding errors mean it's not quite zero, and it needs to be

{

this[i] = 0;

}

}

this.Result += pivotEquation.Result \* factor;

}

public int IndexOfFirstNonZero

{

get

{

for (int i = 0; i < Coefficients.Count(); i++)

{

if (this[i] != 0) return i;

}

return -1;

}

}

public int NonZeroCount

{

get

{

int count = 0;

for (int i = 0; i < Coefficients.Count(); i++)

{

if (this[i] != 0) count++;

}

return count;

}

}

}

}

}

**Graph.cs**

using OxyPlot;

using OxyPlot.Axes;

using OxyPlot.Series;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Resources;

namespace Approximation.Regression

{

public class Graph

{

protected static ResourceManager rm;

protected double a;

protected double b;

protected double r;

protected double det;

protected double err;

protected string name;

protected string funcText;

public Func<double, double> function;

protected List<double> x = new List<double>();

protected List<double> y = new List<double>();

static public List<Graph> graphList = new List<Graph>();

public Graph(List<double> x, List<double> y)

{

this.x = x.ToList();

this.y = y.ToList();

rm = MainForm.rm;

}

public PlotModel getModel(List<FunctionSeries> functionList)

{

var model = new PlotModel { Title = "Approximation", Subtitle = "Graph" };

foreach (FunctionSeries ser in functionList)

{

model.Series.Add(ser);

}

model.Series.Add(Dots.getScatter(x, y));

model.Axes.Add(new LinearAxis { IsPanEnabled = false, IsZoomEnabled = false, Position = AxisPosition.Bottom});

model.Axes.Add(new LinearAxis { IsPanEnabled = false, IsZoomEnabled = false, Position = AxisPosition.Left });

return model;

}

public virtual double getA()

{

return a;

}

public virtual double getB()

{

return b;

}

//

public virtual double getC()

{

return 1 / 0F;

}

public virtual double getD()

{

return 1 / 0F;

}

//Коефіцієнт детермінації

public virtual double getR()

{

return r;

}

//Коефіцієнт детермінації

public virtual double getDet()

{

return det;

}

//Середня помилка апроксимації

public virtual double getRelativeError()

{

return err;

}

public virtual string getName()

{

return name;

}

public virtual string getFuncText()

{

return funcText;

}

}

}

**abExponential.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation.Regression

{

class abExponential : Graph

{

public abExponential(List<double> x, List<double> y) : base(x, y)

{

this.x = x.ToList();

this.y = y.ToList();

b = getB();

a = getA();

r = getR();

det = getDet();

err = getRelativeError();

name = rm.GetString("abExponential");

funcText = "a \* b" + (char)0x02E3;

function = (z) => a \* Math.Pow(b, z);

}

public override double getA()

{

double part1 = Funcs.sumLn(y) / x.Count;

double part2 = Math.Log(b) / x.Count \* Funcs.sum(x);

double a = Math.Exp(part1 - part2);

return a;

}

public override double getB()

{

double part1 = x.Count \* Funcs.sumLnX(y, x) - Funcs.sum(x) \* Funcs.sumLn(y);

double part2 = x.Count \* Funcs.sumPow(x,2) - Math.Pow(Funcs.sum(x), 2);

double b = Math.Exp(part1 / part2);

return b;

}

//Коефіцієнт Кореляції

public override double getR()

{

double r = Math.Sqrt(1f - (sumCor1(y) / sumCor2(y)));

return r;

}

public double sumCor1(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

//sum += Math.Pow((y[i] - yx(y[i])), 2);

sum += Math.Pow((y[i] - a \* Math.Pow(b, x[i])), 2);

}

return sum;

}

public double sumCor2(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow((y[i] - y\_(y)), 2);

}

return sum;

}

//Коефіцієнт детермінації

public override double getDet()

{

return Math.Pow(r, 2);

}

//Середня помилка апроксимації

public override double getRelativeError()

{

double part1 = 1f / x.Count \* sumEr() \* 100f;

return part1;

}

//Сума

private double sumEr()

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Abs((y[i] - yx(x[i])) / y[i]);

}

return sum;

}

//y\*

private double yx(double X)

{

//double yx = a \* Math.Pow(X,2) + b \* X + c;

double yx = a \* Math.Pow(b,X);

return yx;

}

//y\_

private double y\_(List<double> x)

{

double y\_ = Funcs.sum(x) / x.Count;

return y\_;

}

}

}

**Cubic.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation.Regression

{

class Cubic : Graph

{

double c;

double d;

public Cubic(List<double> x, List<double> y) : base(x, y)

{

this.x = x.ToList();

this.y = y.ToList();

name = rm.GetString("Cubic");

Matrix test = new Matrix();

test.AddLinearEquation(Funcs.sum(y), Funcs.sumPow(x,3), Funcs.sumPow(x,2), Funcs.sum(x), x.Count);

test.AddLinearEquation(Funcs.sum(x,y), Funcs.sumPow(x, 4), Funcs.sumPow(x, 3), Funcs.sumPow(x, 2), Funcs.sum(x));

test.AddLinearEquation(Funcs.sumPow(x,y,2), Funcs.sumPow(x, 5), Funcs.sumPow(x, 4), Funcs.sumPow(x, 3), Funcs.sumPow(x, 2));

test.AddLinearEquation(Funcs.sumPow(x, y, 3), Funcs.sumPow(x, 6), Funcs.sumPow(x, 5), Funcs.sumPow(x, 4), Funcs.sumPow(x, 3));

var result = test.Solve();

a = result[0];

b = result[1];

c = result[2];

d = result[3];

funcText = "ax" + (char)0x00B3 + " + " +

"bx" + (char)0x00B2 + " + " + "cx" + " + " + "d";

r = getR();

det = getDet();

err = getRelativeError();

function = (z) => a \* Math.Pow(z,3)+ b\*Math.Pow(z,2) + c \* z + d;

}

public override double getC()

{

return c;

}

public override double getD()

{

return d;

}

//Коефіцієнт Кореляції

public override double getR()

{

double r = Math.Sqrt(1f - (sumCor1(y) / sumCor2(y)));

return r;

}

public double sumCor1(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow((y[i] - a \* Math.Pow(x[i], 3) - b \* Math.Pow(x[i], 2) - c \* x[i] - d), 2);

}

return sum;

}

public double sumCor2(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow((y[i] - y\_(y)), 2);

}

return sum;

}

//Коефіцієнт детермінації

public override double getDet()

{

return Math.Pow(r, 2);

}

private double y\_(List<double> x)

{

double y\_ = Funcs.sum(x) / x.Count;

return y\_;

}

//Середня помилка апроксимації

public override double getRelativeError()

{

double part1 = 1f / x.Count \* sumEr() \* 100f;

return part1;

}

//Сума

private double sumEr()

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Abs((y[i] - yx(x[i])) / y[i]);

}

return sum;

}

//y\*

private double yx(double X)

{

double yx = a \* Math.Pow(X, 3) + b \* Math.Pow(X, 2) + c \* X + d;

return yx;

}

private double[,] getMatrix()

{

double[,] mat = new double[3, 4];

mat[0, 0] = Funcs.sumPow(x, 2);

mat[0, 1] = Funcs.sum(x);

mat[0, 2] = x.Count;

mat[0, 3] = Funcs.sum(y);

mat[1, 0] = Funcs.sumPow(x, 3);

mat[1, 1] = Funcs.sumPow(x, 2);

mat[1, 2] = Funcs.sum(x);

mat[1, 3] = Funcs.sum(x, y);

mat[2, 0] = Funcs.sumPow(x, 4);

mat[2, 1] = Funcs.sumPow(x, 3);

mat[2, 2] = Funcs.sumPow(x, 2);

mat[2, 3] = Funcs.sumPow(x, y, 2);

return mat;

}

}

}

**Exponential.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation.Regression

{

class Exponential : Graph

{

public Exponential(List<double> x, List<double> y) : base(x, y)

{

this.x = x.ToList();

this.y = y.ToList();

b = getB();

a = getA();

r = getR();

det = getDet();

err = getRelativeError();

name = rm.GetString("Exponential");

funcText = "e" + (char)0x1D43 + (char)0x207A + (char)0x1D47 + (char)0x02E3;

function = (z) => Math.Pow(Math.E, a + b \* z);

}

public override double getB()

{

double part1 = x.Count \* Funcs.sumLnX(y,x) - Funcs.sum(x) \* Funcs.sumLn(y);

double part2 = x.Count \* Funcs.sumPow(x,2) - Math.Pow(Funcs.sum(x),2);

double b = part1 / part2;

return b;

}

public override double getA()

{

double a = 1f / x.Count \* Funcs.sumLn(y) - b / x.Count \* Funcs.sum(x);

return a;

}

//Коефіцієнт Кореляції

public override double getR()

{

double r = Math.Sqrt(1f - (sumCor1(y) / sumCor2(y)));

return r;

}

public double sumCor1(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

//sum += Math.Pow((y[i] - yx(y[i])), 2);

sum += Math.Pow((y[i] - Math.Pow(Math.E, a + b \* x[i])), 2);

}

return sum;

}

public double sumCor2(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow((y[i] - y\_(y)), 2);

}

return sum;

}

//Коефіцієнт детермінації

public override double getDet()

{

return Math.Pow(r, 2);

}

//Середня помилка апроксимації

public override double getRelativeError()

{

double part1 = 1f / x.Count \* sumEr() \* 100f;

return part1;

}

//Сума

private double sumEr()

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Abs((y[i] - yx(x[i])) / y[i]);

}

return sum;

}

//y\*

private double yx(double X)

{

//double yx = a \* Math.Pow(X,2) + b \* X + c;

double yx = Math.Pow(Math.E, a + b \* X);

return yx;

}

private double y\_(List<double> x)

{

double y\_ = Funcs.sum(x) / x.Count;

return y\_;

}

}

}

**Hyberbolic.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation.Regression

{

class Hyperbolic : Graph

{

public Hyperbolic(List<double> x, List<double> y) : base(x, y)

{

this.x = x.ToList();

this.y = y.ToList();

b = getB();

a = getA();

r = getR();

det = getDet();

err = getRelativeError();

name = rm.GetString("Hyperbolic");

funcText = "a + (b/x)";

function = (z) => a + b / z;

}

public override double getB()

{

double part1 = x.Count \* divYX(x, y) - div1x(x) \* Funcs.sum(y);

double part2 = x.Count \* div1xPow(x) - Math.Pow(div1x(x), 2);

double b = part1 / part2;

return b;

}

public override double getA()

{

double a = 1f / x.Count \* Funcs.sum(y) - b / x.Count \* div1x(x);

return a;

}

//Коефіцієнт Кореляції

public override double getR()

{

double r = Math.Sqrt(1f - (sumCor1(y) / sumCor2(y)));

return r;

}

public double sumCor1(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

//sum += Math.Pow((y[i] - yx(y[i])), 2);

sum += Math.Pow((y[i] - a - b / x[i]), 2);

}

return sum;

}

public double sumCor2(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow((y[i] - y\_(y)), 2);

}

return sum;

}

//Коефіцієнт детермінації

public override double getDet()

{

return Math.Pow(r, 2);

}

//Середня помилка апроксимації

public override double getRelativeError()

{

double part1 = 1f / x.Count \* sumEr() \* 100f;

return part1;

}

//Сума

private double sumEr()

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Abs((y[i] - yx(x[i])) / y[i]);

}

return sum;

}

//y\*

private double yx(double X)

{

//double yx = a \* Math.Pow(X,2) + b \* X + c;

double yx = a + b / X;

return yx;

}

private double y\_(List<double> x)

{

double y\_ = Funcs.sum(x) / x.Count;

return y\_;

}

private double divYX(List<double> x, List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += y[i] / x[i];

}

return sum;

}

private double div1x(List<double> x)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += 1f / x[i];

}

return sum;

}

private double div1xPow(List<double> x)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += 1f / Math.Pow(x[i], 2);

}

return sum;

} }}

**Linear.cs**

using Approximation.Regression;

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation

{

public class Linear : Graph

{

public Linear(List<double> x, List<double> y) : base(x, y)

{

this.x = x.ToList();

this.y = y.ToList();

a = getA();

b = getB();

r = getR();

det = getDet();

err = getRelativeError();

name = rm.GetString("Linear");

funcText = "ax + b";

function = (z) => a \* z + b;

}

public override double getA()

{

double part1 = Funcs.sum(x) \* Funcs.sum(y) - x.Count \* Funcs.sum(x, y);

double part2 = Math.Pow(Funcs.sum(x), 2) - x.Count \* Funcs.sumPow(x,2);

double a = part1 / part2;

return a;

}

public override double getB()

{

double part1 = Funcs.sum(x) \* Funcs.sum(x, y) - Funcs.sumPow(x,2) \* Funcs.sum(y);

double part2 = Math.Pow(Funcs.sum(x), 2) - x.Count \* Funcs.sumPow(x,2);

double b = part1 / part2; ;

return b;

}

//Коефіцієнт детермінації

public override double getR()

{

double part1 = x.Count \* Funcs.sum(x, y) - Funcs.sum(x) \* Funcs.sum(y);

double part2 = Math.Sqrt((x.Count \* Funcs.sumPow(x,2) - Math.Pow(Funcs.sum(x), 2)) \*

(x.Count \* Funcs.sumPow(y,2) - Math.Pow(Funcs.sum(y), 2)));

double r = part1 / part2; ;

return r;

}

//Коефіцієнт детермінації

public override double getDet()

{

return Math.Pow(r, 2);

}

//Середня помилка апроксимації

public override double getRelativeError()

{

double part1 = 1f / x.Count \* sumEr() \* 100f;

return part1;

}

public override string getFuncText()

{

return funcText;

}

//Сума

private double sumEr()

{

double sum=0;

for(int i = 0; i< x.Count;i++)

{

sum += Math.Abs((y[i] - yx(x[i])) / y[i]);

}

return sum;

}

//y\*

private double yx(double X)

{

double yx = a \* X + b;

return yx;

}

}

}

**Logarithmic.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation.Regression

{

class Logarithmic : Graph

{

public Logarithmic(List<double> x, List<double> y) : base(x, y)

{

this.x = x.ToList();

this.y = y.ToList();

b = getB();

a = getA();

r = getR();

det = getDet();

err = getRelativeError();

name = rm.GetString("Logarithmic");

funcText = "a + b \* ln(x)";

function = (z) => a + b \* Math.Log(z);

}

public override double getB()

{

double part1 = x.Count \* Funcs.sumLnX(x, y) - Funcs.sumLn(x) \* Funcs.sum(y);

double part2 = x.Count \* Funcs.sumPowLn(x, 2) - Math.Pow(Funcs.sumLn(x),2);

double b = part1 / part2;

return b;

}

public override double getA()

{

double a = 1f / x.Count \* Funcs.sum(y) - b / x.Count \* Funcs.sumLn(x);

return a;

}

//Коефіцієнт Кореляції

public override double getR()

{

double r = Math.Sqrt(1f - (sumCor1(y) / sumCor2(y)));

return r;

}

public double sumCor1(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

//sum += Math.Pow((y[i] - yx(y[i])), 2);

sum += Math.Pow((y[i] - a - b \* Math.Log(x[i])), 2);

}

return sum;

}

public double sumCor2(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow((y[i] - y\_(y)), 2);

}

return sum;

}

//Коефіцієнт детермінації

public override double getDet()

{

return Math.Pow(r, 2);

}

//Середня помилка апроксимації

public override double getRelativeError()

{

double part1 = 1f / x.Count \* sumEr() \* 100f;

return part1;

}

//Сума

private double sumEr()

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Abs((y[i] - yx(x[i])) / y[i]);

}

return sum;

}

//y\*

private double yx(double X)

{

//double yx = a \* Math.Pow(X,2) + b \* X + c;

double yx = a + b \* Math.Log(X);

return yx;

}

private double y\_(List<double> x)

{

double y\_ = Funcs.sum(x) / x.Count;

return y\_;

}

}

}

**Power.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation.Regression

{

public class Power : Graph

{

public Power(List<double> x, List<double> y) : base(x, y)

{

this.x = x.ToList();

this.y = y.ToList();

b = getB();

a = getA();

r = getR();

det = getDet();

err = getRelativeError();

name = rm.GetString("Power");

funcText = "a \* x" + (char)0X1D47;

function = (z) => a \* Math.Pow(z, b);

}

public override double getA()

{

double part1 = (1f / x.Count) \* Funcs.sumLn(y);

double part2 = (b / x.Count) \* Funcs.sumLn(x);

double a = Math.Exp(part1 - part2);

return a;

}

public override double getB()

{

double part1 = (x.Count \* Funcs.sumLn(x, y)) - (Funcs.sumLn(x) \* Funcs.sumLn(y));

double part2 = (x.Count \* Funcs.sumPowLn(x,2)) - Math.Pow(Funcs.sumLn(x),2);

double b = (part1 / part2);

return b;

}

//Коефіцієнт Кореляції

public override double getR()

{

double r = Math.Sqrt(1f - (sumCor1(y) / sumCor2(y)));

return r;

}

public double sumCor1(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

//sum += Math.Pow((y[i] - yx(y[i])), 2);

sum += Math.Pow((y[i] - a \* Math.Pow(x[i], b)), 2);

}

return sum;

}

public double sumCor2(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow((y[i] - y\_(y)), 2);

}

return sum;

}

//Коефіцієнт детермінації

public override double getDet()

{

return Math.Pow(r, 2);

}

//Середня помилка апроксимації

public override double getRelativeError()

{

double part1 = 1f / x.Count \* sumEr() \* 100f;

return part1;

}

//Сума

private double sumEr()

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Abs((y[i] - yx(x[i])) / y[i]);

}

return sum;

}

//y\*

private double yx(double X)

{

//double yx = a \* Math.Pow(X,2) + b \* X + c;

double yx = a \* Math.Pow(X, b);

return yx;

}

private double y\_(List<double> x)

{

double y\_ = Funcs.sum(x) / x.Count;

return y\_;

}

}

}

**Quadratic.cs**

using System;

using System.Collections.Generic;

using System.Linq;

namespace Approximation.Regression

{

public class Quadratic : Graph

{

private double c;

public Quadratic(List<double> x, List<double> y) : base(x, y)

{

this.x = x.ToList();

this.y = y.ToList();

findSolution(getMatrix(), ref a, ref b, ref c);

r = getR();

det = getDet();

err = getRelativeError();

name = rm.GetString("Quadratic");

funcText = "ax" + (char)0x00B2 + " + " +

"bx + " + "c";

function = (z) => a \* Math.Pow(z, 2) + b \* z + c;

}

public override double getC()

{

return c;

}

//Коефіцієнт Кореляції

public override double getR()

{

double r = Math.Sqrt(1f - (sumCor1(y) / sumCor2(y)));

return r;

}

public double sumCor1(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

//sum += Math.Pow((y[i] - yx(y[i])), 2);

sum += Math.Pow((y[i] - a \* Math.Pow(x[i], 2) - b \* x[i] - c), 2);

}

return sum;

}

public double sumCor2(List<double> y)

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Pow((y[i] - y\_(y)), 2);

}

return sum;

}

//Коефіцієнт детермінації

public override double getDet()

{

return Math.Pow(r, 2);

}

//Середня помилка апроксимації

public override double getRelativeError()

{

double part1 = 1f / x.Count \* sumEr() \* 100f;

return part1;

}

//Сума

private double sumEr()

{

double sum = 0;

for (int i = 0; i < x.Count; i++)

{

sum += Math.Abs((y[i] - yx(x[i])) / y[i]);

}

return sum;

}

//y\*

private double yx(double X)

{

//double yx = a \* Math.Pow(X,2) + b \* X + c;

double yx = Math.Pow(X, 2) \* a + b \* X + c;

return yx;

}

private double y\_(List<double> x)

{

double y\_ = Funcs.sum(x) / x.Count;

return y\_;

}

private double[,] getMatrix()

{

double[,] mat = new double[3,4];

mat[0, 0] = Funcs.sumPow(x,2);

mat[0, 1] = Funcs.sum(x);

mat[0, 2] = x.Count;

mat[0, 3] = Funcs.sum(y);

mat[1, 0] = Funcs.sumPow(x, 3);

mat[1, 1] = Funcs.sumPow(x, 2);

mat[1, 2] = Funcs.sum(x);

mat[1, 3] = Funcs.sum(x,y);

mat[2, 0] = Funcs.sumPow(x, 4);

mat[2, 1] = Funcs.sumPow(x, 3);

mat[2, 2] = Funcs.sumPow(x, 2);

mat[2, 3] = Funcs.sumPow(x, y, 2);

return mat;

}

// This functions finds the determinant of Matrix

static double determinantOfMatrix(double[,] mat)

{

double ans;

ans = mat[0, 0] \* (mat[1, 1] \* mat[2, 2] - mat[2, 1] \* mat[1, 2])

- mat[0, 1] \* (mat[1, 0] \* mat[2, 2] - mat[1, 2] \* mat[2, 0])

+ mat[0, 2] \* (mat[1, 0] \* mat[2, 1] - mat[1, 1] \* mat[2, 0]);

return ans;

}

// This function finds the solution of system of

// linear equations using cramer's rule

static void findSolution(double[,] coeff, ref double a, ref double b, ref double c)

{

// Matrix d using coeff as given in cramer's rule

double[,] d = {

{ coeff[0,0], coeff[0,1], coeff[0,2] },

{ coeff[1,0], coeff[1,1], coeff[1,2] },

{ coeff[2,0], coeff[2,1], coeff[2,2] },

};

// Matrix d1 using coeff as given in cramer's rule

double[,] d1 = {

{ coeff[0,3], coeff[0,1], coeff[0,2] },

{ coeff[1,3], coeff[1,1], coeff[1,2] },

{ coeff[2,3], coeff[2,1], coeff[2,2] },

};

// Matrix d2 using coeff as given in cramer's rule

double[,] d2 = {

{ coeff[0,0], coeff[0,3], coeff[0,2] },

{ coeff[1,0], coeff[1,3], coeff[1,2] },

{ coeff[2,0], coeff[2,3], coeff[2,2] },

};

// Matrix d3 using coeff as given in cramer's rule

double[,] d3 = {

{ coeff[0,0], coeff[0,1], coeff[0,3] },

{ coeff[1,0], coeff[1,1], coeff[1,3] },

{ coeff[2,0], coeff[2,1], coeff[2,3] },

};

// Calculating Determinant of Matrices d, d1, d2, d3

double D = determinantOfMatrix(d);

double D1 = determinantOfMatrix(d1);

double D2 = determinantOfMatrix(d2);

double D3 = determinantOfMatrix(d3);

Console.Write("D is : {0:F6} \n", D);

Console.Write("D1 is : {0:F6} \n", D1);

Console.Write("D2 is : {0:F6} \n", D2);

Console.Write("D3 is : {0:F6} \n", D3);

// Case 1

if (D != 0)

{

// Coeff have a unique solution. Apply Cramer's Rule

a = D1 / D;

b = D2 / D;

c = D3 / D; // calculating z using cramer's rule

/\*Console.Write("Value of x is : {0:F6}\n", x);

Console.Write("Value of y is : {0:F6}\n", y);

Console.Write("Value of z is : {0:F6}\n", z);\*/

}

// Case 2

/\* else

{

if (D1 == 0 && D2 == 0 && D3 == 0)

Console.Write("Infinite solutions\n");

else if (D1 != 0 || D2 != 0 || D3 != 0)

Console.Write("No solutions\n");

}\*/

}

}

}