Міністерство освіти і науки України

Український державний університет науки і технологій

Факультет Комп'ютерні технології та системи
Кафедра Комп'ютерні інформаційні технології

Пояснювальна записка

до кваліфікаційної роботи
бакалавр

на тему: «Багатопараметричні екстраполяційні моделі та алгоритми вибору та проектування»
за освітньою програмою **12 Інженерія програмного забезпечення**
зі спеціальності: **121 Інженерія програмного забезпечення**

Виконав: студент групи ПЗ1912:

(посада) (підпис) /Антон ГАРКУША/

Керівник:

(посада) (підпис) /Владислав СКАЛОЗУБ/

Нормоконтролер:

(посада) (підпис) /Світлана ВОЛКОВА/

Засвідчую, що у цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент

Дніпро – 2023 рік

Ministry of Education and Science of Ukraine

Ukrainian State University of Science and Technologies

Faculty Computer technologies and systems
Department Computer information technology

Explanatory Note

to Master's Thesis
bachelor

on the topic: « Multiparameter extrapolation models and algorithms for selection and design »

according to educational curriculum **12 software engineering**
in the Speciality: **121 software engineering**

Done by the student of the group PZ1912: ___ _____ /Anton Garkusha/
(посада) (підпис)

Scientific Supervisor: ___ _____ / Vladislav SKALOZUB /
(посада) (підпис)

Normative controller: ___ _____ / Svitlana VOLKOVA/
(посада) (підпис)

Supervisors:
Economic part ___ _____ / Mykola GNENNIYN/
(посада) (підпис)

Dnipro – 2023

Український державний університет науки і технологій

Факультет: «Комп'ютерні технології і системи»
Кафедра: «Комп'ютерні інформаційні технології»
Спеціальність: «Інженерія програмного забезпечення»

«ДО ЗАХИСТУ»
Завідувач кафедри
_____ Вадим ГОРЯЧКІН
(підпис) (ПІБ)
202_ р. _____ «_____»

ЗАВДАННЯ

до дипломної роботи на здобуття ОС «бакалавр»

студента групи _____ Гаркуша Антон Леонідович
номер групи (ПІБ)

1. Тема роботи: «Багатопараметричні екстраполяційні моделі та алгоритми вибору та проектування»
затверджена наказом по університету від «___»_____202_ р. № ___ ст.

2. Термін подання студентом роботи «20» червня 2023 р.

3. Вихідні дані до дипломної роботи Приклади постановок завдань вибору та проектування з використанням багатопараметричної екстраполяції, сфера застосування, особливості завдань і спеціалізовані математичні моделі багатопараметричної екстраполяції, основні алгоритми лінійної багатопараметричної екстраполяції, програмні засоби реалізації методу багатовимірної лінійної екстраполяції.

4. Зміст пояснювальної записки (перелік питань до розробки):

4.1 Аналітична частина: огляд досліджень та розробок щодо можливостей реалізації завдань вибору та проектування з використанням багатопараметричної екстраполяції, структура завдань та процедури формування моделей для застосування методу багатопараметричної лінійної екстраполяції (БЛЕ), постановки нових завдань сфери проектування на основі БЛЕ, аналіз можливостей удосконалення методу БЛЕ та розширення завдань щодо його застосування.

4.2 Основна частина: розробка удосконаленого методу БЛЕ для завдань з суттєво нерівними величинами областей варіювання параметрів моделей проектування та

вибору, розроблена нової процедури застосування методу БЛЕ для реалізації зворотних завдань вибору та проектування, розробка алгоритмів випадкового пошуку для реалізації завдань БЛЕ при неперервних та дискретних параметрах моделей, розробка програмних засобів для реалізації завдань БЛЕ, проведення тестування програми та числових експериментів з аналізу ефективності запропонованих алгоритмів, результати досліджень достовірності та ефективності створених програмних засобів, технічна документація.

5. Перелік демонстраційного матеріалу: презентація результатів розробки програмного забезпечення для дослідження завдань БЛЕ, результати чисельних експериментальних досліджень алгоритмів і процедур БЛЕ, програмні засоби для реалізації алгоритмів і процедур БЛЕ, відео-демонстрація функціонування програмного комплексу.

КАЛЕНДАРНИЙ ПЛАН

№ пор	Зміст роботи (розділу)	Термін виконання розділів роботи	Примітка
1	Вступ	12.09.22 – 26.10.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	27.10.22 – 04.03.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	05.03.23 – 31.04.23	
4	Постановка задачі, технічне завдання	01.05.23 – 07.05.23	
5	Техніко-економічні показники	08.05.23 – 15.05.23	
6	Розробка інструментальних засобів дослідження	16.05.23 – 21.05.23	
7	Виконання досліджень	22.05.23 – 28.05.23	
8	Оформлення результатів дипломної роботи	29.05.23 – 11.06.23	
9	Подання дипломної роботи до кафедри		
10	Захист дипломної роботи на засіданні екзаменаційної комісії	27.06.23	

Дата видачі завдання «___»_____ 202_ р.

Керівник дипломної роботи Владислав СКАЛОЗУБ
(підпис) (ПІБ)

Завдання прийняв до виконання _____

(підпис) (ІПІБ)

РЕФЕРАТ

Об’єктом дослідження та розробки дипломної роботи є багатопараметричні екстраполяційні моделі, алгоритми та програмні засоби із вибору та проектування складних систем і технологій на основі даних аналогів і прототипів.

Мета роботи полягала у розвитку постановок завдань і розробці удосконаленого методу багатопараметричної лінійної екстраполяції (БЛЕ) для завдань з суттєво нерівними величинами областей варіювання параметрів моделей проектування та вибору, а також формуванні нової процедури застосування методу БЛЕ, призначеної для реалізації зворотних завдань вибору та проектування, створення програмного забезпечення щодо їх автоматизації.

Методи дослідження . Для досягнення мети, розв'язання та опрацювання задач розробки були застосовані методи математичного моделювання та лінійної екстраполяції і багатопараметричної оптимізації, методи програмної інженерії щодо формування вимог, проектування та розробки програмного забезпечення. .

При виконанні дипломної роботи було розроблено удосконалений методу багатопараметричної лінійної екстраполяції, сформована нова процедура щодо реалізації зворотних завдань вибору та проектування, створене програмне забезпечення з автоматизації.

Результати дипломної роботи сприяють підвищенню ефективності процесів проектування та аналізу складних систем, контролю достовірності та відновленню даних та ін. шляхом застосування розроблених програмних засобів БЛЕ, за умов існування аналогів або прототипів об’єктів.

Пояснювальна записка до кваліфікаційної роботи бакалавра 99 с., 22 рис.,2 табл., 4 додатків.,241 джерел.

Ключові слова: вибір і проектування, аналоги і прототипи, лінійна екстраполяція, зворотні завдання, випадковий пошук, програмне забезпечення.

ЗМІСТ

ВСТУП

Актуальність роботи.

При вирішенні завдань проектування або удосконалення складних багато параметричних та багато функціональних систем або технологій часто мають місце невизначеності умов, вимог та узгоджених наборів показників, а також відсутність загальних повних моделей створюваних об’єктів. Разом з тим відомими являються певні попередні розробки, тобто прототипи і аналоги створюваної системи, для котрих існують структурні рішення, оцінки функціональних ознак та значення параметрів діючих систем. При цьому суттєвою обставиною, що ускладнює формування моделей і завдання із визначення величин параметрів, є непорівнянність кількості параметрів і відомих аналогів (прототипів). За таких умов використання стандартних методів оцінювання параметрів, навіть на основі формування лінійних моделей, не можливе.

Аналіз літературних і технічних джерел, а також програмних засобів із підтримки проектування складних програмно-технічних та інших систем показав, що для вирішення завдань попереднього оцінювання величин параметрів проектування широкого кола досліджуваних систем на основі даних про їх відомі аналоги можуть бути застосовані багатопараметричні екстраполяційні моделі, алгоритми та відповідні програмні засоби із вибору та проектування. Застосування методу багатопараметричної лінійної екстраполяції (БЛЕ) для наведених вище завдань досить відоме, має широке прикладне значення, пройшло апробації і показало високу ефективність на етапах попереднього аналізу. При цьому нами не було виявлено доступних програмних засобів реалізації завдань БЛЕ, як інструментарію попереднього аналізу, вибору та формування наборів параметрів систем, які проектуються.

У зв’язку з актуальністю та широкими можливостями застосування моделі БЛЕ, як засобу із вибору параметрів при проектуванні складних систем і технологій на основі даних аналогів і прототипів, в дипломній роботі обрано об’єктом дослідження та розробки саме є багатопараметричні екстраполяційні моделі, їх

удосконалені алгоритми та програмне забезпечення. При тому необхідно було ураховувати властивості об’єктів проектування та розробки щодо особливих вимог до параметрів (області значень, неперервні/дискретні), розширити можливості застосування моделі БЛЕ для нових завдань при формуванні наборів параметрів систем тощо. Наведені вище властивості методу БЛЕ та широкі можливості його застосування підтверджують доцільність виконаних розробок, з урахуванням запропонованих в роботі відмінностей у порівнянні з відомими результатами розв’язання означених завдань.

Мета роботи

Мета роботи полягає у розвитку постановок завдань і розробці удосконаленого методу багатопараметричної лінійної екстраполяції (БЛЕ) для завдань з суттєво нерівними величинами областей варіювання параметрів моделей проектування та вибору, а також у формуванні нової процедури застосування методу БЛЕ, призначеної для реалізації зворотних завдань вибору та проектування, створення програмного забезпечення щодо автоматизації процесів їх систематичного застосування.

Експлуатаційне призначення.

Призначення результатів розробки обумовлене широкими можливостями застосування методу БЛЕ, як засобу із вибору параметрів при проектуванні складних систем і технологій на основі даних аналогів і прототипів,
Значні можливості БЛЕ обумовлюють його широке прикладне значення, метод пройшов велику апробації і показав досить високу ефективність на етапах попереднього аналізу. Аналіз програмних засобів не дозволив виявити відомих та загально прийнятих доступних програмних засобів реалізації завдань БЛЕ, як інструментарію попереднього аналізу, вибору та формування наборів параметрів систем, які проектуються, на основі наборів відомих аналогів та прототипів.
Виконане в роботі удосконалення БЛЕ для реалізації завдань з суттєво нерівними величинами областей варіювання параметрів моделей проектування та вибору, а також створення нової процедури застосування методу БЛЕ, призначеної

для реалізації зворотних завдань вибору та проектування, а також створення спеціалізованого програмного забезпечення, визначають експлуатаційне призначення розробки, як програмного інструментарію з автоматизації процесів попереднього аналізу та формування наборів параметрів систем, які проектуються, на основі наборів відомих аналогів та прототипів.

1. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Методи екстраполяції

1. Сутність екстраполяції полягає у вивченні сформованих у минулому та теперішньому стійких тенденцій розвитку об’єкта прогнозу та перенесення їх на майбутнє.

При цьому розрізняють формальну та прогнозну екстраполяцію

Формальна базується на припущенні про збереження в майбутньому минулих та сьогодення тенденцій розвитку об’єкта прогнозу.

При прогнозній екстраполяції фактичне значення пов’язується з гіпотезами про динаміку досліджуваного процесу з урахуванням змін впливу різних факторів у перспективі.

Методи екстраполяції – основа екстраполяції – вчення про динамічні ряди.

Лінія тренду – трендовий аналіз (напрямок зміни рівнів показника).

Екстраполяція представляє метод прогнозування, що полягає у вивченні сформованих у минулому та сьогоденні стійких тенденцій розвитку процесів і явищ і перенесення їх на майбутнє. Метод екстраполяції застосовується, якщо використовуються такі припущення:

а) період часу, для якого побудована функція, повинен бути достатнім для виявлення тенденції розвитку;

б) аналізований процес є стійко динамічним і має інерційністю, тобто. для значних змін характеристик процесу потрібен час;

в) не очікується сильних зовнішніх впливів на процес, що вивчається, які можуть серйозно вплинути на тенденцію розвитку. Прогнозування за допомогою методу екстраполяції – один із найпростіших методів статистичного прогнозування. Його використання виправдане при недостатньому знанні про природу явища, що вивчається, або відсутність даних, необхідних для застосування більш досконалих методів прогнозування.

Розрізняють

а) просту екстраполяцію, яка передбачає, що всі діяли в минулому і сьогоденні тенденції збережуться в повному обсязі, тому що всі фактори, що діяли, залишаться незмінними;

б) прогнозну екстраполяцію, яка базується на припущенні про зміну факторів, що визначають динаміку досліджуваного процесу або явища. До цього розділу відносять рекурентні методи БЛЕ [4].

2.Метод нормативний – одна із основних методів. Сутність даного методу полягає в тому, що на основі заздалегідь встановлених норм та техніко-економічних нормативів розраховується потреба суб'єкта господарювання ресурсів та їх джерел (норми, потреби в оборотних коштах, норми кредиторської заборгованості, нормативи відрахувань до фондів).

Переваги методу – простота використання. Недоліки методу – неможливість встановлення економічно обґрунтованих нормативів стосовно всіх видів діяльності суб’єкта господарювання.

3.Расчетно-аналитический метод – сутність даного методу у тому, що з урахуванням аналізу досягнутих величин показників, прийнятих за основу та індексів їх зміни у плановому періоді, розраховуються планові величини цих показників.

Даний метод планування широко застосовується у тих випадках, коли відсутні техніко-економічні нормативи, а взаємозв’язок між показниками може бути побічно на основі аналізу їх динаміки та зв’язків. В основі цього лежить експертна оцінка.

Цей метод застосовується під час планування суми прибутків і доходів, визначення величини відрахувань від прибутку, розрахунку величин оборотних засобів.

4.Балансовий метод – передбачає розробку балансів, що становлять систему показників, у якій одна частина, що характеризує ресурси за джерелами надходження дорівнює іншій частині, що показує розподіл у всіх напрямах їх витрат. Система балансів у використовуваних у прогнозуванні та плануванні включає: матеріальні, трудові та фінансові. Центральне місце приділяється матеріальним балансам. З

їхньою допомогою пов’язують виробництво та споживання конкретних видів продукції. обґрунтовується виробнича програма підприємства з урахуванням матеріальних балансів.

Всі матеріальні баланси складаються, як правило, з двох частин: ресурсної (інформація про основні джерела надходження) та розподільної (основні напрямки споживання).

Переваги методу: простота узгодження взаємозалежних показників.

Недоліки методу: складно використовуватиме прогнозування значень планових показників (т.к. ситуації швидко змінюються а метод орієнтований відносну стабільність ситуації).

5.Економіко математичне моделювання – полягає у встановленні кількісних взаємозв’язків між показниками та факторами їх визначальними. Цей зв'язок виражається через економіко- математичну модель. Економіко-математична модель – є математичним описом економічного процесу, тобто. опис факторів, що характеризують структуру та закономірності зміни даного економічного явища за допомогою математичних символів та прийомів.

Переваги методу: зручність застосування для прогнозування динаміки математичних показників, опис взаємозв’язків між різними техніко-економічними параметрами.

Недоліки методу: велика кількість припущень може привести до нереалістичних результатів моделювання. Закони, що лежать в основі розвитку економічних систем, що використовуються при моделюванні, можуть змінювати і впливати на ступінь адекватності моделі.

6. Мережеве планування – це сума способів та прийомів, що дозволяють на основі застосування мережевого графіка, раціонально планувати, організовувати, координувати та контролювати будь- який комплекс взаємопов’язаних та послідовно виконуваних робіт. Мережеве планування список джерел Мережевий графік

складається з трьох частин: робота (ті операції), подія (ті результати), шлях (механізм операцій).

Переваги методу: дозволяє наочно уявити організаційну і технологічну послідовність виконуваних операцій, встановити взаємозв'язок з-поміж них, виявити операції, яких залежить тривалість всієї роботи і зосередити ними увагу.

Недоліки методу: наявність повної інформації (зокрема терміни виконання окремих операцій), слабкий облік людського чинника і складність застосування нижньому рівні менеджменту.

7.Програмно-цільовий метод список джерел – передбачає розробку економічної програми задля досягнення певної мети. Оцінку необхідних цього ресурсів, встановлення термінів і відповідальних осіб.

Переваги методу: дозволяє сконцентрувати увагу на певному напрямі, полегшує контроль робіт,
прозорість з ресурсного забезпечення та чіткий механізм відповідальності.

Недоліки методу: неможливість розкладання усієї діяльності підприємства на проекти та програми; більш високі витрати (реалізацію конкретних програм); відсутність обмежень за параметрами та ресурсами.

8. SWOP аналіз – це виявлення сильних та слабких сторін організації, можливості та загроз у зовнішньому середовищі їх кількісна та якісна оцінка та використання результатів у стратегічному плануванні.[10,11]

Переваги методу: можливість комплексної оцінки внутрішнього та зовнішнього середовища та вибору «правильної стратегії розвитку»

Недоліки методу: проблеми формалізації та кількісної оцінки окремих блоків.

9. Портфельний аналіз – це інструмент за допомогою якого керівництво підприємства виявляє та оцінює господарську діяльність з метою вкладення коштів у найбільш прибуткові та перспективні напрямки та припинення інвестицій у неефективні проекти.

Позитивні якості методу: дозволяє узгодити стратегії бізнес одиниць корпорації та загальну корпоративну стратегію, дозволяє розподілити ресурси між господарськими підрозділами.

Недоліки методу: використання даних про поточний стан бізнесу чи підприємства, які завжди можна екстраполювати у майбутнє, неможливість врахування безлічі чинників.[13][14][15][16]

1.2. Методи лінійної екстраполяції багатьох змінних

Завдання оптимального проектування складної системи полягає у визначенні параметрів, що доставляють екстремум визнаного критерію якості системи в заданій ситуації. Під ситуацією можна розуміти, наприклад, вихідні дані проектування: умови роботи системи — обмеження деякі її характеристики, технологія виробництва та т.п.

Залежність оптимальних параметрів системи від ситуації, в якій проводиться оптимізація, зазвичай буває невідомою. Тому будь-які зміни у ситуації, типові процесу проектування, призводять до необхідності проведення нових оптимальних розрахунків для уточнення параметрів системи. Такий спосіб коригування оптимальних параметрів значно подовжує процес проектування, оскільки кожен розрахунок, як правило, потребує великих витрат часу. Цілком природно, виникає завдання оцінки оптимальних параметрів за інформацією, що накопичена в результаті проектування аналогічних систем або виконання розрахунків на оптимальність у ряді аналогічних ситуацій. При цьому передбачається, що процес знаходження параметрів, близьких до оптимальних у новій ситуації повинен відбуватися без проведення трудомісткої процедури багатопараметричної оптимізації.

Розглянемо можливість вирішення цього завдання методом багатовимірної лінійної екстраполяції .

Нехай ефективність проекту визначається скалярною функцією двох векторних аргументів X і У

$$Q = Q(\mathbf{X}, \mathbf{Y}) \tag{1.1}$$

Для невеликої кількості ситуацій **X** відомі значення векторів оптимальних

рішень **Y**, що доставляють екстремум критерію якості Q.

$$Q(\mathbf{X_i}, \mathbf{Y}) \rightarrow \mathbf{Y}_i^*\left(\overline{1,k}\right) \quad (1.2)$$

Ставиться завдання знаходження (без проведення трудомісткої оптимізації) вектору Y_{k+1}, близького у певному сенсі до вектору Y^*_{k+1}, оптимального в новій заданій ситуації X_{k+1}. Нехай функція

$$\mathbf{Y}^* = \mathbf{F}(\mathbf{X}) \quad (1.3)$$

означає невідому залежність оптимальних параметрів від ситуації. Тоді рішення задачі (1.2) можна розглядати як результати *k* спостережень цієї невідомої функції

$$\mathbf{X}_i \rightarrow \mathbf{Y}_i^*, i = \overline{1,k} \quad (1.4)$$

Отже, задача зводиться до відновлення невідомої векторної функції **F** векторного аргументу **X** та відповідного йому значення вектору $\mathbf{Y}^*$.

Множина векторів можливих ситуацій **X** позначимо через **{X}**, а відповідну множину векторів рішень **Y** – через **{Y}**.

Вибір способу рішення поставленої задачі, очевидно, залежить від кількості наявної інформації про відновлювану функцію. Якщо число *k* спостережень достатню велике порівняно з розмірністю вектору випадків, то задачу можна розв'язати апроксимаційними методами. Якщо чисто *k* спостережень порівняно з розмірністю вектору **X** і достатньо для побудови базису у просторі **{X}**, то можна побудувати лінійну модель функції (1.3).

Розглянемо випадок, коли число k спостережень мало та недостатньо для апріорної побудови лінійної моделі.

$$\mathbf{Y}^* = L(\mathbf{X}), k < n + 1 \quad (1.5)$$

де n – розмірність вектору **X**.

Для рішення задачі в умовах інформаційної недостатності, через обмеження типу (1.6), зробимо наступним чином:

побудуємо лінійну модель функції (1.5) на підпросторі **{X'}** та **{Y'}**, створених спостереженнями (1.4)

введемо додаткову гіпотезу для довизначення функції на просторах **{X}** та **{Y}**.

Рівняння для елементів векторних підпросторів (гіперплощин) **{X'}** та **{Y'}** мають вид

$$\{\mathbf{X}'\} = \mathbf{X}_1 + \sum_{i=1}^{k-1} \lambda_i (\mathbf{X}_{i+1} - \mathbf{X}_1); \quad (1.6)$$

$$\{\mathbf{y}'\} = \mathbf{y}_1 + \sum_{i=1}^{k-1} \mu_i (\mathbf{y}_{i+1} - \mathbf{y}_1). \quad (1.7)$$

Припускаючи рішення У оптимальним. використовуватимемо цей символ і далі замість У*.

Лінійну модель функції (1.3) на підпросторах {X`} і {У'} будують, вводячи наступну гіпотезу.

Гіпотеза 1. Перетворення X'-> У' лінійно. Тоді для задоволення відповідності (1.4) стигне покласти

$$\lambda_i = \mu_i, \; i = \overline{1, k-1}. \quad (1.8)$$

Тепер задача відновлення функції(1.3) зводиться до визначення перетворення {X}->{Y} на основі прийнятої гіпотези про лінійність перетворення {x'} {У'}. Якби кількість спостережень було більше n +1, то (з огляду на те, що окремі спостереження можуть виявитися лінійно залежними при деякому k підпросторі { x'} збіглося б з лінійною моделлю простору { X }, і завдання відновлення функції (27) мала єдине рішення - лінійну модель функції. За наявності обмеження k < n + 1 завдання допускає безліч рішень. Ця неоднозначність рішення усувається запровадженням додаткової гіпотези.

Попередньо введемо деяку випуклу скалярну функцію, яку назвемо функцією близькості

$$\Phi = \Phi(\mathbf{X}, \mathbf{X}'), \quad \text{де } \mathbf{X} \in \{\mathbf{X}\}, \mathbf{X}' \in \{\mathbf{X}'\}. \quad (1.9)$$

Кожній парі ситуацій X і X` ця функція ставить у відповідність число, що характеризує віддаленість (близькість) однієї ситуації від іншої.

Гіпотеза 2. Кожному вектору X E { X ставиться у відповідність до такої вектор X' ∈ { X' } який доставляє мінімум функції близькості

(1.9). Наприклад, якщо функцію близькості вибрати у вигляді

$$\Phi(\mathbf{X}, \mathbf{X}') = \|\mathbf{X} - \mathbf{X}'\|^2, \quad (1.10)$$

то гіпотеза 2 набуває наступного чіткого геометричного змісту: кожному вектору ситуацій X ∈ { X } ставиться у відповідність його ортогональна проекція X' на лінійне підпростір { X` } .

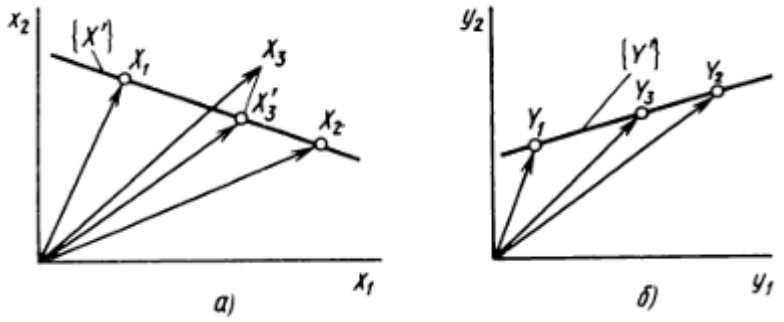

Рисунок 1.2.1 – Екстраполяція за двома навчальними векторами на площині:
а - простір ситуацій; б – простір рішень

Параметри λ_i, що визначають вектор X' ∈{ X' } , найближчий відповідно до функції (1.9) до вектору X ∈{ X }можна знайти із системи рівняння

$$\partial \Phi / \partial \lambda_i = 0, i = \overline{1, k-1}. \quad (1.11)$$

Введення зазначених гіпотез дозволяє побудувати наступний оператор перетворення { X }→{ Y }тобто. алгоритм відновлення функції (1.3):

1) перетворимо { X } →{ X'} за допомогою гіпотези 2;

2) відображаємо {X'}→ {У'} за допомогою гіпотези 1;

3) ототожнюємо {Y} = {У}.

Скорочено алгоритм можна записати так:

{x}→{x'}→{У'} = {У}.

Проілюструємо ідею методу простому прикладі двовимірного простору ситуацій.

Для побудови алгоритму екстраполяції введемо два лінійні параметризовані простори (рис. 1.2.1): двовимірний простір ситуацій X, де кожна ситуація визначається парою параметрів x_1 і x_2, і простір рішень У, в якому кожне рішення визначається двома параметрами y_1 і y_2 . Далі запропонуємо, що відомі дві проектні

ситуації X_1 і X_2, яким у просторі рішень відповідають рішення $У_1$ і $У_2$. Завдання ставиться таким чином: дана нова проектна ситуація X_3, і потрібно знайти для неї рішення $У_3$, маючи лише вказану інформацію. Розв'язання цього завдання відповідно до основної ідеї методу багатовимірної екстраполяції буде мати наступний вигляд.

Через відомі ситуації X_1 і X_2 проведемо пряму лінію, що є підпростором проектних ситуацій {X'}, а через точки $У_1$ та $У_2$ - лінію, що є підпростором рішень {У'} (див.

рис. 1). Для нової проектної ситуації X_3 визначимо найближчу точку на підпросторі ситуацій, навіщо опустимо перпендикуляр із точки X_3 на лінію X_1 X_2. Отримаємо точку X_3', що є відображенням точки X= на підпросторі ситуацій, яка ділить відрізок X_1 X_2

у певній пропорції. Розділивши відрізок $У_1$ $У_2$ у просторі {Y'} в

цієї пропорції, отримаємо проектне рішення $У_3$, відповідне ситуації X3.

Відповідно до викладеного алгоритму підпростору ситуацій і рішень знаходимо за такими формулами:

$$\begin{cases} \{X'\} = X_1 + \lambda (X_2 - X_1), \\ \{У'\} = У_1 + \mu (У_2 - У_1), \end{cases} \quad (1.12)$$

де λ і μ - коефіцієнти пропорційності.

Гіпотеза 1 про лінійність перетворення {X'}→ {У'} в даному прикладі означає наступне: якщо ситуація X лежить на прямій { X' }, то відповідний вектор оптимальних рішенні У лежить на прямої {У'}.

Для задоволення вихідних відповідностей $X_1 \to У_1, X_2 \to У_2$

гадаємо, що λ = μ (перша задана відповідність задовольняється при λ = μ = 0, друге – при λ = μ = 1). Таким чином, введення гіпотези 1 в даному прикладі дозволяє відшукувати вектор, оптимальний (з точністю до гіпотези лінійності) в ситуації X, що лежить на прямій {X'} . Якщо ситуація X не лежить на цій прямій (як, наприклад, X_3), вводимо міру близькості між ситуаціями (функцію близькості). Ситуацію, що не лежить на прямій {x'}, ототожнюємо з найближчою до неї

ситуацією, що лежить на прямій. У цьому полягає гіпотеза 2. Якщо в даному прикладі функцію близькості вибрати у вигляді (1.10), то найближчою до ситуації X3 виявиться ситуація X_{31} (проекція точки X_3 на пряму), що доставляє мінімум функції близькості. На основі гіпотези близькості ситуацію X_3 ототожнюємо з ситуацією x;. Застосовуючи гіпотезу про лінійність, знаходимо вектор $У_3$, оптимальний у ситуації X , за наступною наближеною формулою:

$$\mathbf{У_3} = \mathbf{У_1} + \mu\,(\mathbf{У_2} - \mathbf{У_1}),\ \text{где}\ \mu = \lambda. \quad (1.13)$$

Наведена форма методу багатовимірної екстраполяції дозволяє достовірно вирішити задачу відновлення функції в умовах інформаційної недостатності в просторах малої розмірності. З ростом розмірності простору зростають обчислювальні труднощі, пов'язані зі зберіганням великих обсягів інформації в пам'яті комп'ютера. Цих труднощів можна уникнути, якщо розрахунки проводити періодично. [1]

Початковий алгоритм багатовимірної лінійної екстраполяції заснований на припущенні, що формує підпростір ситуацій проектування { X } є фіксованим і стаціонарним. При цьому алгоритм можна інтерпретувати наступним чином: Розрізняють фіксовані і фіксовані вектори X_1 і X_2, що характеризують простір ситуацій; необхідно знайти оптимальний множник L, який мінімізує відстань в евклідової метриці між новою розрахунковою ситуацією X_3 і відомим підпростором {X} Як показують наведені вище приклади, багатовимірний алгоритм екстраполяції, як правило, забезпечує досить високу точність при вирішенні завдань, пов'язаних з прийняттям оперативних рішень в умовах неповної інформації. Це Висновок тим вірніше, чим повніше виконуються вимоги, що випливають із суті методу (мова йде про виконання гіпотези про лінійність простору, виборі послідовностей навчання і т. д.). Якщо зазначено перераховані вище вимоги виконуються в повній мірі і при постановці завдання, яка повинна вирішуватися методом багатовимірної екстраполяції, допускається деяка «довільність», зокрема, «невдало» вибираються навчальні послідовності, тоді рішення може бути отримано з великою похибкою. І тут

цілком доречно розглянути способи побудови модифікованих алгоритмів, що забезпечують більш високу точність при вирішенні завдань екстраполяції.

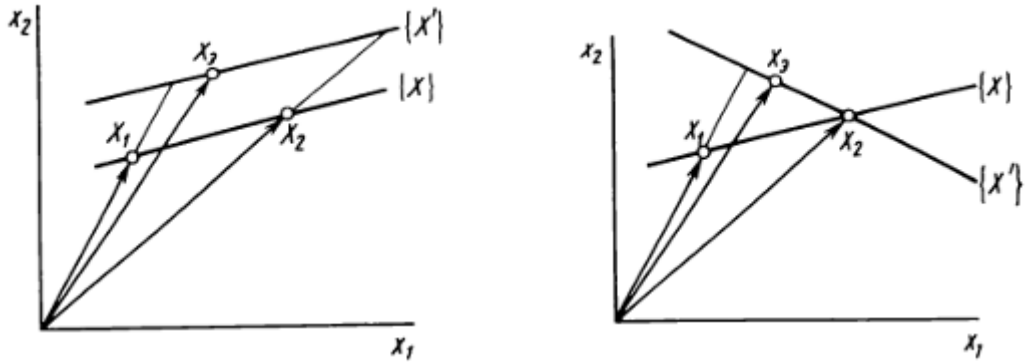

Рисунок 1.2.2 – методи зсуву та повороту підпростору ситуацій

Для підвищення точності лінійної екстраполяції, використовуючи аксіоми лінійного простору, можна дещо інакше підійти до формування підпростору ситуацій проектування.

Зокрема, ми використовуємо ідею рухомих векторних оболонок і намагаємося змістити або повернути підпростір ситуацій таким чином, щоб наблизити підпростір {X'} до нового вектору X (рис. 5, 6). Давайте докладніше розглянемо обидва методи.

Зміщуючи підпростір ситуацій, вводимо параметр p і формуємо вектори pX_1 і pX_2. Тоді підпростір ситуацій можна записати в вигляді

$$\left\{\mathbf{X}'\right\} = p\mathbf{X}_1 + \lambda\,(p\mathbf{X}_2 - p\mathbf{X}_1) = p\,[\mathbf{X}_1 + \lambda\,(\mathbf{X}_2 - \mathbf{X}_1)]. \quad (1.14)$$

Далі побудуємо функцію близькості $\Phi(X'\text{-}X_E)$, яка буде залежати від параметрів p та λ:

$$\Phi\,(\mathbf{X}' - \mathbf{X}_э) = \left\{p\,[\mathbf{X}_1 + \lambda\,(\mathbf{X}_2 - \mathbf{X}_1)] - \mathbf{X}_э\right\}^2 \quad (1.15)$$

Щоб мінімізувати цю строго опуклу функцію, досить знайти похідні від параметрів p та λ прирівняти їх до нуля, тобто.

$$\Phi_\lambda' = 2\left\{p\,[\mathbf{X}_1 + \lambda\,(\mathbf{X}_2 - \mathbf{X}_1)] - \mathbf{X}_э\right\}[p\,(\mathbf{X}_2 - \mathbf{X}_1)] = 0; \quad (1.16)$$

$$\Phi_p' = 2\left\{p\,[\mathbf{X}_1 + \lambda\,(\mathbf{X}_2 - \mathbf{X}_1)] - \mathbf{X}_э\right\}[\mathbf{X}_1 + \lambda\,(\mathbf{X}_2 - \mathbf{X}_1)] = 0. \quad (1.17)$$

З рис. 1.2.2(а) випливає, що в двовимірному просторі при p та λ, знайденої в

результаті вирішення рівнянь (1.16) і (1.17), значення близькості буде не тільки мінімальним, але і буде дорівнює нулю, тобто задача екстраполяції буде вирішена без помилки. Природно, що при навчальній послідовності з двох векторів в тривимірному просторі і в просторі більшої розмірності це вже не буде вірно, тобто в загальному випадку виникне якась остаточна помилка при вирішенні завдання екстраполяції.

$$\left\{\mathbf{X}'\right\} = p\,[\mathbf{X}_1 + \sum_{i=1}^{k-1} \lambda_i\,(\mathbf{X}_{i+1} - \mathbf{X}_1)]; \tag{1.18}$$

$$\Phi\,(\mathbf{X}' - \mathbf{X}_э) = \sum_{l=1}^{n} \left\{ p\,[\mathbf{X}_{1\,l} + \sum_{i=1}^{k-1} \lambda_i\,(\mathbf{X}_{i+1} - \mathbf{X}_1)_l - \mathbf{X}_{э\,l}] \right\}^2. \tag{1.19}$$

Аналогічно двовимірному випадку, вирішуємо задачу мінімізації функції (1.19) і знаходимо значення p та λ;

Екстрапольований розчин отримують за формулою

$$\mathbf{У}_э = p\,[\mathbf{У}_1 + \sum_{i=1}^{k-1} \lambda_i\,(\mathbf{У}_{i+1} - \mathbf{У}_1)]. \tag{1.20}$$

Можлива й інша форма запису рівняння (1.14), яка легко виходить після нескладних перетворень:

$$\left\{\mathbf{X}'\right\} = p\mathbf{X}_1 + \lambda\,(p\mathbf{X}_2 - p\mathbf{X}_1) = \mathbf{X}_1\,(p - \lambda p) + \mathbf{X}_2\,\lambda p. \tag{1.21}$$

У багатовимірному випадку, тобто якщо в послідовності k векторів, формула (1.22.б) матиме вигляд

$$\left\{\mathbf{X}'\right\} = \sum_{i=1}^{k} \lambda_i^*\,\mathbf{X}_i. \tag{1.23}$$

$$\Phi\,(\mathbf{X}' - \mathbf{X}_э) = \sum_{l=1}^{n} [\sum_{i=1}^{k} \lambda_i^*\,\mathbf{X}_i - \mathbf{X}_э]_l^2. \tag{1.24}$$

Далі методика вирішення задачі екстраполяції залишається колишньою, тобто ми згортаємо квадратну форму (74), вирішуємо систему рівнянь, в результаті чого знаходимо параметри λ_i^*.

Подібного результату можна досягти обертанням підпростору ситуацій

проектування (див. рис. 1.2.2.,). [1]

Отже за два вищеописаних підрозділи ми сформували сутність екстраполяційних методів та математичну модель вирішення цього класу задач.

1.3. Аналіз аналогів

Серед аналогів не існую саме програми для вирішення задачі багатопараметричної лінійної екстраполяції. Загалом аналоги можна поділити на дві категорії: програми лінійної екстраполяції для однієї змінної (здебільшого це сайти з онлайн калькулятором) та бібліотеки мов програмування, які включають в себе інструментарій для роботи з БЛЕ.

З сайтів можна виділити наступні:

1. x-engineer.org/linear-interpolation-extrapolation-calculator

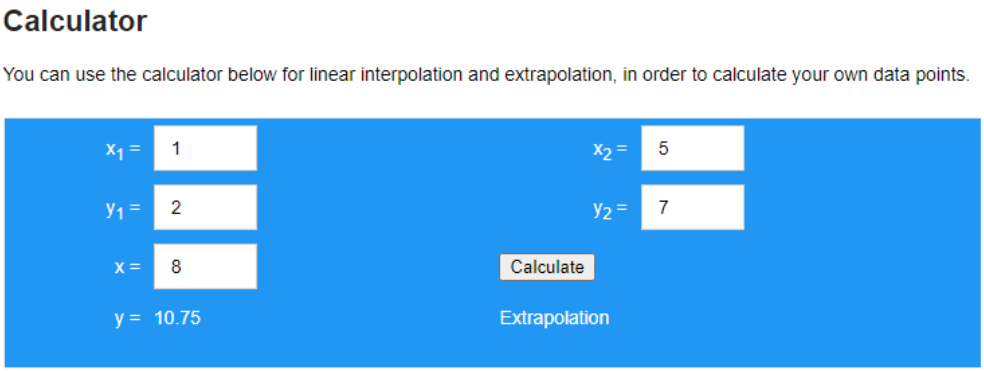

Рисунок 1.3.1 – Калькулятор екстраполяції на сайті «x-engineer.org»

2. www.perfmatrix.com/linear-extrapolation-calculator

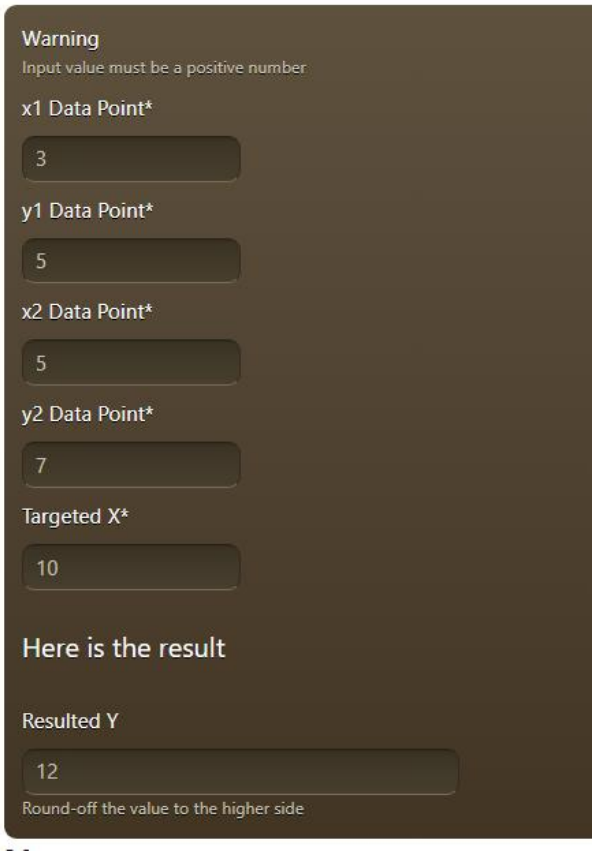

Рисунок 1.3.2 – Калькулятор екстраполяції на сайті «perfmatrix.com»

Можна помітити, що всі сайти для екстраполяції створенні для роботи з двома змінними, двома парами вхідних даних та парою вхідного X та вихідного Y. В цілому весь інтерфейс зводиться до п’яти числових полів та поля виводу шуканого числа.

Цей принцип я також буду використовувати у інтерфейсі своєї програми, але вже для динамічної кількості змінних та екземплярів даних.

Приклади бібліотек для роботи з БЛЕ:

1. Python:

- NumPy: Бібліотека для наукових обчислень у Python, що включає функції для лінійної екстраполяції у багатовимірному просторі.

- SciPy: Бібліотека для наукових та інженерних обчислень, що надає методи та функції для екстраполяції даних, включаючи лінійну екстраполяцію.[13]

- pandas: Бібліотека для аналізу та обробки даних, що містить функції для лінійної екстраполяції у багатовимірних датасетах.

2. MATLAB:

- Curve Fitting Toolbox: Набір інструментів для апроксимації та екстраполяції даних, включаючи можливості для багатовимірної лінійної екстраполяції.

- Optimization Toolbox: Бібліотека для оптимізації та моделювання, яка також надає функції для лінійної екстраполяції.

3. R:

- stats: Вбудована бібліотека мови R, що надає функції для статистичного аналізу даних, включаючи методи екстраполяції, у тому числі лінійну екстраполяцію.

1.4. Формування вимог

1. Введення даних: Програма повинна надавати користувачеві можливість введення початкових даних, включаючи відомі значення вихідної функції або набір даних, на основі яких буде проводитись екстраполяція.
2. Перевірка даних: Програма повинна здійснювати перевірку введених даних на коректність та цілісність, щоб унеможливити помилки або неправильні результати.
3. Лінійна екстраполяція: Програма має реалізовувати метод лінійної екстраполяції, який дозволяє на основі відомих значень функції знаходити значення функції за межами заданого діапазону.
4. Підтримка різних функцій: Програма має бути здатною працювати з різними типами функцій, включаючи лінійні, квадратичні та інші.
5. Вирішення зворотної задачі: Програма повинна надавати змогу навпаки відновлювати втрачені, або зіпсовані дані у вхідних масивах
6. Незалежність від розміру вхідних величин: програма повинна рівнозначно враховувати як великі величини так і малі.
7. Гнучкість налаштувань: Програма повинна надавати користувачеві можливість налаштування відображення таблиць. параметрів екстраполяції, таких як діапазон значень, крок екстраполяції та інші параметри.

8. Обробка помилок: Програма повинна бути стійкою до можливих помилок і винятків, забезпечуючи адекватну обробку та повідомлення про проблеми, що виникають.
9. Ефективність та швидкість роботи: Програма має бути оптимізована для забезпечення високої швидкості роботи та ефективного використання ресурсів, особливо під час роботи з великими обсягами даних.
10. Документація та підтримка: Програма має супроводжуватися докладною документацією, що пояснює її функціональність та використання. Також має бути надана можливість отримання підтримки або відповіді на запитання користувача.

Висновки

В даному розділі ми сформували основні проблеми, які можна вирішувати багато параметричною екстраполяцією, знайшли потенціал для впровадження новизни, вивели основний математичний апарат вирішення цього класу задач. Проаналізувавши аналоги прийшли до висновку, що програми для вирішення саме багато параметричних задач методами екстраполяції не існує, або їх досить важко знайти звичайному користувачу, та винесли основні принципи побудови графічного та програмного інтерфейсу додатку з досвіду схожих за фунціоналом програм та бібліотек.

2. ПРОЕКТУВАННЯ

2.1. Зовнішнє програмування

2.1.1. Функціональне призначення

Функціональне призначення клієнтської програми для вирішення задач багатопараметричної лінійної екстраполяції можна сформулювати так:

Метою клієнтської програми є забезпечення зручного та ефективного вирішення задач багатопараметричної лінійної екстраполяції. Програма надає користувачеві інтуїтивно зрозумілий інтерфейс та набір функцій, що дозволяють вводити та обробляти багатовимірні дані, проводити лінійну екстраполяцію та отримувати точні та достовірні результати.

Основні функції клієнтської програми включають:

1. Введення та обробку даних:
- Можливість вводити багатовимірні дані, надавати відомі значення параметрів вихідної функції.
- Перевірка коректності та цілісності введених даних для виключення можливих помилок.
- Можливість імпорту та експорту даних з різних форматів для полегшення роботи з наявними наборами даних.

2. Багатопараметрична лінійна екстраполяція:
- Реалізація алгоритмів багатопараметричної лінійної екстраполяції для знаходження значень функції поза заданим діапазоном.
- Врахування меж похибки та забезпечення точності результатів екстраполяції.

3. Гнучкість та налаштування:
- Надання користувачеві можливості налаштування параметрів екстраполяції,

включаючи діапазон значень, крок екстраполяції та інші параметри.

- Простий та інтуїтивно зрозумілий інтерфейс налаштування, що дозволяє користувачеві легко керувати процесом екстраполяції відповідно до вимог завдання.

4. Стійкість та повідомлення про помилки:

- Забезпечення стійкості програми до можливих помилок і винятків, забезпечуючи адекватну обробку та повідомлення про проблеми, що виникають.
- Надання користувачеві зрозумілих та інформативних повідомлень про помилки, що допомагають у розумінні та усуненні проблем, що виникли.

Клієнтська програма розроблена з урахуванням простоти використання, ефективності обчислень та точності результатів, щоб полегшити користувачеві вирішення завдань багатопараметричної лінійної екстраполяції та покращити якість одержуваних результатів.

2.1.2. Експлуатаційне призначення

Експлуатаційне призначення програми полягає у наданні користувачеві засобу для виконання точної та надійної екстраполяції значень функції за межами заданого діапазону на основі відомих даних. Програма може бути використана в різних областях та додатках, де потрібне проведення екстраполяції на основі багатовимірних даних.

Деякі приклади експлуатаційного призначення програми включають:

1. Інженерні розрахунки: Програма може використовуватися для передбачення значень фізичних або хімічних параметрів на основі виміряних даних, дозволяючи інженерам та науковим дослідникам проводити розрахунки та оцінювати поведінку систем за межами доступних даних.[12][18]

2. Фінансова аналітика: У фінансових моделях та аналітиці програма може бути

застосована для прогнозування та оцінки фінансових показників, таких як ціни акцій, вартість інвестицій або прибутковість портфеля на основі відомих даних.[19][7]

3. Прогнозування та планування: Програма може бути використана для прогнозування майбутніх значень показників та параметрів, що дозволяє планувати та приймати рішення на основі передбачень за межами наявних даних.[5][6][8][19][20]

4. Моделювання та симуляція: У наукових та інженерних моделях, а також у симуляціях процесів, програма може використовуватися для заповнення прогалин у даних та розширення уявлення про систему, дозволяючи отримати більш повне уявлення про поведінку та результати моделювання поза відомим діапазоном.[5][6][9][12]

5. Медична діагностика: В галузі медичної діагностики програма може використовуватися для прогнозування та оцінки параметрів здоров'я на основі наявних даних, що допомагає лікарям та фахівцям приймати більш поінформовані рішення та робити точні прогнози.[20]

Експлуатаційне призначення програми полягає у забезпеченні користувачам можливості проводити екстраполяцію значень функцій з високою точністю та достовірністю, що може мати важливе значення у різних галузях та додатках, де потрібна робота з багатовимірними даними та прогнозування за межами відомих даних.

2.1.3. Функціональні вимоги

Функціональні вимог:

1. Введення даних:
    a. Можливість введення багатовимірних даних, включно зі значеннями

відомих параметрів вихідної функції.

b. Підтримка різних форматів даних для імпорту та експорту (наприклад JSON, CSV, Excel та ін.).
c. Перевірка коректності введених даних і обробка можливих помилок.

2. Обробка та аналіз даних:
    a. Реалізація алгоритмів для багатопараметричної лінійної екстраполяції на основі введених даних.
    b. Підтримка можливості зазначення діапазону екстраполяції та кроку екстраполяції.
    c. Обробка та врахування похибок даних під час проведення екстраполяції.

3. Налаштування та параметри:
    a. Можливість налаштування параметрів екстраполяції, включно з діапазоном значень, кроком екстраполяції та іншими параметрами.
    b. Інтуїтивно зрозумілий і простий інтерфейс налаштування параметрів для зручності користувача.

4. Управління помилками:
    a. Обробка можливих помилок і винятків, надання зрозумілих та інформативних повідомлень про помилки.
    b. Надання користувачеві можливості виправлення помилок і повторного виконання екстраполяції.

5. Експорт результатів:
    a. Можливість експорту результатів екстраполяції в різні формати для подальшого аналізу та використання (наприклад, CSV, Excel та ін.).

6. Інтерфейс користувача:
    a. Інтуїтивно зрозумілий і дружній до користувача інтерфейс, що

забезпечує зручність роботи з програмою.

b. Зрозумілі інструкції та підказки для використання функцій програми.

2.1.4. Вхідні та вихідні дані

Вхідними даними для програми є масив відомих відповідностей між змінними та значенням функції. Також вхідними даними можна вважати файл з даними. Для коректної роботи потрібно щоб кількість відповідностей не перевищувала кількість змінних, або, якщо даних більше, ніж невідомих, щоб данні були коректні і мали під собою лінійні залежності.

Необов'язкові данні – імена змінних. За замовчування вони іменуються Y, X0, X1, X3 тощо.

Вихідними даними є результат роботи програми – значення функції у відповідній точці. Також вихідними даними можна вважати файл зі збереженими даними.

2.1.5. Діаграма прецедентів

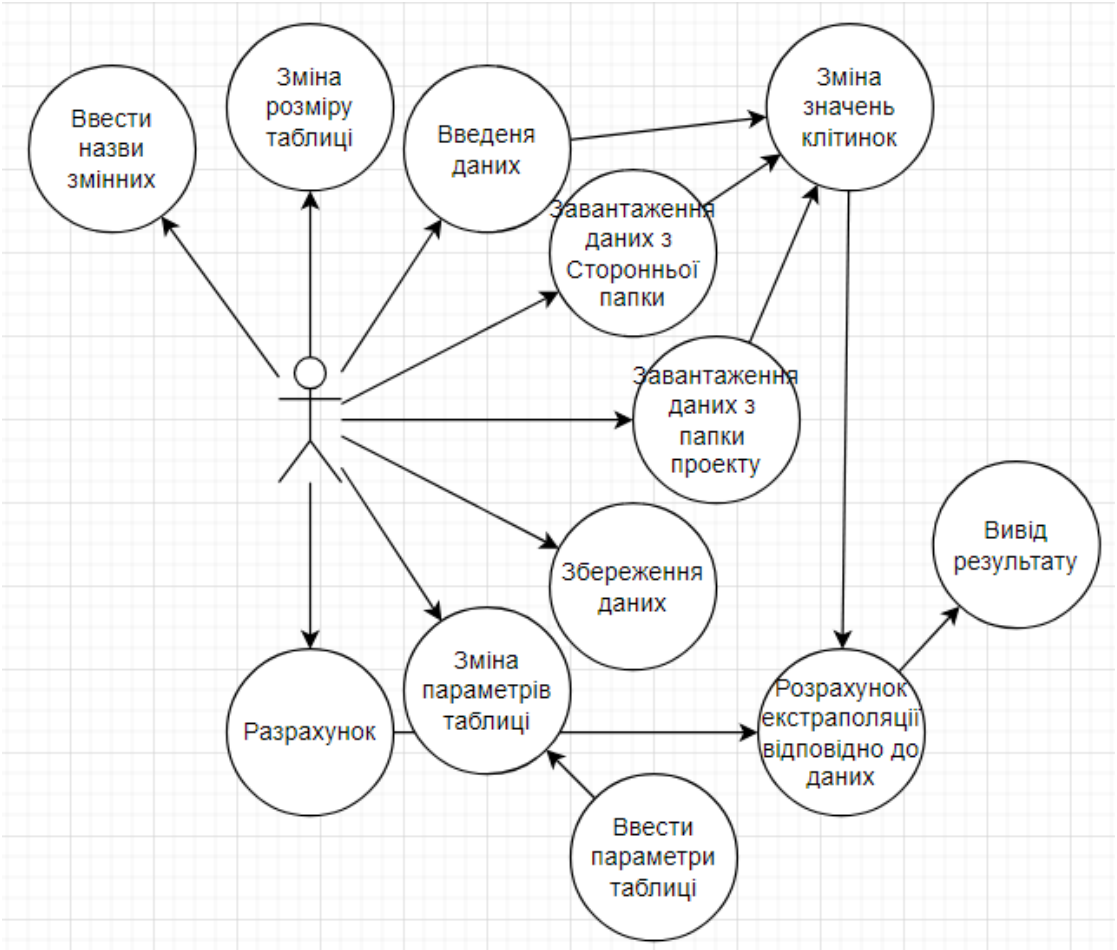

Рисунок 2.1 – діаграма прецедентів

На діаграмі прецедентів зображені основні можливі дії користувача та подальші сценарії, які відкриваються перед ним в ході виконання дій.[21]

2.2. Внутрішнє програмування

Внутрішнє проектування полягає у визначенні складових системи (програми), яка розробляється, їх структури та поведінки. В рамках даного проектування може бути визначено:

– статична архітектура системи: модулі, їх структура, відповідальність та взаємозв’язок;

– поведінка системи: взаємодія модулів системи з уточненням повідомлень, які передаються, життєвого циклу створених об’єктів, станів об’єктів;

– топологія системи (фізичне розміщення системи та її частин);

– структура (логічна схема) баз та сховищ даних;

– особливості інтерфейсу користувача: набір апаратних та програмних засобів, які будуть використовуватися для взаємодії з програмою, сценарій та структура діалогу, візуальне представлення елементів інтерфейсу тощо.

2.2.1. Проектування архітектури системи

Проектування архітектури системи для програми вирішення задач методами багатопараметричної лінійної екстраполяції включає наступні аспекти:

1. Модульна структура: Поділ системи на модулі, які відповідають різні аспекти розв'язання задач екстраполяції. Наприклад, модуль для введення вихідних даних, модуль для алгоритмів екстраполяції, модуль для відображення результатів і т.д. Кожен модуль має відповідати за конкретні функціональні області.

2. Компоненти системи: Визначення компонентів системи, таких як класи, об'єкти та функції, що реалізують функціональність кожного модуля. Наприклад, класи для читання та обробки вхідних даних, класи для реалізації алгоритмів

екстраполяції та класи для графічного відображення результатів.

3. Інтерфейси та взаємодія: Визначення інтерфейсів між компонентами системи для обміну даними та виклику функцій. Наприклад, інтерфейс передачі вхідних даних від модуля введення даних до модуля алгоритмів екстраполяції, інтерфейс передачі результатів екстраполяції від модуля алгоритмів до модуля відображення результатів.

4. Управління даними: Визначення структури даних та способів зберігання та обробки вихідних даних та результатів екстраполяції. Наприклад, використання бази даних або файлової системи для зберігання даних, визначення структур даних для подання багатовимірних значень та результатів екстраполяції.

5. Алгоритми екстраполяції: Реалізація алгоритмів та методів для проведення багатопараметричної лінійної екстраполяції. Це включає вибір та реалізацію відповідних математичних моделей екстраполяції, розробку алгоритмів для обробки та аналізу даних, а також забезпечення точності та надійності результатів екстраполяції.

6. Графічний інтерфейс користувача: Розробка інтерфейсу користувача, який дозволяє користувачам вводити дані, задавати параметри екстраполяції, переглядати і аналізувати результати. Графічний інтерфейс повинен бути інтуїтивно зрозумілим, зручним у використанні та надавати необхідну інформацію та контроль користувача

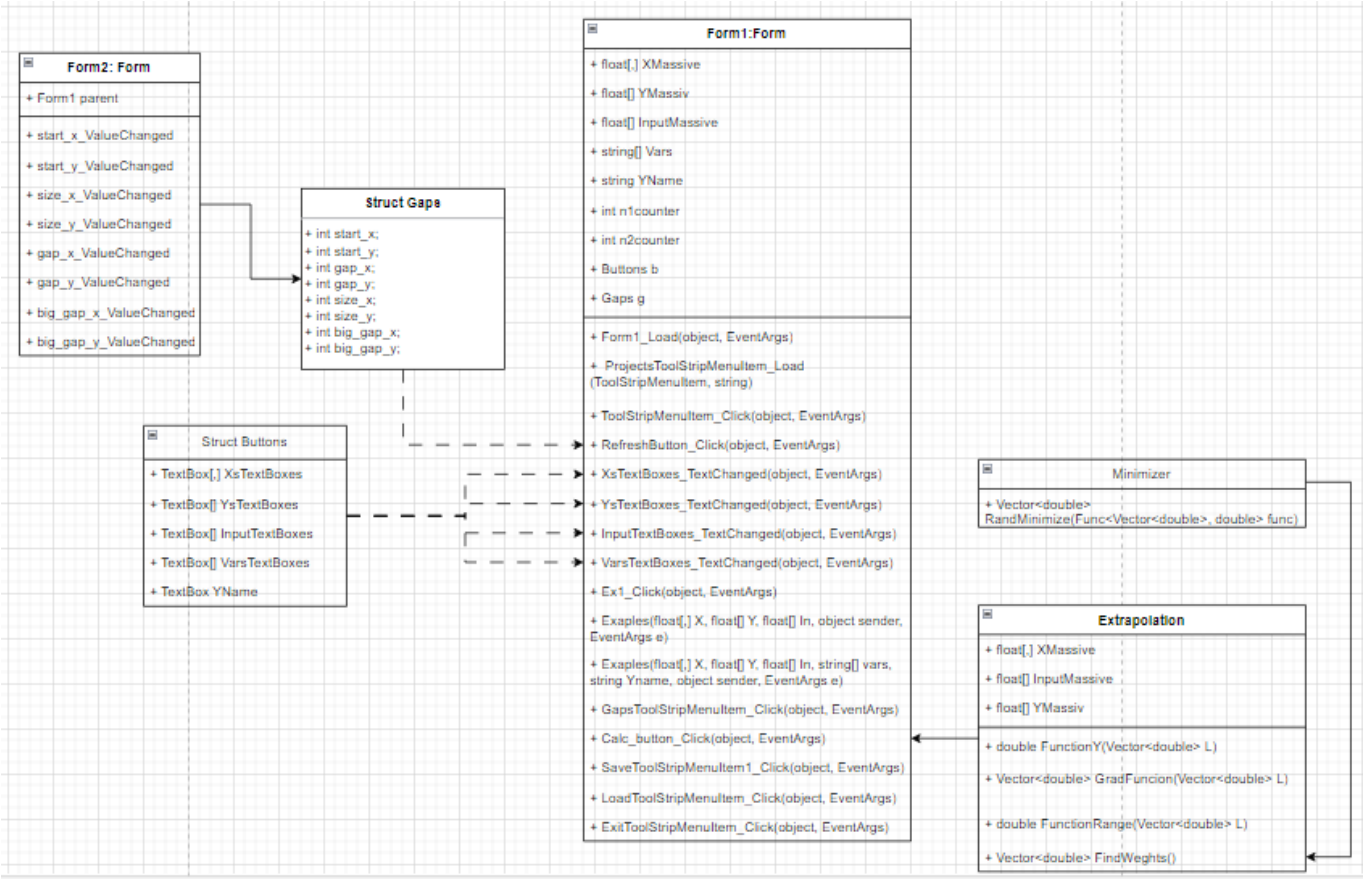

Рисунок 2.2.1 – Діаграма класів

На діаграмі класів показано взаємодію між класами та структурами програми. Form1 являє собою основний клас, який викликає, використовує та об’єднує інші сутності програми

2.2.2. Проектування інтерфейсу користувача

Інтерфейс користувача можна поділити на смисловий та візуальний. Смисловий складається з діалогів між користувачем та комп’ютером і спрямований на вирішення проблеми.

Кожен діалог складається з окремих процесів введення-виведення, які фізично за допомогою монітору, клавіатури, миші і інших маніпуляторів забезпечують зв'язок користувача і комп'ютера. В процесі різних діалогів система перебуває в різних станах. Ці стани відображені на «Рисунок 2.2 – Діаграма станів».

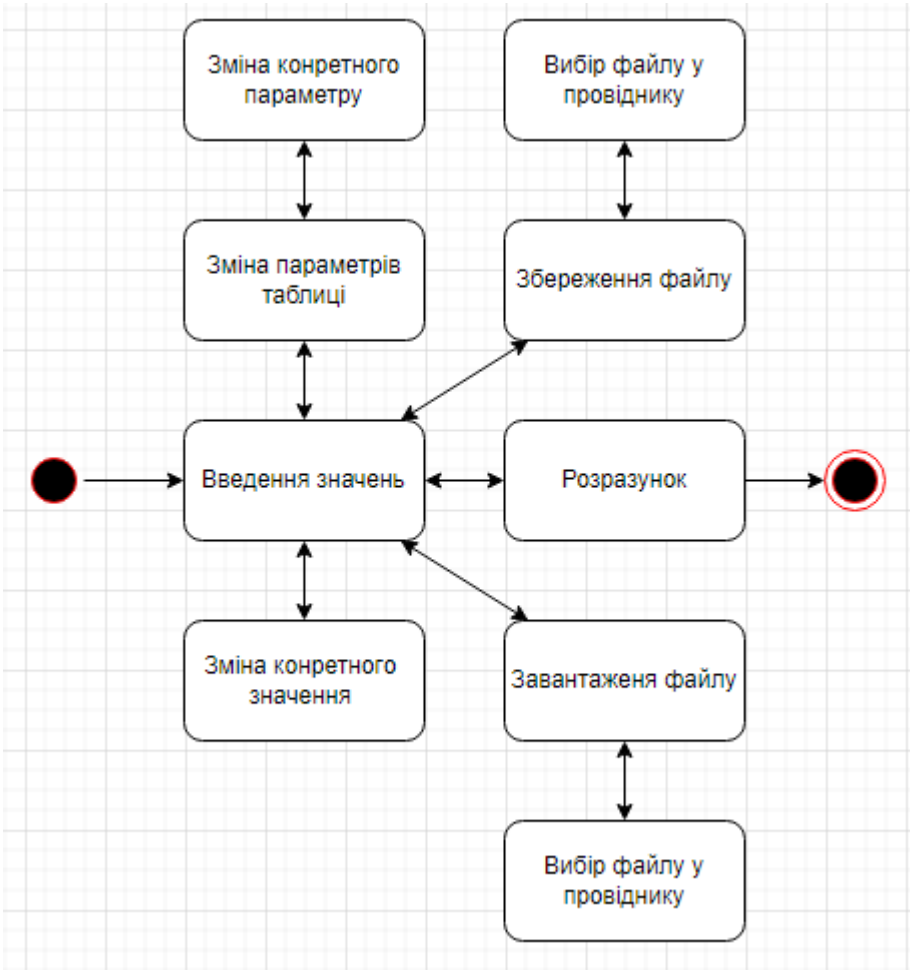

Рисунок 2.2 – Діаграма станів

Крім діалогів ще є візуальна складова інтерфейсу - Графічний інтерфейс

Графічний інтерфейс користувача (ГІК, англ. GUI, Graphical user interface) — тип інтерфейсу, який дає змогу користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.[3]

Для розробки графічного інтерфейсу використовують ескізи інтерфейсу.

Ескізи інтерфейсу (Interface sketches) є рудиментарними або грубими нарисами графічного користувальницького інтерфейсу (GUI) програми або веб-сайту. Вони є початковим етапом процесу проектування інтерфейсу і допомагають візуалізувати загальну структуру та компоненти інтерфейсу.

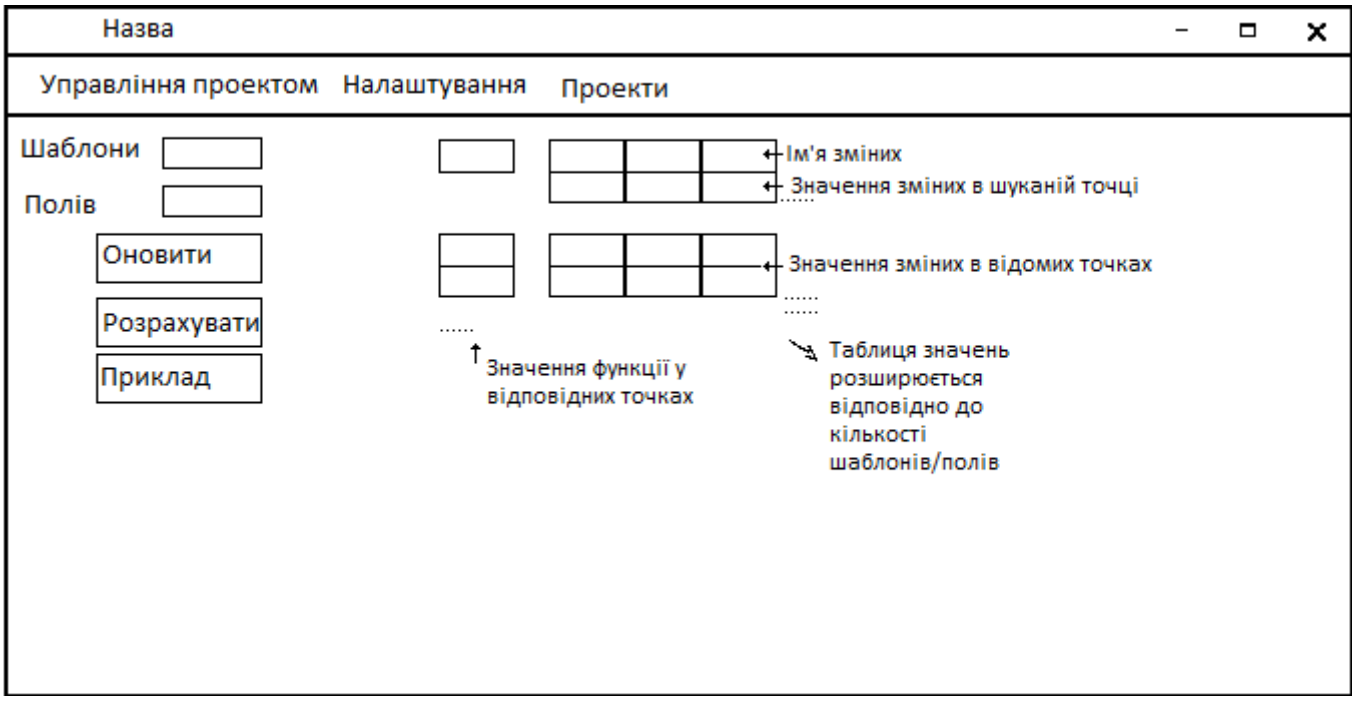

Рисунок 2.3 – Ескіз інтерфейсу

2.2.3. Проектування бази даних

Бази даних, як такої, програма не використовує з наступних причин

- для використання програми треба було б встановлювати відповідну БД на комп’ютер користувача
- функціонал програми не потребує постійної роботи з великою кількістю даних і постійно їх оновлювати

Замість Бази Даних в нашому проекті використовується файлова система збереження даних. Дані з програми форматуються у JSON формат та записуються у файл. При необхідності ці файли можна завантажити і продовжити роботу з проектом. Якщо Файли зберігати у папці з проектами, то їх можна швидше відкрити через вкладку «Проекти», як показано на Рисунку 2.4 «відкриття проектів»

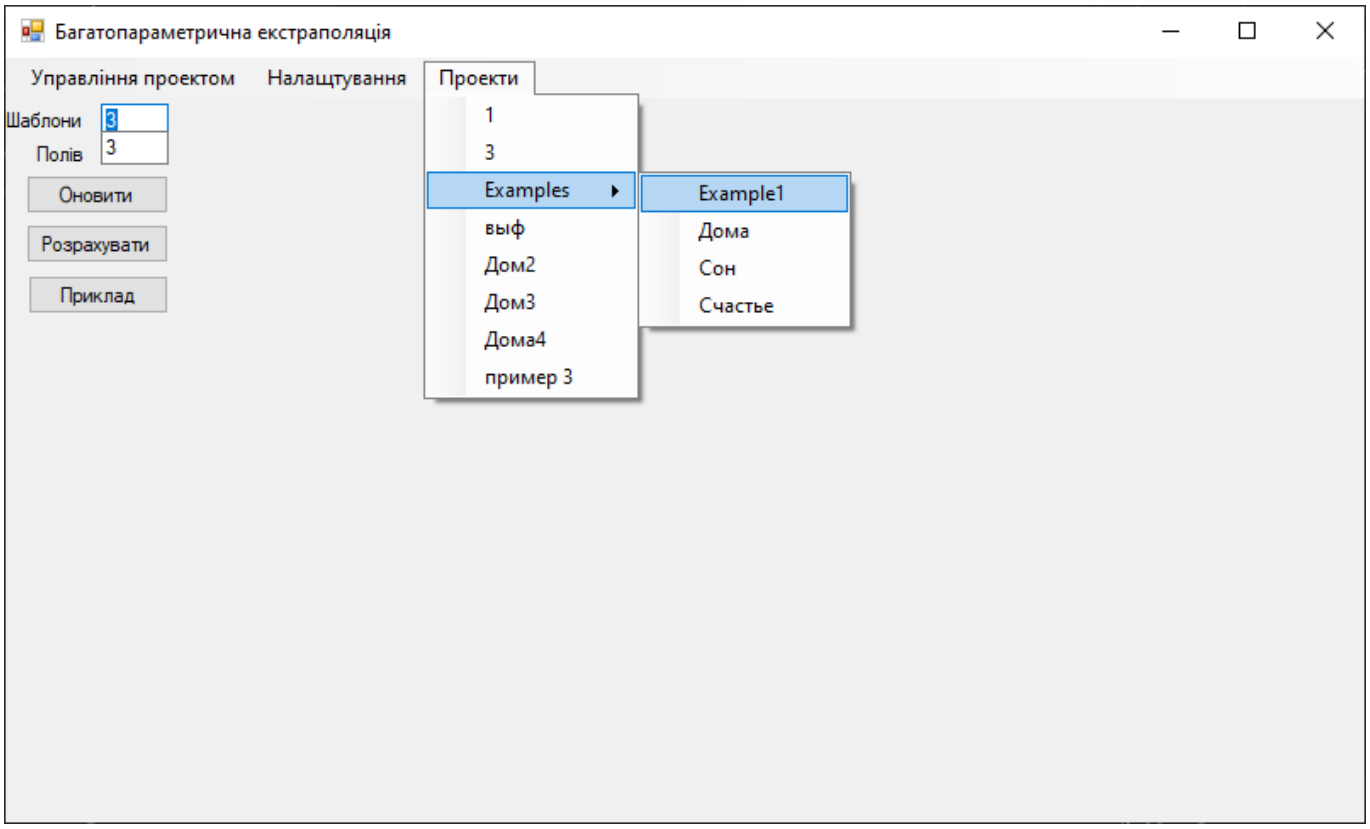

Рисунок 2.4 – відкриття проектів

Висновки

В цьому розділі ми спроектували зовнішню та внутрішню архітектуру програми, створили діаграми прецедентів, класів та станів, накреслили ескіз графічного інтерфейсу та описали роботу системи збереження файлів.

3. РОЗРОБКА ПРОГРАМИ

В даному розділі проводиться опис роботи по розробці алгоритмів та інтерфейсу програми. Для розробки була обрана мова програмування C#[23] з використанням Windows Froms для зручного налагодження інтефейсу.[22]

3.1. Алгоритми

В програмі два основних алгоритми: головний – алгоритм лінійної екстраполяції та допоміжний – алгоритм випадкового пошуку мінімуму заданої функції. Пошук мінімуму потрібен для коректної роботи екстраполяційного алгоритму.

3.1.1. Алгоритм лінійної екстраполяції

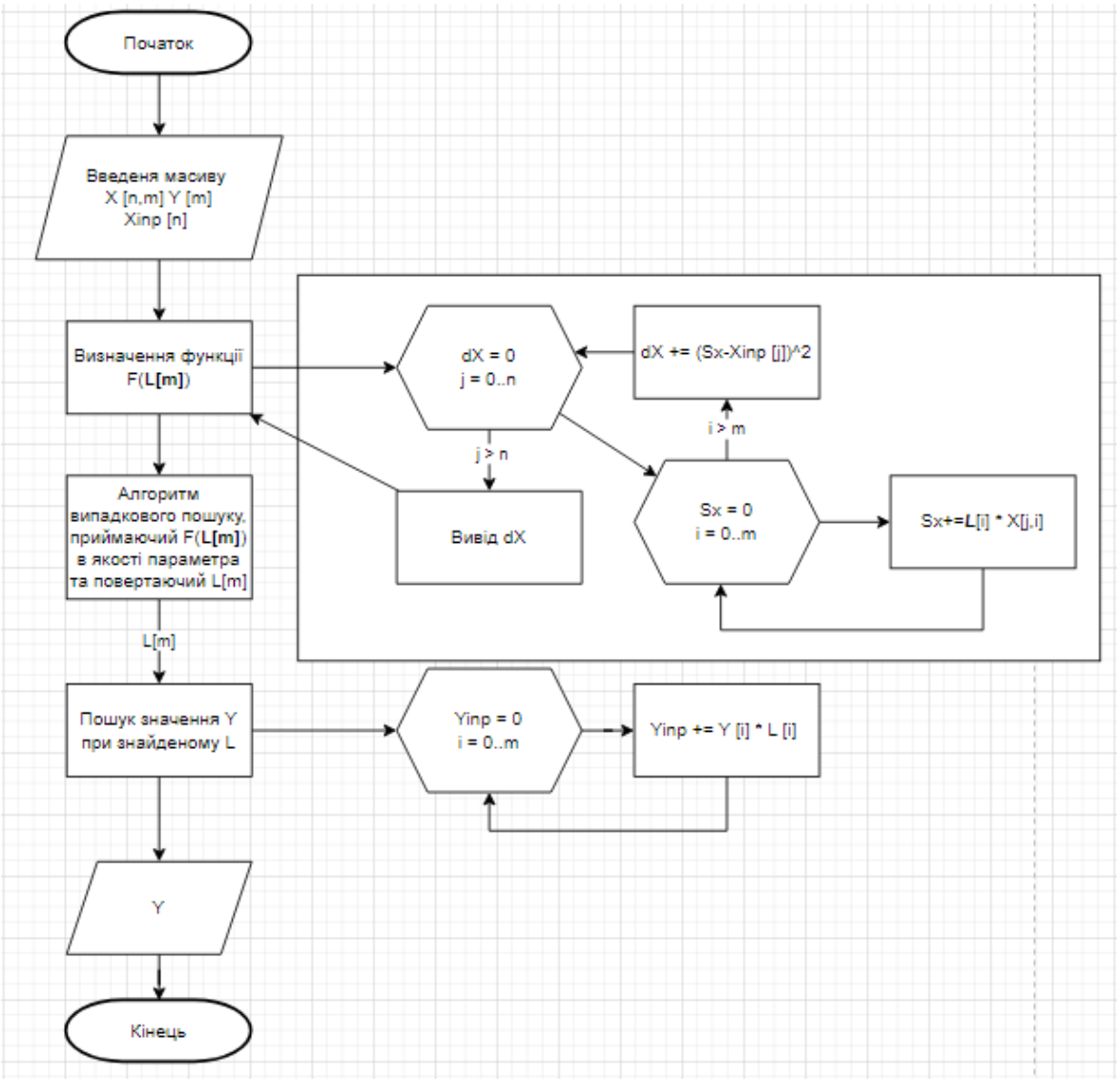

Рисунок 3.1.1.1 – Блок-схема алгоритму БЛЕ

3.1.2. Алгоритм пошуку мінімуму

Існує кілька методів для пошуку мінімуму чи максимуму багатопараметричної функції. Деякі з них включають:

1. Метод градієнтного спуску (Gradient Descent): Цей метод ґрунтується на ітеративному оновленні параметрів функції у напрямку найбільш крутого зменшення градієнта. Він ітеративно оновлює параметри до мінімуму функції.

2. Метод найшвидшого спуску (Steepest Descent): Цей метод використовує напрям, протилежний градієнту функції, і ітеративно оновлює параметри в цьому напрямку до досягнення мінімуму функції.

3. Метод Ньютона (Newton's Method): Цей метод використовує другі похідні функції (гесіан) знаходження оптимального кроку кожної ітерації. Він сходиться швидше, ніж градієнтний спуск, але потребує обчислення та обігу гесіана функції.

4. Метод Бройден-Флетчер-Гольдфарб-Шанно (BFGS): Цей метод комбінує ідеї методу Ньютона і градієнтного спуску. Він наближає гесіан функції ітеративно, використовуючи інформацію про градієнти.

5. Метод сполучених градієнтів (Conjugate Gradient): Цей метод використовується для оптимізації без обмежень і ґрунтується на напрямках сполучених градієнтів. Він ітеративно оновлює параметри функції у цих напрямках до досягнення мінімуму функції.

6. Генетичні алгоритми (Genetic Algorithms): Це еволюційний підхід до оптимізації, що ґрунтується на принципах природного відбору та генетичного

схрещування. Генетичні алгоритми можуть використовуватись для пошуку оптимальних параметрів функції у просторі.

Був обраний випадковий алгоритм. Випадковий пошук - це метод оптимізації, що ґрунтується на випадковому виборі параметрів функції та оцінці їх якості. Ось переваги випадкового пошуку:

1. Глобальний пошук: Випадковий пошук має властивість глобального пошуку, що означає, що він може досліджувати велику кількість параметрів функції і знаходити глобальні оптимуми. Це особливо корисно у випадках, коли функція має багато локальних оптимумів.

2. Уникнення застрягання в локальних оптимумах: Оскільки випадковий пошук ґрунтується на випадковому виборі параметрів функції, він може уникнути застрягання в локальних оптимумах. На відміну від інших методів оптимізації, які можуть бути вразливими до локальних оптимумів, випадковий пошук може "вискочити" з локального оптимуму та продовжити дослідження простору параметрів.

3. Незалежність від градієнтів та гесіанів: Випадковий пошук не вимагає обчислення градієнтів або гесіанів функції, тому він може бути використаний для оптимізації функцій, для яких складно чи неможливо отримати аналітичні градієнти.

4. Застосування до великих просторів параметрів: Випадковий пошук добре масштабується для роботи з великими просторами параметрів. Оскільки він не вимагає обчислення градієнтів або іншої додаткової інформації про функцію, він може ефективно дослідити велику кількість параметрів.

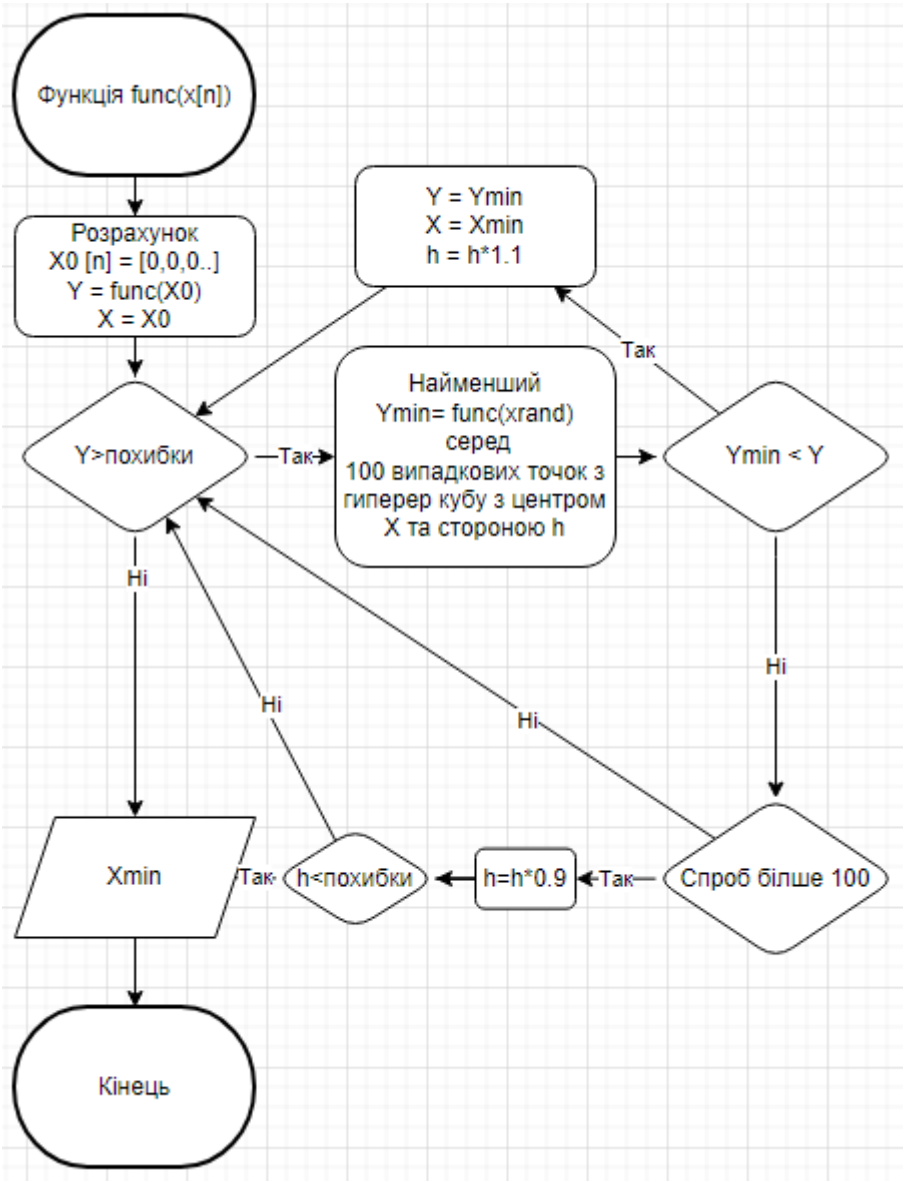

Рисунок 3.1.2.1 – Блок-схема випадкового пошуку мінімуму

Вище представлена блок-схема представляє роботу випадкового пошуку мінімуму функції. На вхід приймає делагат функції, яка приймає динамічний масив double та повертає значення функції в цій точці. Сам алгоритм повертає динамічний масив того ж розміру, що і вхідний параметр у вхідній функції. Значення вихідної точки найменше для цієї функції.

Алгоритм спроектован для роботи з функцією похибки, яка не приймає від'ємних значень (як в методу найменших квадратів)

3.2. Розробка інтерфейсу

Графічний інтерфейс програми розроблявся на основі C# Windows Forms. [22][23]

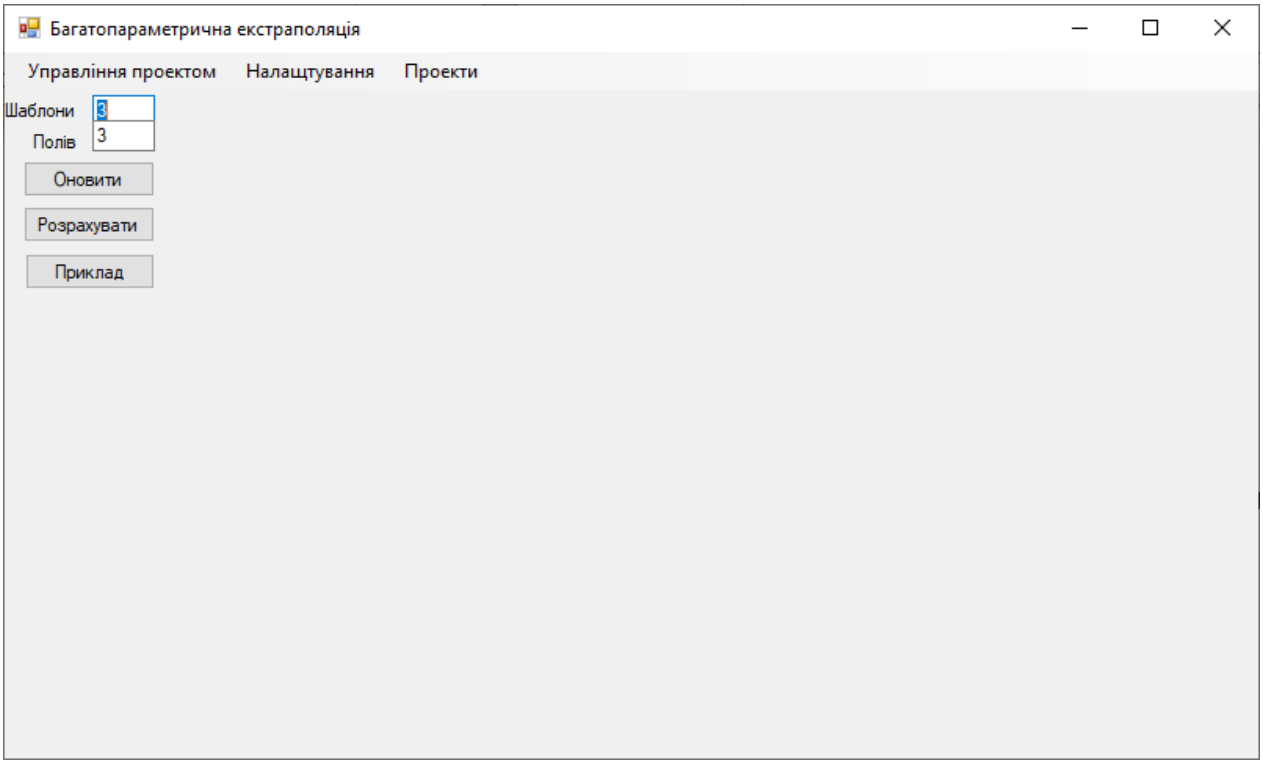

Рисунок 3.2.1 – Графічний інтерфейс після запуску

Складові графічного інтерфейсу:

1. Статичні текстові поля, кнопки та написи
   1.1. Шаблони – поле, в яке вказується кількість рядків
   1.2. Полів – поле, в яке вказується кількість змінних
   1.3. Оновити – кнопка, яка змінює кількість динамічних текстових полів відносно значень «Шаблони» та «Полів»
   1.4. Розрахувати – кнопка, яка запускає процес екстраполяції зі значеннями з динамічних текстових полів. В результаті створює написи «Результат» та текстове значення результату екстраполяції з вказівкою похибки, якщо вона значна
   1.5. Приклад – кнопка, яка заповню поля тестовими даними 5/5.
2. Динамічні текстові поля, які створюються в залежності від введеної кількості шаблонів та полів
3. Стрічка меню яка містить:
   3.1. Управління проектом:
      3.1.1. Завантаження проекту – завантажує проект з файлу JSON – заповнює

відповідні статичні та динамічні поля

3.1.2. Збереження проекту – зберігає данні з статичних та динамічних текстових полів у JSON-файл

3.2.Налаштування – Таблиці: відкриває форму налаштування таблиці:

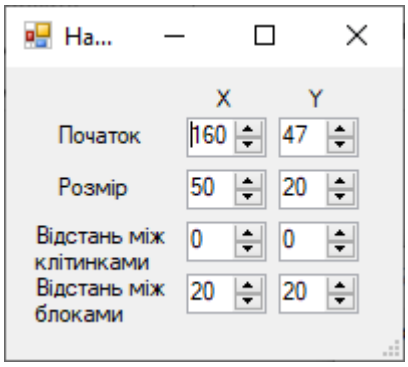

Рисунок 3.2.2 – Вікно «Налаштування таблиці»

3.2.1. Початок (X,Y) – встановлює відстань таблиці від верхнього лівого краю

3.2.2. Розмір (X,Y) – встановлює розмір клітинок з даними

3.2.3. Відстань між клітинками (X,Y) - встановлює відстань клітинок з даними

3.2.4. Відстань між блоками (X,Y) встановлює відстань між блоками X Y та X`

3.3.Проекти – динамічний список папок та JSON-файлів, які знаходяться у папці проекту.

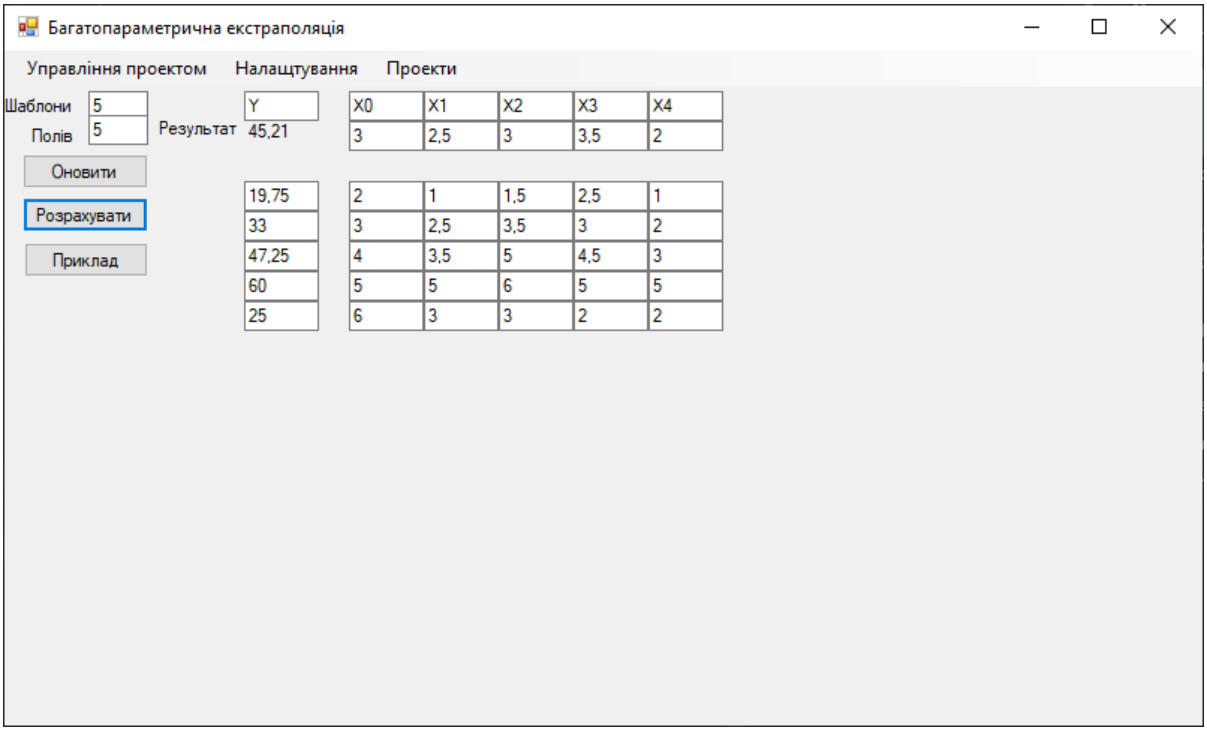

Рисунок 3.2.3 – Програма в процесі роботи

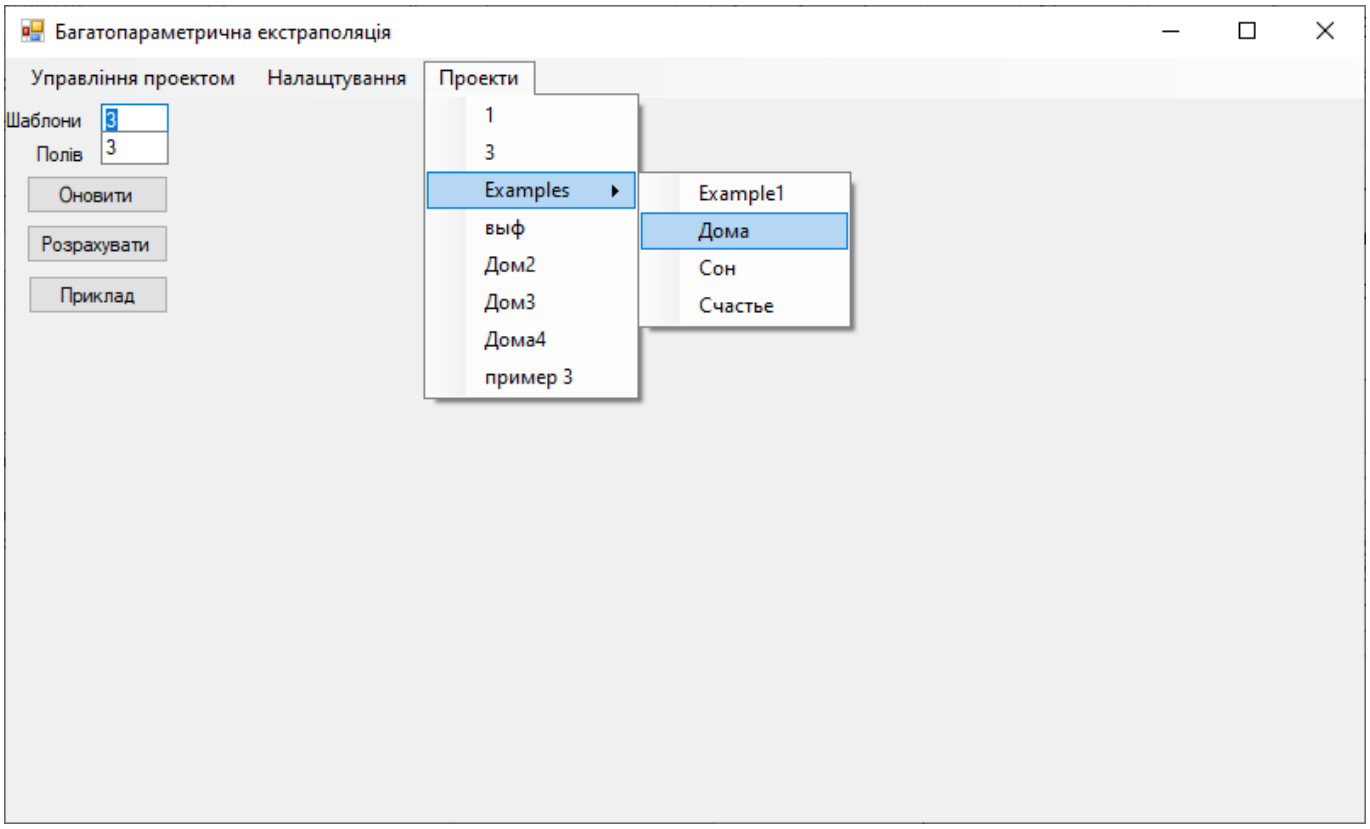

Рисунок 3.2.4 – відкриття проектів з панелі швидкого доступу

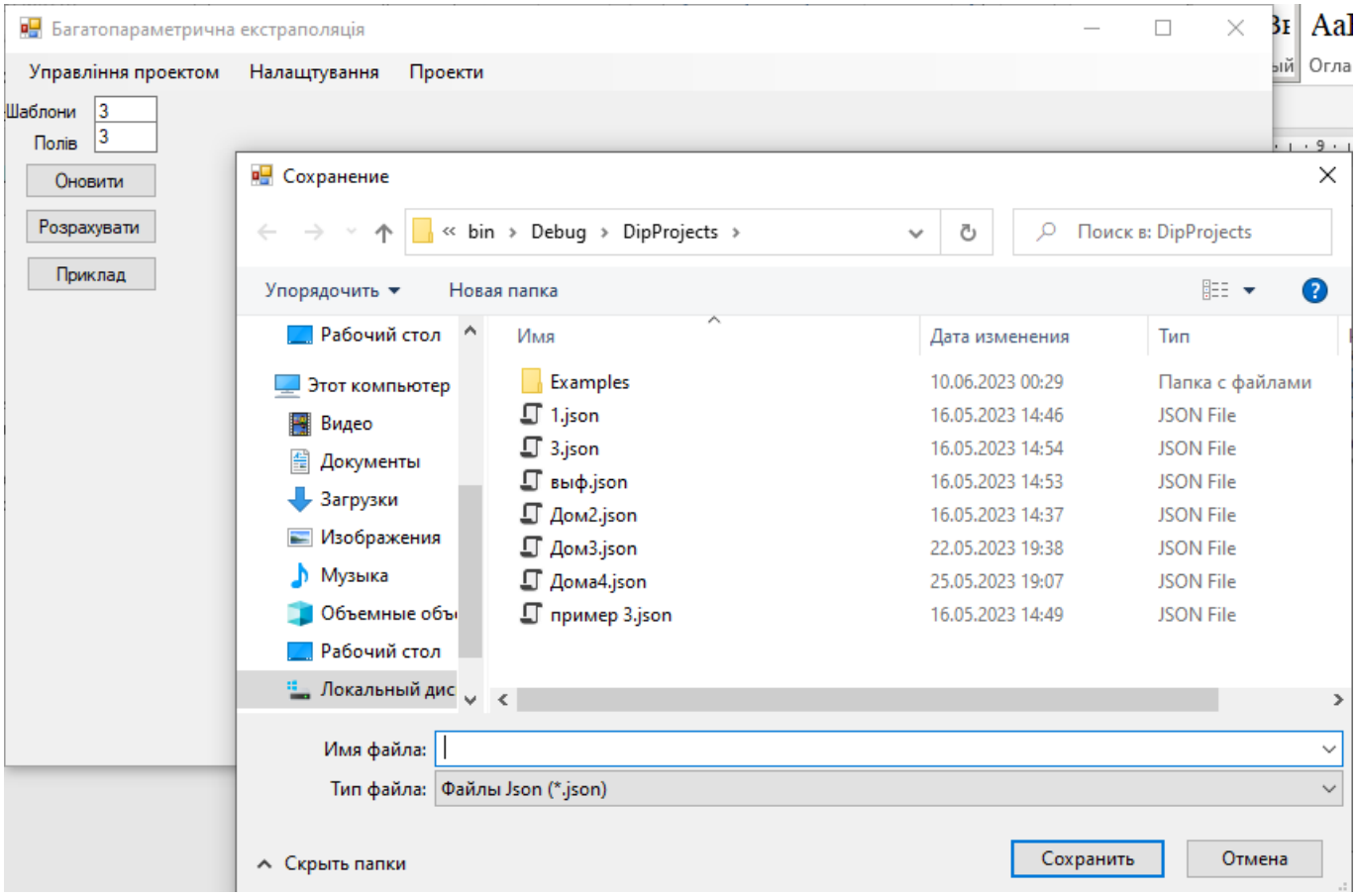

Рисунок 3.2.5 – Відкриття/збереження файлу через «управління проектом»

Висновки

На стадії розробки було розроблено алгоритми, графічний інтерфейс, була обрана мова проектування, написана та налагоджена програма та підготовлена до подальшого тестування.

4. ТЕСТУВАННЯ ТА НАЛАГОДЖЕННЯ

Тестування ми проводимо за допомогою UNinitTest(Юнит-тестів) [24][23]

4.1. Тестування функцій визначення значення та похибки в даній точці

Функції визначення значення та похибки в даній точці являють собою алгоритмізовані математичні розрахунки, а отже основі моменти, в яких можуть виникнути питання це:

1. Точність обчислень: При додаванні великої кількості чисел з плаваючою комою може виникнути втрата точності. Таким чином, слід перевірити, що функція обробляє великі значення та підсумовує їх із високою точністю.

2. Обробка особливих значень: Функція має коректно обробляти спеціальні значення, такі як NaN (Not-a-Number), Infinity та Negative Infinity. Перевірте, чи функція правильно обробляє ці значення і повертає очікуваний результат.

3. Граничні випадки: Перевірте, як функція обробляє граничні значення, наприклад підсумовування порожнього списку або підсумовування списку з одним елементом. Переконайтеся, що результати відповідають очікуванням.

4. Обробка помилок: Якщо функція може генерувати винятки при некоректних вхідних даних, переконайтеся, що вона обробляє такі ситуації та повертає коректні винятки чи помилки.

5. Продуктивність: Якщо функція призначена для роботи з великими обсягами даних, перевірте її продуктивність та швидкість виконання. Переконайтеся, що функція виконується ефективно і не витрачає надто багато часу на обчислення.

6. Перевірка типу, що повертається: Переконайтеся, що функція повертає очікуваний тип даних (наприклад, double) і не виникають проблеми з

перетворенням типів або втратою даних.

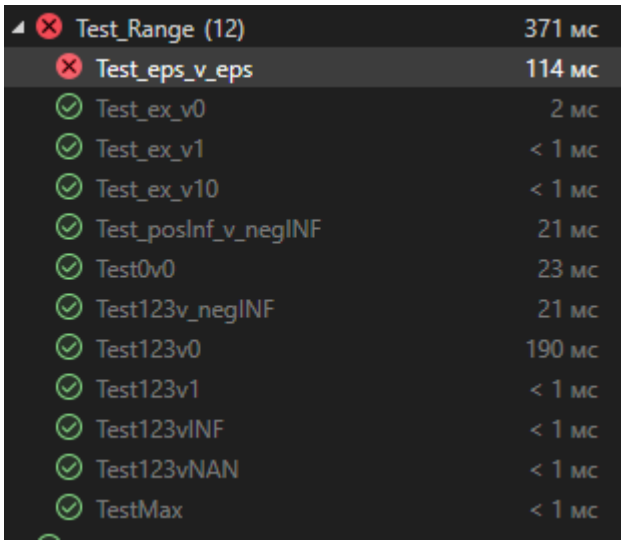

Рисунок 4.1.1 – тестування функції похибки

Результати тестів функції похибки: робота на звичайних значеннях нормальна. Робота на граничних значеннях помилку не видає. Але робота в масштабах похибки double випадає в нуль та видає неочікуваний результат.

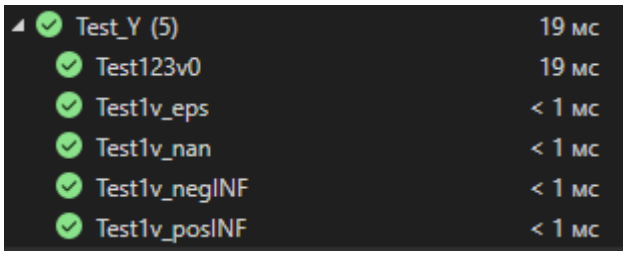

Рисунок 4.1.2 – тестування функції значення

Функції працює стабільно при всіх «небезпечних» даних

4.2. Тестування функції екстраполяції

Функція екстраполяції приймає 3 параметри: двомірний масив X, масив Y та масив X у шуканій точці та видає значення Y у цій точці. Для тестування був використаний наступний метод:

1. Формується n розмірна лінійна функція з випадковими значеннями параметрів.
2. Для m кількості n-розмірних масивів випадкових X знаходиться Y за вище

визначеною функцією. Також знаходиться значення функції для шуканої точки X, також з випадкових значень.

3. Знаходиться значення методом екстраполяції для шуканої точки з масивів.
4. Порівнюються знайдене значення з заздалегідь вирахуваним. Так як наша функція створена для роботи з реальними даними, які генеруються не функціями, а отже мають похибки, то і сама функція видає значення з похибками. Вдалим результатом при тестуванні вважається точність менше встановленої похибки.

Маємо три випадки відносно n(кількість змінних)\m(кількість рівнянь):

1. n<m найгірший випадок для реальної ситуації, бо при значних похибках в вхідних даних формується безліч точок, в які може зійтись функція, а отже при повторних запусках алгоритму точки буду кожен раз різні. В разі ж точних даних навпаки є найкращим випадком, бо алгоритм швидко сходиться до шуканої точки та завжди знаходить точку з малою похибкою. З рисунку бачимо, що великий час буває в тих випадках, коли розмірність велика, а кількість змінних наближається до кількості рівнянь.

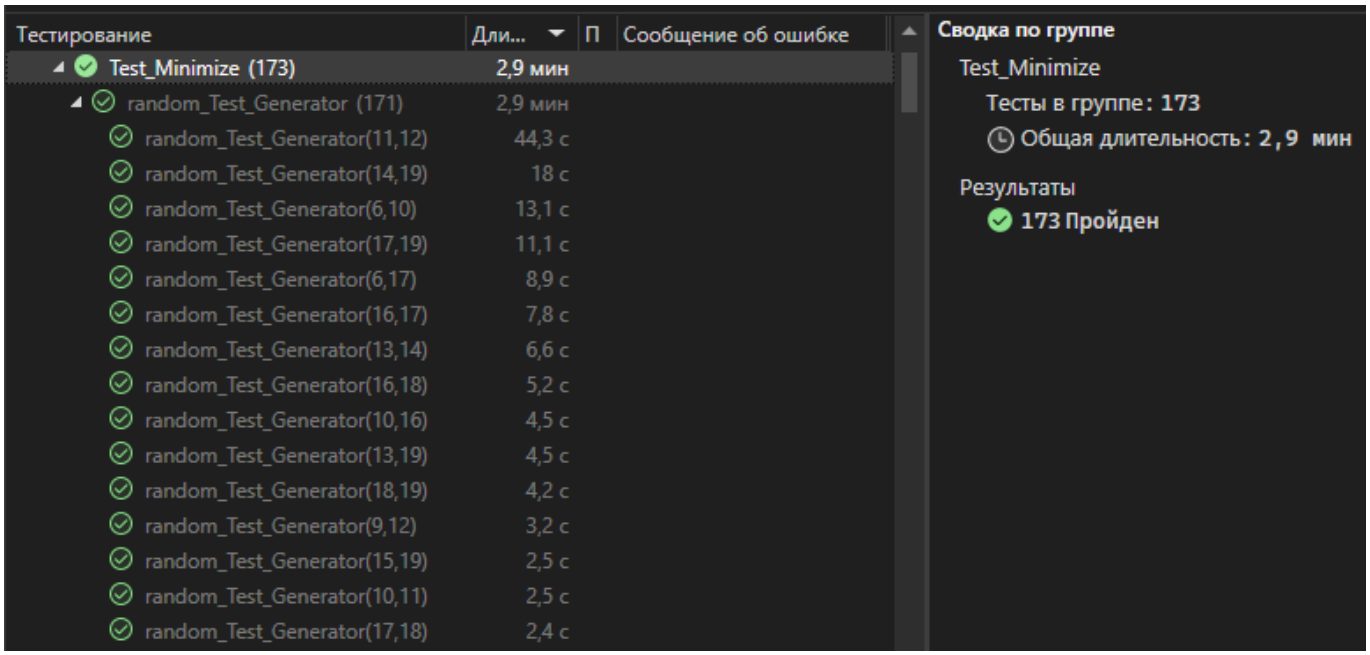

Рисунок 4.2.1 – тести для

2. n>m є цільовим випадком, на який заточена програма. В незалежності від роду даних точка буде знайдена з великою точністю (наскільки дозволяють дані) та

за невеликий час. З тестів видно, що деякі випадки видають помилку. Але помилка в тому, що похибка надто велика. Середня величина похибки пропорційна різниці n та m, як і швидкість виконання. Як бачимо на рисунку 4.2.2.а навіть у досить важких випадках похибка складає 20%.

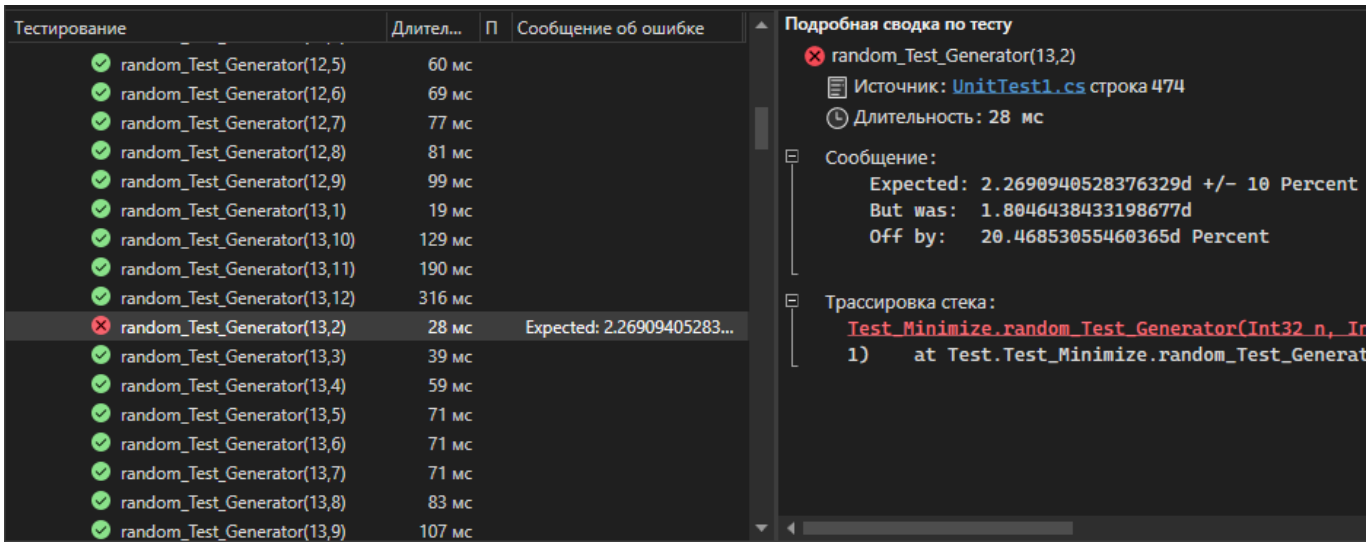

Рисунок 4.2.2.а – приклад похибки при малій кількості даних

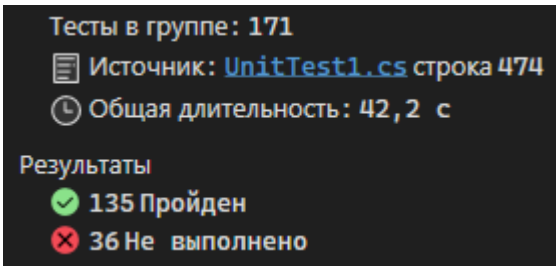

Рисунок 4.2.2.б – загальна статистика тестів з малими даними

3. n=m «квадратний» випадок, в якому час найбільший. В випадку точних даних точніше було б скористатися аналітичними методами.

   На рисунку 4.2.3 показано результати тестування для кожного квадрату. Для кожного розміру проводилось по 3 тести. З результатів тестування можна зробити висновок, що алгоритм завжди знаходить результати, але в деяких випадках це потребую значно більше часу.

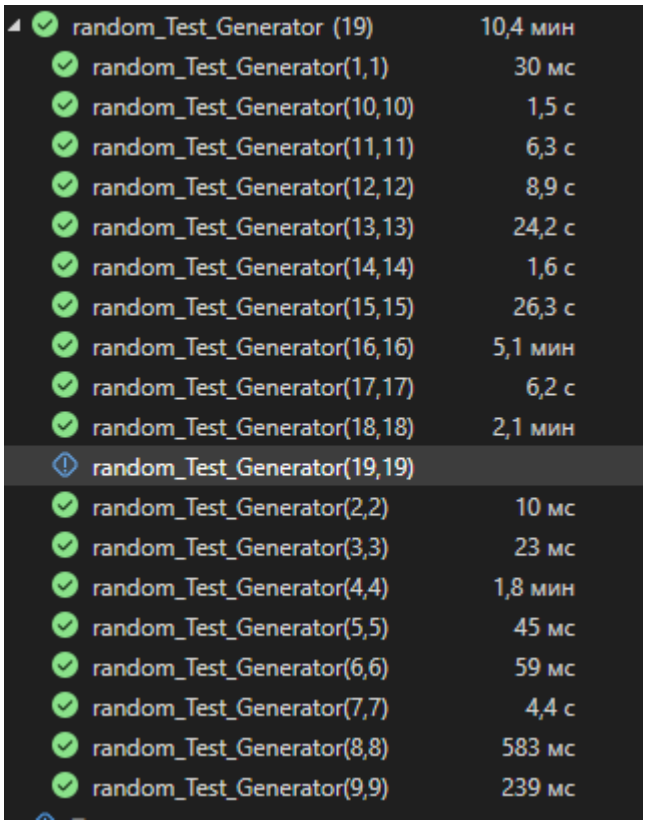

Рисунок 4.2.3 – результати тестів квадратних випадків випадковою генерацію

Висновки

В результаті проведення тестів виявлено, що програма працює як очікувалось, точність та швидкість виконання залежать від повноти та розміру вхідної інформації. При великій кількості даних збільшується точність результатів, але і час виконання.

5. АНАЛІЗ ТА ВИСНОВКИ

Дипломна робота присвячена завданням щодо розробки багатопараметричних екстраполяційних моделей проектування складних систем, їх удосконалених алгоритмів реалізації та створенню відповідного програмного забезпечення. В роботі досліджені і використані можливостями застосування моделі багатопараметричної лінійної екстраполяції як обчислювальної процедури із вибору параметрів при проектуванні складних систем і технологій на основі даних аналогів і прототипів. В них були враховані особливості об'єктів проектування щодо вимог до параметрів (області значень, неперервні/дискретні), а також створені процедури, призначені для розширення можливості застосування моделі БЛЕ для реалізації зворотних завдань, які виникають при формуванні наборів параметрів систем то проектуються.

Завданнями роботи складали питання щодо розвитку постановок завдань, розробки удосконаленого методу БЛЕ для завдань з суттєво нерівними величинами областей варіювання параметрів, формуванні нової процедури методу БЛЕ, призначеної для реалізації зворотних завдань вибору та проектування, створення програмного забезпечення щодо їх автоматизації. Для досягнення мети були використані методи математичного моделювання, лінійної екстраполяції, багатопараметричної оптимізації на основі випадкового пошуку, методи проектування та розробки програмного забезпечення, як засобу автоматизації для завдань застосування БЛЕ.

В результаті виконання дипломної роботи було розроблено удосконалений методу багатопараметричної лінійної екстраполяції, сформована нова процедура щодо реалізації зворотних завдань вибору та проектування, створене програмне забезпечення з автоматизації відповідних завдань процесів вибору та проектування на основі даних про аналоги та прототипи створюваних систем.

Наукова та технологічна новизна роботи визначається наступним: було удосконалена процедура багатовимірної лінійної екстраполяції при суттєво нерівних величинах параметрів моделей, розроблена нова процедура застосування методу

багатовимірної лінійної екстраполяції із реалізації зворотних завдань вибору та проектування, що відрізняється структурами шаблонів даних та забезпечує розрахунок оцінок параметрів моделі (типу підбору параметру), сформовані математичні моделі та реалізовані нові завдання вибору та проектування на основі БЛЕ.

Дипломна робота складає 99 с., 22 рис.,2 табл., 4 додатків.,242 джерел.

Значні можливості БЛЕ обумовлюють його широке прикладне значення, цей метод має велику апробації і показав досить високу ефективність на етапах попереднього аналізу. Аналіз не дозволив виявити загально відомих та прийнятих і доступних програмних засобів реалізації завдань БЛЕ. На основі наведено можна розглядати розроблені програмні засоби з застосування БЛЕ в якості інструментарію попереднього аналізу, вибору та формування наборів параметрів систем, які проектуються, на основі наборів відомих аналогів та прототипів. Ц цілому результати дипломної роботи сприяють підвищенню ефективності процесів проектування та аналізу складних систем, за умов існування аналогів або прототипів об’єктів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Растригін Л.А. «Екстраполяційні методи проектування та управління». М. Машинобудування, 1986 – 120 с.
2. uk.wikipedia.org/wiki/Графічний_інтерфейс_користувача
3. Пономарьов Ю.П., Агладзе В.А., Гавицріна Л.В. Розв’язання задач методом багатовимірної екстраполяції. – У кн. Адаптація у обчислювальних системах. Рига: Зінатне, 1978. С. 129 - 138.
4. Агладзе В.А., Пономарьов Ю.П. Рекурентна форма методу багатовимірної екстраполяції. У кн. Машинні способи виявлення закономірностей Рига: Ризький пілітехн. ін-т, 1981, с. 73 - 76.
5. Растригін Л.А. Адаптація складних систем Рига: Ризький політехн. ін-т, 1981, с. 68 - 73.
6. Растригін Л.А., Самсонс Д.В. Метод багатовимірної лінійної екстраполяції у задачах виявлення динамічних закономірностей. У кн. Машинні способи виявлення закономірностей Рига: Ризький пілітехн. ін-т, 1981, с. 68 - 73.
7. Растригін Л.А., Ейдук Я.Ю. Пошукові алгоритми визначення множини Парето. – У кн. Адаптація у обчислювальних системах Рига: Зінатне, 1978. С. 69 - 76.
8. Дексніс Г.К., Растригін Л.А. Синтез алгоритмів пошукової оптимізації методом багатовимірної лінійної екстраполяції. У кн: Проблеми випадкового пошуку Рига: Зінатне, 1978. С. 184 - 197.
9. Заруцький В.А., Почтман Ю.М., Скалозуб В. В. Оптимізація підкріплених циліндричних оболонок Київ Рига: Вища школа, 1990. 138 с.
10. http://pidruchniki.com/1577111551903/marketing/swot-analiz - «Сутність і значення SWOT-аналізу»
11. http://osvita.ua/vnz/reports/management/15382/ - «SWOT-аналіз. Сутність та сфера застосування».

12. Пилипенко О. В. Стратегічний аналіз: навч. посіб. для студ. вищ. навч. закл./ О. В. Пилипенко За заг. ред. М. І. Коваля. — К.: ДП “Вид. дім “Персонал”, 2018. — 350 с. —Бібліогр.: с. 347 — 350.
13. Extrapolation tips and tricks docs.scipy.org/doc/scipy/tutorial/interpolate/extrapolation_examples
14. Мовчан А.П. Навчальний посібник: Методи статичної оптимізації. Навч. посіб. /Мовчан А.П., Степанець О.В. — К.: НТУУ «КПІ», 2012. — 138 с.
15. Ткаченко О., Ткаченко К. Огляд сучасних систем управління ІТ-проектами. Цифрова платформа: інформаційні технології в соціокультурній сфері. 2019. Т. 2. № 1. С. 27–40. Мережеве планування список джерел
16. Юрчук Н. П. Система моніторингу в управлінні ІТ-проектами. Ефективна економіка. 2018. № 4. URL: http://www.economy.nayka.com.ua/?op=1&z=6248
17. Чайковська М. П. Комплексний підхід моделювання в управлінні ІТ-проектами. Економічний вісник Національного технічного університету України «Київський політехнічний інститут». 2014. № 11. С. 590–596. URL: http://nbuv.gov.ua/UJRN/evntukpi_2014_11_90
18. Інформаційно-методичне забезпечення оцінки та прогнозування ресурсно-функціональних параметрів біофізичних об'єктів © А.А. Барзов, А.В. Пролетарський, В.А. Пролетарська 2019
19. «Проблема вибору метода прогнозування результатів інвестиційного проектування» А.Ф. Шоріков, О.В.Буценко 2006
20. «Застосування математичних моделей у клінічній практиці» Карякіна О.Є., Добродєєва Л. До., Мартинова Н. А., Красильников С. В., Карякіна Т. І. Том 19, № 7 (2012) с. 55-64
21. «UML. Основи. 3-тє видання» Мартин Фаулер, Кендалл Скотт
22. Вивчаємо WINUI 3.0. Ешкрафт Е. Символ-Плюс 2018
23. CLR via C# (Developer Reference) Jeffrey Richter Microsoft Press 2012
24. Мистецтво автономного тестування з прикладами на C#, Рой Ошероув ДМК Пресс 2014**Ошибка! Закладка не определена.**

ДОДАТКИ

Додаток А – Технічне завдання

ЗАТВЕРДЖЕНО

1116130.01304-01-ЛЗ

БАГАТОПАРАМЕТРИЧНІ ЕКСТРАПОЛЯЦІЙНІ МОДЕЛІ ТА АЛГОРИТМИ ВИБОРУ ТА ПРОЕКТУВАННЯ

Технічне завдання

1116130.01304-01

Листів 11

2023

Зміст

1. ВВЕДЕННЯ

Програмне забезпечення для роботи з багатопараметричними лінійними екстраполяціями призначений для поліпшення аналізу даних та виявленню закономірностей в сукупності даних, для яких незручно використовувати аналітичні методи.

Причиною виникнення продукту є відсутність програм багатовимірної екстраполяції та потенційна необхідність в самих.

Область застосування – будь які дослідження, де може бути лінійні закономірності з малою кількістю даних

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є наказ від 07.12.22 №1209ст ректора Українського державного університету науки і технологій “Про призначення наукових керівників та затвердження тем бакалаврських робіт” за спеціальністю 121 “Інженерія програмного забезпечення» факультету “Комп’ютерних технологій і систем” по кафедрі “Комп’ютерні інформаційні технології”.

Тема дипломної роботи –«Багатопараметричні екстраполяційні моделі та алгоритми вибору та проектування». Керівник – Скалозуб В.В.

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Функціональне призначення

Функціональне призначення є забезпечення зручного та ефективного вирішення задач багатопараметричної лінійної екстраполяції. Програма надає користувачеві інтуїтивно зрозумілий інтерфейс та набір функцій, що дозволяють вводити та обробляти багатовимірні дані, проводити лінійну екстраполяцію та отримувати точні та достовірні результати.

Експлуатаційне призначення програми полягає у наданні користувачеві засобу для виконання точної та надійної екстраполяції значень функції за межами заданого діапазону на основі відомих даних. Програма може бути використана в різних областях та додатках, де потрібне проведення екстраполяції на основі багатовимірних даних.

4. ВИМОГИ ДО ПРОГРАМИ

4.1. Вимоги до функціональних характеристик

Забезпечення зручного та ефективного вирішення задач багатопараметричної лінійної екстраполяції. Програма надає користувачеві інтуїтивно зрозумілий інтерфейс та набір функцій, що дозволяють вводити та обробляти багатовимірні дані, проводити лінійну екстраполяцію та отримувати точні та достовірні результати.

4.2. Вимоги до надійності

Вимоги до надійності наступні: забезпечення стійкого функціонування програми; контроль вхідної і вихідної інформації; наявність архівної копії тексту програми на зовнішньому носії.

4.3. Умови експлуатації

Вимоги до кліматичних умов: температура – 21-25 С, відносна вологість 40-60%. Обслуговування не потрібне. Для роботи із ПЗ достатньо однієї людини, що має досвід роботи із ПК, і бажає проводити дослідження у сфері навчання і оброблення даних за допомогою лінійної багатопараметричної екстраполяції .

4.4. Вимоги до складу і параметрів технічних засобів

Склад технічних засобів: процесор з тактовою частотою 2 ГГЦ або вище; 6 Мб. місця на накопичувачі; 2 Гб. оперативної пам’яті.

4.5. Вимоги до інформаційної і програмної сумісності

Програма має функціонувати під управлінням ОС Windows 10\11.

Програма написана на мові програмування C#. Перевагою цієї мови програмування є те, що вона об’єктно-орієнтована, тобто код можна поділити на логічні частини. Також ця мова має великий рівень безпеки і створення для написання додатків на операційну систему Windows. На системі має бути встановлений .NET Framework 4.5 або вище.

4.6. Вимоги до маркування і упаковки

Упаковка програмного продукту, включаючи документацію повинна бути захищена від пошкоджень різного роду (механічних, кліматичних). На упаковці повинно бути вказана назва продукту, мінімальні системні вимоги. На зворотній стороні упаковки вказується розробник та його юридична адреса.

4.7. Вимоги до транспортування і зберігання

Транспортування повинно проводитись в упаковці. Умови зберігання повинні забезпечувати безпеку продукту.

5. ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

До складу програмної документації мають входити:

1. специфікація;
2. текст програми.

Програмна документація повинна відповідати вимогам ДСТУ [1].

6. СТАДІЇ ТА ЕТАПИ РОЗРОБКИ

таблиця А.1

№ пор	Зміст роботи (розділу)	Термін виконання розділів роботи	Примітка
1	Вступ	12.09.22 – 26.10.22	
2	Аналіз сучасного стану дослідження проблеми за науковими літературними джерелами	27.10.22 – 04.03.23	
3	Аналіз сучасного стану програмно-апаратного забезпечення, яке потребує вдосконалення для вирішення проблем дослідження	05.03.23 – 31.04.23	
4	Постановка задачі, технічне завдання	01.05.23 – 07.05.23	30%
5	Техніко-економічні показники	08.05.23 – 15.05.23	
6	Розробка інструментальних засобів дослідження	16.05.23 – 21.05.23	
7	Виконання досліджень	22.05.23 – 28.05.23	60%
8	Оформлення результатів дипломної роботи	29.05.23 – 11.06.23	100%
9	Подання дипломної роботи до кафедри		
10	Захист дипломної роботи на засіданні екзаменаційної комісії	27.06.23	

7. ПОРЯДОК КОНТРОЛЮ І ПРИЙМАННЯ

Контроль виконання здійснює керівник розробки Скалозуб В.В. Прийом здійснюється уповноваженою комісією.

8. БІБЛІОГРАФІЧНИЙ СПИСОК

1. Івченко, Ю.М. Основи стандартизації програмних систем [Текст]: методичні вказівки до дипломного проектування та лабораторних робіт / уклад.: Ю. М. Івченко, В. І. Шинкаренко, В. Г. Івченко; Дніпропетр. нац. ун-т залізн. трансп. ім. акад. В. Лазаряна. – Д.: Вид-во Дніпропетр. нац. ун-ту залізн. трансп. ім. акад. В. Лазаряна, 2009. - 38 с.

Ошибка! Закладка не определена.

Додаток Б – Специфікація

ЗАТВЕРДЖЕНО

1116130.01304-01-ЛЗ

БАГАТОПАРАМЕТРИЧНІ ЕКСТРАПОЛЯЦІЙНІ МОДЕЛІ ТА АЛГОРИТМИ ВИБОРУ ТА ПРОЕКТУВАННЯ

Специфікація

1116130.01304-01

Листів 2

2023

Специфікації

Таблиця А.1. – Специфікації

Позначення	Найменування	Примітка
	Документація	
1116130.01304-01-ЛЗ	Лист затвердження	
1116130.01304-01	Технічне завдання	
1116130.01304-01-ЛЗ	Лист затвердження	
1116130.01304-01	Специфікація	
1116130.01304-01 12 01-ЛЗ	Лист затвердження	
1116130.01304-01 12 01	Текст програми	

Додаток В – Листи затвердження

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського державного університету науки і технологій
_____ Анатолій РАДКЕВИЧ
07.12.2022

БАГАТОПАРАМЕТРИЧНІ ЕКСТРАПОЛЯЦІЙНІ МОДЕЛІ ТА АЛГОРИТМИ ВИБОРУ ТА ПРОЕКТУВАННЯ

Технічне завдання
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01304-01-ЛЗ

Представники підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки
_____ Владислав СКАЛОЗУБ
07.12.22
Виконавець
Антон ГАРКУША
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського державного університету науки і технологій
_____ Анатолій РАДКЕВИЧ
07.12.2022

БАГАТОПАРАМЕТРИЧНІ ЕКСТРАПОЛЯЦІЙНІ МОДЕЛІ ТА АЛГОРИТМИ ВИБОРУ ТА ПРОЕКТУВАННЯ

Специфікація
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01304-01-ЛЗ

Представники підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки ___
Владислав СКАЛОЗУБ
07.12.22
Виконавець
___ Антон ГАРКУША
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАТВЕРДЖУЮ
Проректор Українського державного університету науки і технологій
_____ Анатолій РАДКЕВИЧ
07.12.2022

БАГАТОПАРАМЕТРИЧНІ ЕКСТРАПОЛЯЦІЙНІ МОДЕЛІ ТА АЛГОРИТМИ ВИБОРУ ТА ПРОЕКТУВАННЯ

Текст програми
ЛИСТ ЗАТВЕРДЖЕННЯ
1116130.01304-01 12 01-ЛЗ

Представники підприємства-розробника
Завідувач кафедри КІТ
_____ Вадим ГОРЯЧКІН
07.12.22
Керівник розробки
_____ Владислав СКАЛОЗУБ
07.12.22
Виконавець
_____ Антон ГАРКУША
07.12.22
Нормконтролер
_____ Світлана ВОЛКОВА
07.12.22

2023

Додаток Г – текст програми

ЗАТВЕРДЖЕНО
1116130.01304-01 12 01-ЛЗ

БАГАТОПАРАМЕТРИЧНІ ЕКСТРАПОЛЯЦІЙНІ МОДЕЛІ ТА АЛГОРИТМИ ВИБОРУ ТА ПРОЕКТУВАННЯ

Текст програми

1116130.01304-01 12 01

Листів **Элементы оглавления не найдены.**

Form1.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
using MathNet.Numerics;
using MathNet.Numerics.LinearAlgebra;
using System.IO;
using Newtonsoft.Json;

namespace Diplom
{
    public struct Buttons
    {
        public System.Windows.Forms.TextBox[,] XsTextBoxes;
        public System.Windows.Forms.TextBox[] YsTextBoxes;
        public System.Windows.Forms.TextBox[] InputTextBoxes;
        public System.Windows.Forms.TextBox[] VarsTextBoxes;
        public System.Windows.Forms.TextBox YName;
        public Buttons(System.Windows.Forms.TextBox[,] XsTextBoxet,
            System.Windows.Forms.TextBox[] YsTextBoxest,
            System.Windows.Forms.TextBox[] InputTextBoxest,
            System.Windows.Forms.TextBox[] VarsTextBoxest,
            System.Windows.Forms.TextBox YNamet
            )
        {
```

```
        YName = YNamet;
        XsTextBoxes = XsTextBoxet;
        YsTextBoxes = YsTextBoxest;
        InputTextBoxes = InputTextBoxest;
        VarsTextBoxes = VarsTextBoxest;
    }

    public Buttons(int a)
    {
        XsTextBoxes = new System.Windows.Forms.TextBox[a, a];
        YsTextBoxes = new System.Windows.Forms.TextBox[a];
        InputTextBoxes = new System.Windows.Forms.TextBox[a];
        VarsTextBoxes = new System.Windows.Forms.TextBox[a];
        YName = new System.Windows.Forms.TextBox();
    }
}
public struct Gaps
{
    public int start_x;
    public int start_y;
    public int gap_x;
    public int gap_y;
    public int size_x;
    public int size_y;
    public int big_gap_x;
    public int big_gap_y;
```

```
    public Gaps(int a)
    {
        start_x     = 160;
        start_y     = 47;
        gap_x       = 0;
        gap_y       = 0;
        size_x      = 50;
        size_y      = 20;
        big_gap_x   = 20;
        big_gap_y   = 20;
    }
}
public partial class Form1 : Form
{

    // Создаем двухмерный динамический массив целых чисел
    float[,] XMassive;
    float[] YMassive;
    float[] InputMassive;
    string[] Vars;
    string YName;
    int n1counter = 0, n2counter = 0;
    public Buttons b;
    public Gaps g = new Gaps(0);
    string FilesDirectory = Path.Combine(Directory.GetCurrentDirectory(),
```

```
"DipProjects\\");
        string ExampsFilesDirectory = "\\DipProjects\\Examples\\";

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            ProjectsToolStripMenuItem_Load(ProjectsToolStripMenuItem,
FilesDirectory);
        }
        private void ProjectsToolStripMenuItem_Load(ToolStripMenuItem Root,
string path)
        {
            string[] files = Directory.GetFileSystemEntries(path);
            foreach (string FilePath in files)
            {
                if (FilePath.EndsWith(".json"))
                {
                    // Создаем новый ToolStripMenuItem с текстом из массива
                    var toolStripMenuItem = new
ToolStripMenuItem(FilePath.Replace(path, "").Replace(".json", ""))
                    {
                        Name = FilePath
```

```
            };

            // Добавляем обработчик события Click для каждого пункта
меню
            toolStripMenuItem.Click += ToolStripMenuItem_Click;

            // Добавляем ToolStripMenuItem в ToolStripMenu
            Root.DropDownItems.Add(toolStripMenuItem);
        }
        else if (!FilePath.Contains("."))
        {
            var toolStripMenuItem = new
ToolStripMenuItem(FilePath.Replace(path, ""))
            {
                Name = FilePath
            };
            ProjectsToolStripMenuItem_Load(toolStripMenuItem,
FilePath+"\\");

            Root.DropDownItems.Add(toolStripMenuItem);
        }
    }
}

private void ToolStripMenuItem_Click(object sender, EventArgs e)
{
```

```
        var item = (ToolStripMenuItem)sender;

        var deserializedData =
JsonConvert.DeserializeObject<dynamic>(File.ReadAllText(item.Name));

        Exaples(deserializedData.XMassive.ToObject<float[,]>(),
            deserializedData.YMassive.ToObject<float[]>(),
            deserializedData.InputMassive.ToObject<float[]>(),
            deserializedData.Vars.ToObject<string[]>(),
            deserializedData.YName.ToObject<string>(),
            sender, e);

    }
    private void TextBox2_TextChanged(object sender, EventArgs e)
    {

    }
    public void RefreshButton_Click(object sender, EventArgs e)
    {

        if (int.TryParse(textBox1.Text, out int n1) &&
            int.TryParse(textBox2.Text, out int n2) &&
            n1 > 0 && n2 > 0)
        {
            XMassive = new float[n1, n2];
            YMassive = new float[n1];
```

```
InputMassive = new float[n2];
Vars = new string[n2];
//
for (int i = 0; i < n1counter && i < n1; i++)
{
    float n;
    for (int j = 0; j < n2counter && j < n2; j++)
    {
        if (float.TryParse(b.XsTextBoxes[i, j].Text, out n))
        {
            XMassive[i, j] = n;
        }
    }
    if (float.TryParse(b.YsTextBoxes[i].Text, out n))
    {
        YMassive[i] = n;
    }
}
for (int i = 0; i < n2counter && i < n2; i++)
{
    if (float.TryParse(b.InputTextBoxes[i].Text, out float n))
    {
        InputMassive[i] = n;
    }
    Vars[i] = b.VarsTextBoxes[i].Text;
}
```

```
//удаление кнопок
for (int i = 0; i < n1counter; i++)
{
    for (int j = 0; j < n2counter; j++)
    {
        b.XsTextBoxes[i, j].Dispose();
    }
    b.YsTextBoxes[i].Dispose();
}
for (int i = 0; i < n2counter; i++)
{
    b.InputTextBoxes[i].Dispose();
    b.VarsTextBoxes[i].Dispose();
}
b.YName?.Dispose();

var XsTextBoxes = new System.Windows.Forms.TextBox[n1, n2];
var YsTextBoxes = new System.Windows.Forms.TextBox[n1];
var InputTextBoxes = new System.Windows.Forms.TextBox[n2];
var VarsTextBoxes = new System.Windows.Forms.TextBox[n2];
var YNameBoxe = new System.Windows.Forms.TextBox();


for (int i = 0; i < n1; i++)
{
    for (int j = 0; j < n2; j++)
```

```
            {
                XsTextBoxes[i, j] = new System.Windows.Forms.TextBox
                {
                    Location = new Point(
                    g.start_x + g.big_gap_x + (j + 1) * (g.size_x + g.gap_x),
                    g.start_y + g.big_gap_y + (i + 1) * (g.size_y + g.gap_y)),
                    Width = g.size_x,
                    Height = g.size_y,
                    Text = XMassive[i, j].ToString(),
                };
                XsTextBoxes[i, j].TextChanged +=
XsTextBoxes_TextChanged;
                this.Controls.Add(XsTextBoxes[i, j]);
                XMassive[i, j] = float.Parse(XsTextBoxes[i, j].Text);
            }
            YsTextBoxes[i] = new System.Windows.Forms.TextBox
            {
                Location = new Point(
                g.start_x,
                g.start_y + g.big_gap_y + (i + 1) * (g.size_y + g.gap_y)),
                Size = new Size(g.size_x, g.size_y),
                Text = YMassive[i].ToString()
            };
            YsTextBoxes[i].TextChanged += YsTextBoxes_TextChanged;
            this.Controls.Add(YsTextBoxes[i]);
            YMassive[i] = float.Parse(YsTextBoxes[i].Text);
```

```
            }
            for (int j = 0; j < n2; j++)
            {
                InputTextBoxes[j] = new System.Windows.Forms.TextBox
                {
                    Location = new Point(
                    g.start_x + g.big_gap_x + (j + 1) * (g.size_x + g.gap_x),
                    g.start_y),
                    Size = new Size(g.size_x, g.size_y),
                    Text = InputMassive[j].ToString()
                };
                InputTextBoxes[j].TextChanged +=
InputTextBoxes_TextChanged;
                this.Controls.Add(InputTextBoxes[j]);

                InputMassive[j] = float.Parse(InputTextBoxes[j].Text);

                VarsTextBoxes[j] = new System.Windows.Forms.TextBox
                {
                    Location = new Point(
                    g.start_x + g.big_gap_x + (j + 1) * (g.size_x + g.gap_x),
                    g.start_y - g.big_gap_y),
                    Size = new Size(g.size_x, g.size_y)
                };
                if (Vars[j] != null)
                    VarsTextBoxes[j].Text = Vars[j];
```

```
    else
        VarsTextBoxes[j].Text = "X" + j;
    VarsTextBoxes[j].TextChanged += VarsTextBoxes_TextChanged;
    this.Controls.Add(VarsTextBoxes[j]);

    Vars[j] = VarsTextBoxes[j].Text;
}
YNameBoxe = new System.Windows.Forms.TextBox
{
    Location = new Point(
    g.start_x,
    g.start_y - g.big_gap_y),
    Size = new Size(g.size_x, g.size_y)
};
if (YName != null)
    YNameBoxe.Text = YName;
else
    YNameBoxe.Text = "Y";
YNameBoxe.TextChanged += VarsTextBoxes_TextChanged;
this.Controls.Add(YNameBoxe);

YName = YNameBoxe.Text;

n1counter = n1;
n2counter = n2;
b = new Buttons(XsTextBoxes, YsTextBoxes, InputTextBoxes,
```

```
VarsTextBoxes, YNameBoxe);
            rangeLabe.Location = new Point(g.start_x, g.start_y + g.gap_y + 15);
            OutputLable.Location = new Point(g.start_x, g.start_y);
        }
    }
    private void XsTextBoxes_TextChanged(object sender, EventArgs e)
    {
        // Обработчик события TextChanged
        System.Windows.Forms.TextBox textBox =
(System.Windows.Forms.TextBox)sender;

        int x = (textBox.Location.X - (g.start_x + g.big_gap_x + g.size_x +
g.gap_x))
            / (g.size_x + g.gap_x);
        int y = (textBox.Location.Y - (g.start_y + g.big_gap_y + g.size_y +
g.gap_y))
            / (g.size_y + g.gap_y);
        if (float.TryParse(textBox.Text, out float n))
        {
            XMassive[y, x] = n;
        }
        else
        {
            textBox.Text = XMassive[y, x].ToString();
        }
    }
```

```
        private void YsTextBoxes_TextChanged(object sender, EventArgs e)
        {
            // Обработчик события TextChanged
            System.Windows.Forms.TextBox textBox =
(System.Windows.Forms.TextBox)sender;

            int y = (textBox.Location.Y - (g.start_y + g.big_gap_y + g.size_y +
g.gap_y))
                / (g.size_y + g.gap_y);
            if (float.TryParse(textBox.Text, out float n))
            {
                YMassive[y] = n;
            }
            else
            {
                textBox.Text = YMassive[y].ToString();
            }
        }
        private void InputTextBoxes_TextChanged(object sender, EventArgs e)
        {
            // Обработчик события TextChanged
            System.Windows.Forms.TextBox textBox =
(System.Windows.Forms.TextBox)sender;
            int x = (textBox.Location.X - (g.start_x + g.big_gap_x + g.size_x +
g.gap_x))
                / (g.size_x + g.gap_x);
```

```
            if (float.TryParse(textBox.Text, out float n))
            {
                InputMassive[x] = n;
            }
            else
            {
                textBox.Text = InputMassive[x].ToString();
            }
        }
        private void VarsTextBoxes_TextChanged(object sender, EventArgs e)
        {
            // Обработчик события TextChanged
            System.Windows.Forms.TextBox textBox =
(System.Windows.Forms.TextBox)sender;
            int x = (textBox.Location.X - (g.start_x + g.big_gap_x + g.size_x +
g.gap_x))
                / (g.size_x + g.gap_x);
            if (x >= 0)
                Vars[x] = textBox.Text;
            else YName = textBox.Text;

        }
        private void Ex1_Click(object sender, EventArgs e)
        {
            float[,] X = { {2f, 1f ,1.5f , 2.5f, 1f},
                           {3f, 2.5f, 3.5f, 3f, 2f },
```

```
                {4f, 3.5f, 5f, 4.5f, 3f },
                {5f, 5f, 6f, 5f, 5f },
                {6f, 3f, 3f, 2f, 2f }};
        float[] Y = { 19.75f, 33f, 47.25f, 60f, 25f };
        float[] In = { 3f, 2.5f, 3f, 3.5f, 2f };
        Exaples(X, Y, In, sender, e);



    }
    private void Exaples(float[,] X, float[] Y, float[] In, object sender,
EventArgs e)
    {
        if (X.GetLength(0) != Y.GetLength(0) || X.GetLength(1) !=
In.GetLength(0)) { return; }
        textBox1.Text = X.GetLength(0).ToString();
        textBox2.Text = X.GetLength(1).ToString();
        RefreshButton_Click(sender, e);
        for (int i = 0; i < X.GetLength(0); i++)
        {
            for (int j = 0; j < X.GetLength(1); j++)
            {
                b.XsTextBoxes[i, j].Text = X[i, j].ToString();
            }
            b.YsTextBoxes[i].Text = Y[i].ToString();
        }
        for (int i = 0; i < X.GetLength(1); i++)
```

```
            {
                b.InputTextBoxes[i].Text = In[i].ToString();
                b.VarsTextBoxes[i].Text = "X" + i;
            }
            b.YName.Text = "Y";
        }
        private void Exaples(float[,] X, float[] Y, float[] In, string[] vars, string
Yname, object sender, EventArgs e)
        {
            if (X.GetLength(0) != Y.GetLength(0) || X.GetLength(1) !=
In.GetLength(0)) { return; }
            textBox1.Text = X.GetLength(0).ToString();
            textBox2.Text = X.GetLength(1).ToString();
            RefreshButton_Click(sender, e);
            for (int i = 0; i < X.GetLength(0); i++)
            {
                for (int j = 0; j < X.GetLength(1); j++)
                {
                    b.XsTextBoxes[i, j].Text = X[i, j].ToString();
                }
                b.YsTextBoxes[i].Text = Y[i].ToString();
            }
            for (int i = 0; i < X.GetLength(1); i++)
            {
                b.InputTextBoxes[i].Text = In[i].ToString();
                b.VarsTextBoxes[i].Text = vars[i];
```

```
        }
        b.YName.Text = Yname;
    }
    public double FunctionRange(Vector<double> L)
    {
        double s = 0f;
        for (int j = 0; j < n2counter; j++)
        {
            double ss = 0f;
            for (int i = 0; i < n1counter; i++)
            {
                ss += L[i] * XMassive[i, j];
            }
            s += (ss - InputMassive[j]) * (ss - InputMassive[j]);
        }
        return s;
    }
    public Vector<double> GradFuncion(Vector<double> L)
    {
        Vector<double> grad = Vector<double>.Build.Dense(n1counter, 0d);
        for (int i = 0; i < n1counter; i++)
        {
            for (int j = 0; j < n2counter; j++)
            {
                grad[i] += 2 * L[i] * XMassive[i, j] * XMassive[i, j] - 2 *
XMassive[i, j] * InputMassive[j];
```

```
        }
    }
    return grad;
}
public double FunctionY(Vector<double> L)
{
    double y = 0d;
    for (int i = 0; i < n1counter; i++)
    {
        y += YMassive[i] * L[i];
    }
    return y;
}

private void GapsToolStripMenuItem_Click(object sender, EventArgs e)
{
    Form2 form2 = new Form2(this);
    form2.ShowDialog();
}
private void Calc_button_Click(object sender, EventArgs e)
{
    var label3 = new System.Windows.Forms.Label
    {
        AutoSize = true,
        Location = new System.Drawing.Point(100, 45),
        Name = "label3",
```

```
            Size = new System.Drawing.Size(59, 13),
            TabIndex = 10,
            Text = "Результат"
        };
        this.Controls.Add(label3);
        var k = RandMinimize(FunctionRange);
        string text = FunctionY(k).Round(2).ToString();
        if (FunctionRange(k).Round(3) > 0) text += "±" +
Math.Sqrt(FunctionRange(k)).Round(1).ToString();
        OutputLable.Text = text;
    }
    private void SaveToolStripMenuItem1_Click(object sender, EventArgs e)
    {

        SaveFileDialog saveFileDialog = new SaveFileDialog
        {
            // Настройка фильтра файлов и начальной директории
            Filter = "Файлы Json (*.json)|*.json|Все файлы (*.*)|*.*|Текстовые
файлы (*.txt)|*.txt",
            InitialDirectory =
            //Directory.GetCurrentDirectory() +
            FilesDirectory
        };

        // Открытие диалогового окна сохранения файла
        if (saveFileDialog.ShowDialog() == DialogResult.OK)
```

```
        {
            // Сохранение файла

            using (StreamWriter sw = new
StreamWriter(saveFileDialog.FileName))
            {
                JsonSerializer serializer = new JsonSerializer();
                var data = new
                {
                    XMassive,
                    YMassive,
                    InputMassive,
                    Vars,
                    YName,
                };
                serializer.Serialize(sw, data);
            }

        }

    }
    private void LoadToolStripMenuItem_Click(object sender, EventArgs e)
    {
        OpenFileDialog openFileDialog = new OpenFileDialog
        {
            // Настройка фильтра файлов и начальной директории
```

```
            Filter = "Файлы Json (*.json)|*.json|Все файлы (*.*)|*.*|Текстовые файлы (*.txt)|*.txt",
            InitialDirectory =
            //Directory.GetCurrentDirectory() +
            FilesDirectory
        };

        // Открытие диалогового окна открытия файла
        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            // Чтение файла
            var deserializedData = JsonConvert.DeserializeObject<dynamic>(File.ReadAllText(openFileDialog.FileName));

            Exaples(deserializedData.XMassive.ToObject<float[,]>(),
                deserializedData.YMassive.ToObject<float[]>(),
                deserializedData.InputMassive.ToObject<float[]>(),
                deserializedData.Vars.ToObject<string[]>(),
                deserializedData.YName.ToObject<string>(),
                sender, e);

        }
    }

    private void ExitToolStripMenuItem_Click(object sender, EventArgs e)
```

```
    {

    }

    private void Label1_Click(object sender, EventArgs e)
    {

    }

    public Vector<double> RandMinimize(Func<Vector<double>, double>
func)
    {
        double h = 0.2, h0 = 1e-6;
        int mp = 100;

        Vector<double> x0 = Vector<double>.Build.Dense(n1counter, 1);
        int r = 0;
        double[] xmin = x0.ToArray();
        double ymin = func(x0);
        Vector<double> xt = Vector<double>.Build.Dense(n1counter);
        Random random = new Random();

        double[] txmin = new double[n1counter];
        double tymin = func(x0);

        while (ymin > h0)
```

```
{
    for (int c = 0; c < mp; c++)
    {
        for (int i = 0; i < n1counter; i++)
        {
            xt[i] = xmin[i] + (-h + 2 * h * (random.NextDouble()));
        }
        var f = func(xt);
        if (tymin > f)
        {
            txmin = xt.ToArray();
            tymin = f;
        }
    }
    if (ymin > tymin)
    {
        xmin = txmin;
        ymin = tymin;
        h *= 1.1;
    }
    else
    {
        r++;
        if (r >= mp)
        {
            h *= 0.9;
```

```
                    if (h <= h0)
                    {
                        break;
                    }
                }
            }
        }
        return Vector<double>.Build.Dense(xmin);
    }

}

}
```

Form2.cs

UnitTest1.sc