Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу”

КбПЗ_2025

Виконав здобувач вищої освіти
IV курсу, групи КБ-21
ОПП «Кібербезпека»
спеціальності 125 «Кібербезпека»
_____ Панкратьєва В.О.
«____» _____ 2025 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Смірнов С.А.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *125 “*Кібербезпека*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Кібербезпека*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Панкратьєвій Вероніці Олегівні
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу*

2. Керівник роботи *Смірнов Сергій Анатолійович, канд. техн. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 57-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи кібербезпеки в промислову експлуатацію.

6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи кібербезпеки	*1 аркуш*
Функціональна схема системи кібербезпеки	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

КОПІЯ 2025

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

КбПЗ_2025

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Смірнов С.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Панкратьєва В.О.
(прізвище та ініціали)

АНОТАЦІЯ

Панкратьєва В.О. Програмне забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу. 125 Кібербезпека. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

Метою розробки є програмне забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

Результат роботи – програмна реалізація системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Visual C#.

Ключові слова: кібербезпека, стегоаналіз

КБПЗ_2025

ABSTRACT

Pankrat'eva V.O. Software for a cybersecurity system to detect unauthorized leakage of confidential information using stegoanalysis. 125 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for a cybersecurity system to detect unauthorized leakage of confidential information using stegoanalysis.

The purpose of the development is software for a cybersecurity system to detect unauthorized leakage of confidential information using stegoanalysis.

The result of the work is a software implementation of a cybersecurity system to detect unauthorized leakage of confidential information using stegoanalysis.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A user-friendly user interface has been developed. Instructions for working with the software are provided.

The program can be used on a PC with Windows 10/11.

The program was developed in Visual C#.

Keywords: cybersecurity, steganalysis

КОПЗ – 2025

ЗМІСТ

КОПЗ – 2025

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ДСТУ	–	державний стандарт України
ЕД	–	електронний документ
ЗІ	–	захист інформації
ЕЦП	–	електронний цифровий підпис

КБПЗ_2025

ВСТУП

Актуальність теми. Стеганографія та стеганаліз зображень, які передбачають приховування та виявлення прихованих даних у зображеннях, привернули значну увагу в останні роки, знайшовши застосування в різних сферах, як-от військова справа, медицина, електронний уряд та соціальні мережі. Незважаючи на їхню важливість у реальних додатках, деякі практичні аспекти залишаються поза увагою. Щоб подолати цю прогалину, у поточному дослідженні порівнюються інструменти та методики стеганографії та стегааналізу зображень для цифрових судових слідчих (DFI) для виявлення прихованої інформації в зображеннях. Проведено ретельний аналіз методів штучного інтелекту, статистики та сигнатурного стеганалізу, оцінюємо як безкоштовні, так і платні версії та експериментуємо з різними характеристиками зображення, такими як розмір, колір, середньоквадратична помилка (MSE), середньоквадратична помилка (RMSE) і пікове співвідношення сигнал/шум (PSNR), використовуючи набір даних JPEG/PNG. Дослідження дає цінну інформацію для професіоналів у сфері кібербезпеки.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

– Дослідження системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

– Програмна реалізація системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

КПІ – 2025

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КБПЗ_2025

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Існують різні методи стеганалізу. Ці методи були розроблені для боротьби зі специфічними методами стеганографії та виявлення даних, прихованих у певних форматах зображень. Проте жоден метод або інструмент стеганографії не може виявити всі типи стеганографії або підтримувати всі доступні формати зображень. Однією з проблем є потреба в більш загальній системі, яка б охоплювала різні типи форматів зображень, і здатність виявляти ширший діапазон стегозображень, створених наосліп за допомогою багатьох методів стеганографії. У цьому документі представлено систему стегоаналізу зображень, щоб розрізнити чисті та стегозображення за допомогою трьох різних методів.

Перший метод полягає в вилученні великої кількості характеристик зображення з матриці спільного появи градієнта кольорів (CGCM).

Другий – виділення ряду характеристик гістограми шляхом використання гістограми різницевого зображення, яка зазвичай є узагальненим розподілом Гауса з центром у 0. Нарешті, перевірені функції CGCM і характеристики гістограми були об’єднані для покращення продуктивності системи.

Об’єднання двох різних типів функцій дозволяє скористатися корисними властивостями кожної з них, щоб підвищити здатність системи щодо виявлення. Результати експерименту демонструють, що запропонована система має надійну здатність виявлення та точність.

Запропонована система є більш узагальненим детектором, ніж попередні системи, що охоплює більшу різноманітність типів і форматів стегозображень. Крім того, експериментальні результати показують, що запропонована система стеганалізу працює значно краще, ніж деякі попередні методи виявлення

КОПЗ-2025

1.2 Область застосування

Областю застосування є системи виявлення несанкціонованого витоку конфіденційної інформації. Безпека стеганосистем, що використовують той або інша методи стеганографічного приховання, описується й оцінюється їхньою стійкістю.

Стеганографія – це процес приховування повідомлень всередині об’єкта, відомого як носій. Ідея полягає у створенні прихованого каналу зв’язку, де повідомлення залишаються непоміченими для спостерігачів, які мають доступ до цього каналу. Steganalysis призначений для виявлення таких прихованих повідомлень; ці повідомлення можна вставляти в кілька різних типів цифрових носіїв, таких як зображення, відео, аудіо, звичайні текстові файли та приховані канали. Традиційні схеми стеганалізу поділяються на два етапи; перший передбачає ручне вилучення високоякісних функцій, а другий – класифікацію за допомогою методів машинного навчання (ML). Сьогодні глибоке навчання (DL) може об’єднати обидва традиційні етапи в наскрізну схему з багатообіцяючими результатами. У цьому розділі ми показуємо найактуальніші методи стеганалізу з використанням статистичних методів, методів ML і DL на цифрових носіях.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

КОПЗ-2025

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Stegdetect

Stegdetect досить ефективний проти великого числа стеганографічних програм: JSTEG, JPHS, Gifshuffle, Hide-and-Seek, Steganos. Stegdetect першою справою перевіряє самі доступні сховища: поля-коментарі й поля розширень різних форматів файлів, наявність штучно створених зображень, а також зображень із більшою кількістю ділянок однотипного заливання. Програма порівнює частоту розподілу кольорів для можливого носія схованої інформації й теоретично очікувану частоту розподілу кольорів для файлу-носія схованої інформації. Це, можливо, не найшвидший метод захисту, але якщо виникають підозри на рахунок протиправної діяльності, те цей метод може бути найефективнішим. Однак подужати більше складну «значеннєве навантаження» у текстових документах програмі вже не під силу. Але ж безглузді фрази, швидше за все, згенеровані спеціальними стего-програмами по словниках (або написані божевільними роботами-колонізаторами).

У той же час Stegdetect показує досить упевнені результати в тому випадку, якщо зміст вкладення перевищує 10 % обсягу файлу-контейнера.

FTK Imager

FTK Imager дозволяє швидко створити образ жорсткого диска для наступного вивчення, а також на льоту переглянути файли MS Office, архівів або зображень. Ви можете самостійно вибрати, які формати хочете переглянути – всі вони розсортовані за типом.

КПЗ-2025

Аналіз даних здійснюється завдяки убудованій у програму базі контрольних сум (є можливість і імпортувати ззовні) – всі файли, що містять у собі вкладення або додаткові зміни, будуть відразу ж відображені. Як відомо, ще не вдалося створити файл-контейнер, що не змінив би свій розмір, прийнявши стего-файл. Дубльовані, архівірувані, шифровані, файли з невірним розширенням і так далі виводяться окремо для наступного вивчення. Можливі будь-які маніпуляції із вкладеннями плюс максимум інформації, що тільки можна одержати з наявного у вас файлу. Програму відрізняє досить дружелюбний інтерфейс, інформативність і високу ціну.

Stego Suite

Автоматичний програмний сканер, що містить 9 стеганографічних алгоритмів детектування, розрахованих на всі загальні типи файлів цифрового зображення й аудіо файлів. Складається з модулів:

– Stego Analyst – візуального аналітичного пакета для всебічного аналізу цифрових зображень і аудіо файлів.

– Stego Break – інструмента злому стеганографічного захисту.

Дозволяє разом знищити дані, заховані по алгоритму найменш значущих біт.

Stego Suite змінює молодші розряди кожного байта мультимедіафайла на нульовий біт. При цьому якість зображення або звуку не змінюється.. Необразливу картинку ми в такий спосіб не зіпсуємо, але несанкціоновану передачу даних обов'язково припинимо.

File Signature Header

Деякі користувачі зловмисно змінюють розширення файлу. Або ховають конфіденційні відомості у файлі, аналогічному вже існуючому. Деякі просто перейменовують файл картинки, приміром, test.jpg в test.txt і інтелектуальна ОС Windows відкриє його в блокноті – замість картинки одержите безглуздий текст. Якщо у файлу немає розширення (а називається він dfhhu457785h), то при відсутності деяких знань його відкриття може стати проблемою.

КБПЗ – 2025

File Signature Header чудовий тим, що дозволяє не тільки визначити приналежність якого-небудь файлу, але й найчастіше ідентифікувати програму, його що створила, або загострити увагу на яких-небудь файлах (у рамках конкретної справи).

Використовую базу файлів з Інтернету, File Signature Header дозволяє досить швидко оцінити, у якому напрямку вести пошук. Якщо перевірка показала наявність графічних файлів-контейнерів, тобто зміст далі використовувати програму Stegdetect.

ProDiscover

Необхідний для роботи мінімум у програмі присутня, але для повного аналізу однаково знадобиться додаткове програмне забезпечення. Програма може працювати як з диском, так і з образом, також дозволяє роботу в мережі. Можлива перевірка файлів по File signature header (не плутайте з однойменною програмою) і hash set (бази контрольних сум файлів), але сам результат відображення не дає ніяких додаткових відомостей про дані. Програма лише висловлює свою підозру про ті або інші файли, нічим не підкріплюючи свою точку зору.

Ilook Investigator

Продукт поширюється безкоштовно. По кількості функцій можна зрівняти з FTK Imager і File Signature Header. Їсти можливість створити образ файлів, що перевіряються, змонтувати їх, перевірити по hash set file signature header (усе з можливістю додавання, редагування). У порівнянні з попередніми програмами, інтерфейс може здатися антиюзабілітним (лихо багатьох безкоштовних і/або програм, розроблених для держави).

Maresware Forensic Suite

По розповідях користувачів, самі краще стеганографічні програми ті, що прекрасно поводяться в командному рядку. Як видно, цим і керувалися творці MFS. Усе, що можна зробити в командному рядку, є присутнім. Всі можливості стандартні – створення й відновлення образа, підрахунок і порівняння

КБПЗ_2025

контрольних сум, перевірка за hash sets, file signature header, пошук по багатьом параметрам, включаючи ADS для NTFS, визначення файлів PGP і багато чого-багато чого іншого. Кожна з утиліт, що входить в Maresware Forensic Suite, гарна як окремий продукт, а в наборі просто винятково приємна річ. Так, і невелике додавання: програма реалізоване для роботи в середовищі Linux & Unix.

Paraben E-mail Examiner

Аналогічна версія – MailBag Assistant. Програма реалізована під електронну пошту. Підрахунок контрольних сум, перевірка тексту, підрахунок md5, і багато чого іншого. Прекрасно зарекомендувала себе програма.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Програмне забезпечення написане мовою Visual C#. Ця мова обрана виходячи з наступних міркувань. Visual C# – строго типізована об'єктно-орієнтована мова, призначена для розробки різноманітних безпечних і потужних додатків, виконуваних у середовищі .NET Framework. Мовою Visual C# можна розробляти звичайні клієнтські додатки Windows, веб-служби XML, розподілені компоненти, додатки типу " сервер-клієнт", додатки баз даних і багато яких інших. В Visual C# 2012 є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку додатків мовою Visual C# версії 5.0 і .NET Framework версії 4.5.

Синтаксис Visual C# дуже виразний, але простий у вивченні. Усі, хто знаком з мовами C, C++ або Java з легкістю взнають синтаксис із фігурними дужками, характерний для мови Visual C#. Розроблювачі, що знають кожну із цих мов, як правило, зможуть домогтися ефективної роботи з мовою Visual C# за дуже короткий час. Синтаксис Visual C# робить простіше те, що було складно в C++, і забезпечує потужні можливості, такі як типи значень Nullable,

КПЗ-2025

перерахування, делегати, лямбда-вираження й прямий доступ до пам'яті, чого немає в Java. Visual C# підтримує універсальні методи й типи, забезпечуючи більше високий рівень безпеки й продуктивності, а також ітератори, що дозволяють при реалізації колекцій класів визначати власне поводження ітерації, що може легко використовуватися в клієнтському коді. В Visual C# 5.0 вираження LINQ (Language-Integrated Query) роблять строго-типізований запит першокласною конструкцією мови.

Як об’єктно-орієнтована мова, Visual C# підтримує поняття інкапсуляції, спадкування й поліморфізму. Всі змінні й методи, включаючи метод Main – точку входу додатка – інкапсуюється у визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначають віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення. У мові Visual C# структура схожа на полегшений клас: це тип, що розподіляється по стопках, що реалізує інтерфейси, але не підтримуюче спадкування.

На додаток до основних описаних об’єктно-орієнтованих принципів, мова Visual C# спрощує розробку компонентів програмного забезпечення завдяки декільком інноваційним конструкціям мови, у число яких входять наступні:

– Інкапсульовані підписи методів, називані делегатами, які підтримують строго-типізовані повідомлення про події.

– Властивості, що виступають у ролі методів доступу для закритих змінних-членів.

– Атрибути з декларативними метаданими про типи під час виконання.

– Вбудовані коментарі XML-документації.

– LINQ (Language-Integrated Query), що пропонує вбудовані можливості запитів у різних джерелах даних.

Якщо буде потрібно забезпечити взаємодію з іншим програмним забезпеченням Windows, таким як об'єкти COM або власні бібліотеки DLL Win32,

КПЗ – 2025

у мові Visual C# можна використовувати процес, що називається "Interop". Процес Interop дозволяє програмам на Visual C# виконувати практично будь-які дії, які може виконувати вихідний додаток на C++. Мова Visual C# підтримує навіть покажчики й поняття "небезпечного" коду для тих випадків, коли прямий доступ до пам'яті має вкрай важливе значення.

Процес побудови Visual C# у порівнянні з C і C++ простий і є більше гнучким, чим в Java. Немає окремих файлів заголовка, а методи й типи не потрібно повідомляти в певному порядку. У вихідному файлі Visual C# може бути визначене будь-яке число класів, структур, інтерфейсів і подій.

Архітектура платформи .NET Framework

Програма мовою Visual C# виконується в середовищі .NET Framework – інтегрованому компоненті Windows, що містить віртуальну систему виконання (середовище CLR) і уніфікований набір бібліотек класів. Середовище CLR являє собою комерційну реалізацію корпорацією Майкрософт інфраструктури CLI, що є міжнародним стандартом, який лежить в основі створення середовищ виконання й розробки, у яких забезпечується тісна взаємодія між мовами й бібліотеками.

Вихідний код, написаний мовою Visual C#, компілюється в проміжну мову (IL) у відповідності зі специфікацією CLI. Код IL і ресурси, такі як растрові зображення й рядки, зберігаються на диску у файлі, що виконується, називаному складанням, з розширенням EXE або DLL у більшості випадків. Складання містить маніфест із відомостями про типи складання, версії, мови й регіональні параметри та вимоги безпеки.

При виконанні програми на Visual C# складання завантажується в середовище CLR залежно від відомостей у маніфесті. Далі, якщо вимоги безпеки дотримані, середовище CLR виконує JIT-компіляцію для перетворення коду IL в інструкції машинного коду. Середовище CLR також надає інші служби, що відносяться до автоматичного збору сміття, обробки виключень і керуванню ресурсами. Код, виконуваний середовищем CLR, іноді називають "керованим

КОПЗ_2025

кодом" у протиставлення "некерованому коду", що компілюється в машинний код, призначений для певної системи. Далі показані відносини під час компіляції й час виконання між файлами з вихідним кодом Visual C#, бібліотеками класів .NET Framework, складаннями й середовищем CLR.

Взаємодія між мовами є ключовою особливістю .NET Framework. Оскільки код IL, створюваний компілятором Visual C# відповідає специфікації CTS, код IL на основі Visual C# може взаємодіяти з кодом, створюваним версіями мов Visual Basic, Visual C++, Visual J# платформи .NET Framework і ще більш ніж 20 CTS-сумісних мов. В одному складанні може бути кілька модулів, написаних на різних мовах платформи .NET Framework, і типи можуть посилатися один на одного, як якби вони були написані на одній мові.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи кібербезпеки контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на

КПЗ-2025

екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи кібербезпеки в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

КбПЗ_2025

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Зі зростанням використання Інтернету та соціальних мереж безпека даних стала серйозною проблемою. Таким чином, дослідники зосереджуються на таких методах захисту даних, як стеганографія та стеганаліз. Стеганографія – це підхід до приховування існування секретних повідомлень у цифрових носіях для безпечної передачі. Методи стеганалізу спрямовані на виявлення прихованих повідомлень і вилучення їх. Методи стеганографії та стеганалізу цифрових зображень поділяються на просторову та трансформаційну області. У цій статті ми надаємо детальний огляд найсучасніших робіт, які були виконані в двовимірному та тривимірному стеганалізі зображень. Ми представляємо найпопулярніші набори даних і пояснюємо деякі стеганографічні методи вбудовування прихованих даних. Стеганаліз є дуже складним завданням через брак інформації про характеристики обкладинок, які можна використати для виявлення прихованих повідомлень. Тому ми розглядаємо дослідження, проведені зі стеганалізу зображень у просторовій та трансформаційній сферах, використовуючи підходи класичного машинного та глибокого навчання. Крім того, ми представляємо відкриті виклики та обговорюємо деякі напрямки майбутніх досліджень.

Щодня в Інтернеті публікується велика кількість цифрових медіа, таких як зображення, відео та аудіо. Ці цифрові носії можуть містити приховані повідомлення, які можна вставити на видноті за допомогою добре відомого методу під назвою «приховування інформації» [1], методу, який дозволяє користувачам приховувати інформацію за допомогою цифрових даних без відчутного впливу на дані. Таким чином, люди не можуть виявити, чи є приховане повідомлення всередині цифрових даних. Термін «стеганографія»

КОПІЯ 2025

утворено з двох грецьких термінів «steganos», що означає «покритий», і «graphein», що означає «письмо» [2]. Таким чином, стеганографія є методом, який приховує певну інформацію всередині цифрових даних [3]. Вхідні дані називаються об'єктом обкладинки, а вихідні – стегооб'єктом, який містить приховане повідомлення.

Існують великі відмінності між стеганографією та іншими методами вбудовування інформації. Основна відмінність між стеганографією та криптографією, наприклад, полягає в тому, що криптографія приховує повідомлення, тоді як стеганографія приховує присутність повідомлення. Водяні знаки використовуються для захисту прав власності власників і мають на меті додати до джерела додаткову інформацію (дані обкладинки). На даний момент існує багато методів стеганографії в просторовій та трансформаційній областях [4], [], [1]. Зі збільшенням розвитку та використання методів стеганографії з'явилася потреба у виявленні прихованих повідомлень. Стеганаліз – це підхід, який розрізняє, чи повідомлення приховано стеганографією всередині певного медіа [2]; його поділяють на пасивний і активний типи [2]. Приховані дані в просторовій області вбудовуються безпосередньо шляхом налаштування значення пікселів у зображенні обкладинки. Навпаки, приховані дані в домені перетворення вбудовані в коефіцієнти зображення обкладинки. Таким чином, пасивний стеганаліз виявляє наявність прихованих повідомлень, тоді як активний стеганаліз витягує приховані повідомлення, і вони далі класифікуються на просторові та трансформаційні домени. У методах стеганалізу зображень використовуються підходи на основі ознак для вилучення дискримінаційних атрибутів із зображень, таких як локальні двійкові шаблони (LBP) [4] і субтрактивна модель суміжності пікселів (SPAM) [4]. Методи стеганалізу можуть бути розроблені для виявлення конкретного алгоритму вбудовування, як у [1], [2] і [3]; вони також можуть діяти універсально для виявлення існування прихованих даних незалежно від алгоритму вбудовування, як у [4], [4] і [15]. Багато методів було запропоновано

КОПЗ–2025

для додатків стеганалізу на основі машинного навчання та алгоритмів глибокого навчання.

Класичне машинне навчання складається з двох частин: перша частина – це екстрактор ознак, а друга частина – це навчальний класифікатор. Екстрактор ознак використовується для отримання відмінних ознак із вхідних даних, які використовуються класифікатором для навчання. Машинне навчання включає багато методів, таких як машина опорних векторів (SVM) [16], інструмент машинного навчання, представлений Вапником, який можна використовувати для задач класифікації та регресії [17]. Інші популярні методи машинного навчання включають лінійну регресію, в якій результат прогнозується за допомогою відомого параметра, аналіз головних компонент (PCA), який зменшує розмірність набору даних [18], найближчий сусід [19], K-середнє кластеризування [20] тощо.

За останні кілька років моделі глибокого навчання привернули значну увагу в багатьох областях. У 2015 році була розроблена перша техніка стеганалізу з використанням згорткової нейронної мережі (CNN), яка є технікою глибокого навчання [21]. Глибоке навчання вважається підмножиною машинного навчання, яке можна розглядати як структуру чорної скриньки. Він об’єднує етапи виділення ознак і класифікації в один процес, таким чином дозволяючи наскрізний процес навчання машини. Моделі глибокого навчання використовують прямі процеси, щоб вивчати вилучення ознак і виконувати класифікацію безпосередньо з вхідних даних. Потім, у зворотному напрямку, виконується оновлення вилучених ознак на основі рішення класифікатора. Цей процес автоматично повторюється, доки похибка моделі не зменшиться. Рис. 1 ілюструє різницю між концепціями класичного машинного навчання та глибокого навчання. На рис. 1 (a) представлена класична концепція машинного навчання. На рис. 1 (b) представлено концепцію глибокого навчання.

Стеганаліз є дуже складною сферою через брак інформації про характеристики обкладинок, які можна використати для виявлення прихованих

КОПІЯ 2025

повідомлень. Існує багато алгоритмів для поля стеганалізу, які використовують три загальні засоби обкладинки: зображення, відео та аудіо. У цьому оглядовому документі ми зосереджуємося на алгоритмах стеганалізу для двовимірних (2D) і тривимірних (3D) зображень. Основна відмінність між стеганографією 2D і 3D зображень полягає в зображенні обкладинки. Для стеганографії у 2D-зображеннях обкладинка – це зображення, де повідомлення буде приховано в межах інтенсивності пікселів, тоді як у стеганографії 3D-зображень обкладинка – це тривимірна сітка, що складається з точок або вершин у 3D-геометрії, і нею буде маніпулювати, щоб приховати інформацію.

Вивчення сучасних методологій і розуміння майбутніх проблем у стеганалізі цифрових зображень допомагає дослідникам досягти кращих результатів у стеганографії та стеганалізі. Таким чином, у цьому огляді ми представляємо короткий виклад різних типів алгоритмів стеганалізу цифрових зображень, які були розроблені з використанням технологій класичного машинного та глибокого навчання. Ми в основному зосереджені на алгоритмах для зображень у просторовій та трансформаційній областях. Розробка та адаптація методів стеганалізу починається з гарного розуміння стеганографії.

3.2 Розробка структурної схеми

Для стеганографічних методів виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу першорядними стають питання забезпечення теоретичної й практичної стійкості. Найбільш перспективним у цьому напрямку, з огляду на сучасний стан теоретичної бази, бачиться побудова гібридних систем схованої передачі електронних документів на основі щільної взаємодії або навіть синтезу методів криптографії й стеганографії.

КБПЗ-2025

Рисунок 3.1 – Структурна схема системи

КбПЗ_2025

Набори даних стегааналізу зображень

Стеганографію можна використовувати на різних типах медіа, таких як зображення, відео, аудіо тощо. Тому дослідникам необхідно оцінити свої методи стеганалізу на великих наборах даних. Ми класифікуємо набори даних за двома категоріями: 2D і 3D, а 2D-набори можна розділити на сірі та кольорові зображення. У цьому розділі пояснюється набори даних, які зазвичай використовуються в області стеганалізу.

Двовимірні набори даних

Набори даних відтінків сірого

Завдання під назвою Breaking Our Watermarking System (BOWS) [5] було

створено у 2007 році спільнотою International Challenges in Information Forensics and Security для видалення водяних знаків із трьох зображень. Друге видання виклику (BOWS-2) було представлено в 2008 році [1] . Цей виклик став натхненням для створення виклику BOSS у 2010 році [28] . Набір даних BOSSbase, який включає 10 000 зображень у градаціях сірого, став одним із найбільш часто використовуваних наборів даних у галузі стеганалізу. Хоча BOSSbase мав великий вплив на дослідження стеганалізу, він має деякі обмеження, особливо з появою нових технологій. У 2017 році було створено набір даних BURSTbase [2] . Він містить7 × 9,310 зображення у форматі JPEG, зроблені камерою в режимі серійної зйомки. Таблиця. 1 представлені деталі наборів даних у відтінках сірого.

Кольорові набори даних

У 2018 році стартував новий конкурс під назвою Alaska1 [4] . Його метою було надати великий набір даних зображень різного розміру, зроблених різними камерами. Зображення стискалися з різними коефіцієнтами якості. У рамках завдання Alaska1 учасники отримали коди для стеганографічних схем для створення власних навчальних наборів, не хвилюючись про невідповідності обкладинки та джерела. Alaska2 [1] є наступником Alaska1. Він містить загалом 300 000 кольорових зображень, розділених на 75 000 наборів обкладинок, стегозображень (зображень із застосованими до них стеганографічними схемами, такими як J-UNIWARD [29] , J-MiPOD [2] і Uniform Embedding Revisited Distortion (UERD) [3]) і 5000 тестових зображень. IStego100K – це набір даних, створений у 2019 році [4] . Він містить 208 104 кольорових зображення розміру1024 × 1024 . Зображення були розділені на 100 000 зображень для обкладинки та 100 000 зображень для стего, а решта 8104 зображення використовувалися для тестування. Цей набір даних враховує невідповідність між навчальним набором і тестовим набором у реальному середовищі. ImageNet є одним із найпопулярніших наборів даних, які використовуються в задачах класифікації [4] . Він був створений для сприяння дослідженням комп’ютерного

КПЗ_2025

зору. Деякі дослідники використовують ImageNet як набір даних для стеганалізу, створюючи стегозображення за допомогою схеми стеганографії на свій вибір; його використовували для створення нових моделей для стеганалізу, оскільки він містить понад 14 мільйонів кольорових зображень різного розміру. RAISE – це ще один набір даних, створений для підтримки досліджень у галузі криміналістики зображень і обробки зображень [15] .

Відтінки сірого та кольорові набори даних

Небагато наборів даних містять як кольорові, так і сірі зображення. Одним із небагатьох, які це роблять, є Steganalysis Real Test Version 1 (STEGRT1) [17] . STEGRT1 – це новий набір даних, створений у 2020 році для оцінки систем стеганалізу в реальних сценаріях. STEGRT1 містить 8000 обкладинок і стего зображень різних розмірів і властивостей. Широкомасштабна база даних стегааналізу (LSSD) – це комбінація різних наборів даних [18] . Ідея об'єднання наборів даних полягає в тому, щоб збільшити різноманітність і представити сценарії реального світу.

Тривимірні набори даних

В даний час завдяки прогресу в 3D-технологіях і апаратному забезпеченні стало легко отримувати 3D-моделі природних об'єктів. Ці моделі стали широко використовуватися в різних сферах, таких як медична візуалізація, віртуальна реальність, доповнена реальність, ігри, фільми та багатьох інших областях. 3D-стеганографія стала однією з таких галузей завдяки високій здатності вбудовування в 3D-сітки, які можуть бути чудовими носіями даних. Таблиця. 4 містить додаткові відомості про набори 3D-даних.

Стеганаліз

Більшість методів стеганалізу були сформульовані як проблема бінарної класифікації. Розширений стеганаліз на основі моделі є одним із цих методів, який забезпечує кращу точність виявлення, ніж більшість інших алгоритмів стеганалізу. Метод спочатку витягує різні ручні характеристики з відфільтрованих цифрових зображень на етапі навчання. Потім класифікатор

КПЗ - 2025

ансамблю навчається відрізняти зображення обкладинки від зображень стего. Навчений класифікатор використовується на етапі тестування, щоб визначити, чи містить нове вхідне зображення приховані дані. У стеганалізі з використанням класичного машинного навчання ознаки виділяються ручними методами та відокремлюються від етапу класифікації. Тому точність класифікатора залежить від ефективності методу виділення ознак. Ці витягнуті ознаки передаються на етап класифікації. Навпаки, у глибокому навчанні етап виділення ознак змішується з етапом класифікації, а рішення класифікатора використовується для оновлення вилучених ознак. Багато методів виділення ознак використовуються в стеганалізі в машинному навчанні. Однак велика кількість функцій у зображеннях викликає прокляття розмірності (CoD) і часову складність, особливо під час роботи над універсальним стеганалізом. Виділення дискримінаційних ознак може допомогти підвищити точність стеганалізу. Попередні дослідження запропонували деякі алгоритми для зменшення розмірності даних. Застосування вибору підмножини ознак у контексті стеганалізу дає багато переваг, а саме:

1. Точність методів стеганографічного аналізу зображень залежить від чутливих функцій, які можуть виявляти наявність прихованих повідомлень у всіх типах методів стеганографії, і не залежить від великого розмірного набору функцій.

2. Вибираючи життєво важливі ознаки, зайві ознаки видаляються, а дискримінантні ознаки зберігаються для навчання класифікатора.

3. Зменшується час обчислення для етапу виділення ознак і навчання класифікатора. Це допоможе виявити приховані повідомлення в різних програмах реального часу, де важлива безпека.

Новий метод обробки ознак у двох фазах оптимізації був запропонований у [28] . Перша оптимізаційна модель – це власне значення матриці розсіювання в межах класу. На другому етапі оптимізації використовується лінійний дискримінант Фішера (FLD) випадкового підпростору. Кулкарні та Горкар [29] досліджували наявність прихованих шкідливих програм у

КПІ – 2025

зображеннях. Вони використовували PCA на основі власних значень, щоб зменшити кількість вимірів і зберегти всі важливі функції, необхідні для класифікатора. Універсальна техніка стеганалізу зображень, зосереджена на виборі ознак, була представлена Десаї та Пателем [40] . Цей метод є групуванням ознак на основі PCA. Desai та ін. обчислив власне значення коваріаційної матриці, а потім кластеризував ознаку за допомогою методу K-середніх.

Деякі методи можна використовувати для прискорення вилучення ознак. Лі та ін. [41] виявили, що вартість класичного методу «розділяй і володарюй» залежить від оновлення сингулярних векторів, яке включає два матричних множення. У результаті вони дійшли висновку, що сингулярні векторні матриці ламаної матриці є матрицями Коші та мають недіагональні та низькорангові властивості, тому їх можна оцінити за допомогою ієрархічно напівроздільних (HSS) матриць. Вони представили прискорений алгоритм постійного струму, де використовується структурований метод оцінки низького рангу. Їхнє дослідження показало, що АЦП може бути в три рази швидшим за постійний. З іншого боку, Liao et al. [42] запропонував паралельно структурований «розділяй і володарюй», спрямований на зменшення обчислювальних витрат. Їхній метод будує локальні матриці за допомогою генераторів матриць, подібних до Коші, без будь-якого зв’язку, а потім зменшує витрати на обчислення, використовуючи структурований метод апроксимації низького рангу.

Ці методи АЦП показали, що обчислювальна вартість методів суттєво зменшилась і значно допоможе в задачах стеганалізу.

У наступних розділах ми представляємо основний внесок цього огляду, який полягає в тому, щоб висвітлити роботи, які були виконані з використанням класичного машинного навчання та методів глибокого навчання в просторових областях і областях трансформації в області стеганалізу зображень.

Методи стеганалізу двовимірного зображення на основі класичних методів машинного навчання

Було запропоновано різні методи стеганалізу двовимірних зображень з

КПЗ-2025

використанням методів машинного навчання. Ці методи використовують дві фази для вирішення проблеми стеганалізу. Перший етап – це ручне виділення ознак, яке має можливість моделювати вбудовані спотворення в зображенні за допомогою будь-якого стеганографічного алгоритму. Другий етап – це процес класифікації, який використовує інтегрований класифікатор для навчання ознак. Для стеганалізу зображень можна використовувати різні класифікатори, наприклад SVM і ансамблеві класифікатори. У наступних розділах обговорюються методи стеганалізу в просторовій та трансформаційній областях.

Стеганаліз просторової області

Просторова область використовується для простоти реалізації та її високої ємності для прихованої інформації. Деякі методи в просторовій області включають HUGO [27] , схему відповідності пар пікселів [43] , [44] , MiPOD [32] , HILL [31] , конструкцію Гіббса [45] , просторові багатофункціональні моделі (SRM) [46] і WOW [30] .

Метод виявлення стеганографічної відповідності найменш значущих бітів (LSBM) був представлений Pevny та ін. [4] . Цей метод визначає відхилення, спричинені стеганографічним вбудовуванням, шляхом моделювання відмінностей між сусідніми пікселями на зображеннях. Фільтр використовується в стеганалізі для придушення вмісту зображення під час виявлення стегошуму. Ланцюги Маркова першого та другого порядку використовуються для моделювання залежностей між сусідніми пікселями у відфільтрованому зображенні. Потім вибіркова матриця ймовірності переходу використовується для отримання вектора ознак у стеганалізаторі на основі машинного навчання. Хоча представлений набір функцій був розроблений для виявлення стеганографії просторової області, він також міг виявити алгоритми, які приховані в блочній області DCT.

SRM, запропонований Фрідріхом і Кодовським [46], є широко використовуваним сучасним стеганалізатором зображень. SRM виділяє залишкові функції, застосовуючи нелінійні та лінійні фільтри високих частот.

КБПЗ-2025

Модель виявляє стегографічні зображення, реєструючи розривність шаблону шуму в сусідніх пікселях у підроблених і непідроблених областях. Автори довели, що ознака середньої розмірності, яка подається в гаусівську SVM, і ознака високої розмірності, яка подається в класифікатор ансамблю, можуть підвищити точність виявлення для всіх перевірених методів.

Віна та Аріважаган [47] запропонували універсальний кількісний стеганалізатор, використовуючи зменшені екземпляри та ознаки, в яких як локальні, так і глобальні ознаки розглядаються для простору ознак. Глобальні особливості є ознаками спільного виникнення з моделі Маркова, тоді як локальні особливості складають модель локального фільтра (LFP) [48] . Витягнуті характеристики об'єднуються за допомогою жадібної процедури рандомізованого адаптивного пошуку (GRASP) [49] . Потім дискретизований весь конденсований найближчий сусід (D-AllCNN) застосовується для зменшення екземпляра, а RFE застосовується для зменшення розмірності функції на основі принципу розділяй і володарюй. Оцінювач AdaBoost із деревами регресії використовується для прогнозування корисного навантаження в стегографічному зображенні. Запропонований сліпий кількісний стеганалізатор підходить для просторових методів на основі LSB і може бути використаний для вдосконалення існуючих мультимодельних стеганалітичних функцій. Оскільки алгоритми стеганографії негативно впливають на кореляції між градієнтними амплітудами колірних каналів, Kang et al. [40] представили метод стеганалізу з використанням кореляції амплітуди градієнта каналу для кольорових зображень. Витягнуті ознаки являють собою матрицю спільного повторення із залишків амплітуди градієнта, які описують кореляцію різних колірних каналів, а потім ці характеристики поєднуються з існуючими ознаками, як у [41] і [42] для стеганалізу кольорового зображення. Запропонований метод перевірено на базі даних BOSSbase. Розмір функцій становить 5404, що займає багато місця та часу, коли функції витягуються та зберігаються. Через високу розмірність функцій стеганалізу запропонований алгоритм використовує ансамблевий класифікатор, який є

КПЗ-2025

поширеною технікою навчання для стеганалізу зображень. Щоб покращити виявлення збурень локальних візерунків у стегографічних зображеннях, функції використовуються в задачах класифікації текстур, таких як LBP. Особливості LBP можуть характеризувати локальні зміни структури, і вони здаються багатообіцяючими. LBP може ефективно узагальнювати локальні структури зображення шляхом порівняння пікселів з їхніми сусідами. Натхненні цією ідеєю Gui et al. [43] запропонував витягти багатомасштабні інваріантні обертання LBP з гладких пікселів як унікальних текстурних особливостей, які потім подаються в лінійний SVM. Експериментальні результати показали, що метод добре показав стегографічні зображення та мав високу точність.

Лю та ін. [44] представили метод стеганалізу сліпого зображення, заснований на методі вибору природних ознак. Функції витягуються для стеганалізу зображень за допомогою спаму. Потім ідеальна підмножина ознак вибирається з вихідних ознак за допомогою бінарного методу кажанів (BBM) [45] . Класифікатори, які використовуються для перевірки запропонованого методу: KNN, RF, AdaBoost, DCA, NB і SVM. Запропонований метод перевірено з використанням набору даних BOSSBase v1.01, і точність склала 68,08% з класифікатором SVM.

Стеганаліз домену трансформації

Домен трансформації вбудовує приховані повідомлення в коефіцієнти зображення обкладинки. Таким чином, домен перетворення має перевагу над просторовим доменом, де на приховані повідомлення в домені перетворення не впливає обробка зображення, стиснення або кадрування. Методи в області трансформації включають UED [37] , UERD для стеганографії JPEG [3] , статистичні особливості контурного перетворення [46] і стеганаліз зображень на основі блоків на основі DCT і марковських ознак [47] . Ці методи стеганографії залишають мінімальні сліди прихованих даних, тому для переходу до наступного етапу необхідно виділити із зображення незалежні характеристики. Таким чином, ефективні функції для процесу стеганалізу включають ймовірності марковського

КОПІЯ 2025

переходу пікселів, гістограму залишків, матриці спільного повторення, оператори LBP тощо. Наступною фазою є процес класифікації, у якому використовуються інтегровані класифікатори для навчання ознак. Класифікатори, які можна використовувати для стеганалізу зображень, включають SVM і ансамблеві класифікатори.

Лю та ін. [48] представив новий метод, заснований на аналізі функцій, домені DCT і SVM для стеганалізу зображень JPEG. Вони витягли характеристики, використовуючи як внутрішньоблочну, так і міжблокову щільність сусідніх з'єднань з коефіцієнта DCT; потім вони передали ці функції в SVM для виявлення. Щоб передбачити приховану кількість у стеганографії JPEG, автори застосували нейронно-нечітку систему висновку. Їхні експериментальні результати показали, що їхній метод працює краще, ніж добре відомий метод, заснований на марковському процесі.

Холуб і Фрідріх запропонували новий набір функцій для стеганалізу JPEG під назвою дискретне косинусне перетворення залишку (DCTR) [49] . Ці ознаки мають низьку складність і малий розмір, і вони створюються як гістограми залишків, отриманих за допомогою 64 баз DCT. Автори використали лінійний дискримінант Фішера (FLD) [10] як бінарний класифікатор. Результати показують, що DCTR досяг конкурентоспроможних виявлень у багатьох методах JPEG.

Пісня та ін. [2] запропонував метод стеганалізу шляхом застосування 2D-фільтрів Габора для фази виділення ознак для виявлення вбудованих змін, обмежених у складних областях текстури зображень JPEG. Результати показали, що запропоновані функції для стеганалізу зображень отримали конкурентоспроможні характеристики в порівнянні з досягнутими функціями стеганалізу UED [37] , J-UNIWARD [29] і SI-UNIWARD [29] .

Shankar і Azhakath [3] запропонували сліпий стеганаліз зображення на основі функцій для формату файлу JPEG. Загалом було виділено 274 ознаки зображення за допомогою DCT з першопорядкових (подвійні, глобальні та

КОПІЯ 2025

індивідуальні гістограми), другого порядку (сумісність, дисперсія та блочність) і спеціальних марковських ознак. Класифікаторами були SVM і SVM-particle swarm optimizations (SVMPSO). Класифікатори були адаптовані з 10% вбудовуванням і 10-кратною перехресною перевіркою. Ядра, використані в процесі класифікації, були лінійними, багатоквадратичними, Епанечникова, радіальними, поліноміальними та ANOVA. Двома наборами даних зображень, використаними для запропонованого сліпого стеганалізу, були свята INIRA [4] і UCID [4] . Було показано, що класифікатори PSO досягли кращої продуктивності, ніж SVM, для всіх ядер і зразків.

Лю та ін. [15] представили структуру, засновану на злитті класифікаторів SVM; він складається з трьох етапів: навчання підкласифікатора, навчання класифікатора з'єднання та тестування класифікації з'єднання. Автор використав багаті модельні ознаки, запропоновані в [16], які розділені на різні групи на основі кореляційних ознак. Класифікатор Fusion може вивчати кореляцію результатів виявлення для підкласифікаторів, і точність підвищується, коли класифікатори збільшуються. Лу та ін. [17] представив структуру вдосконалення для стеганалізу на основі вибору ознак і попередньої класифікації. Функції виділяються за допомогою аналізу залежностей суміжних даних зображення. Алгоритми K-середніх застосовуються для попередньої класифікації зображень, які мають різну складність вмісту та текстури з набору даних зображень. Потім були обрані оптимальні характеристики з кожного кластера для остаточного рішення, спрямованого на покращення загальної продуктивності стеганалізу. Shankar і Azhakath [18] дослідили чотири ознаки вилучення для стеганалізу, які були першого порядку, розширеного DCT, другого порядку та ознаки Маркова. Вони використовували метод LSBM [23] і F5 [20] для просторової області та області перетворення відповідно. Вони використовували шість різних ядер і чотири види вибірок SVM із перехресною перевіркою. Їхнє дослідження прийшло до висновку, що домен трансформації забезпечує кращу точність класифікації, ніж просторовий домен. Таблиця. 5 надає більш детальну

КПЗ-2025

інформацію про роботу, виконану в просторовій та трансформаційній областях.

Методи стегааналізу тривимірного зображення на основі класичних методів машинного навчання

У цьому розділі ми описуємо деякі алгоритми стеганалізу, які використовувалися для 3D-зображень. Оскільки метою 3D-стеганалізу є пошук прихованих даних у 3D-зображеннях, це складна проблема порівняно зі стеганалізом 2D-зображень, оскільки 3D-зображення є складними 3D-об'єктами, які мають довільну топологію та неправильну геометрію.

Янг і Іврісімтзіс [25] представили перші функції 3D стеганалізу (YANG208) для виявлення прихованих повідомлень у трикутних сітках. Для кожної сітки вони розрахували характерний вектор ознак, який фіксував геометричну інформацію з її декартових і лапласівських координат. Потім вони застосували техніку калібрування до виділеного вектора ознак шляхом обчислення різниці між сіткою та еталонною сіткою для виділення дискримінаційних ознак. Витягнуті ознаки потім вводили в метод навчання під наглядом на основі квадратичного дискримінантного аналізу (QDA). Метод був перевірений на шести відомих стеганографічних системах і показав задовільні показники точності.

Лі та Борс у [26] запропонували метод (LFS52), який витягнув 52-D локальний вектор ознак для задачі 3D стеганалізу. 52-D вектор ознак об'єднав три компоненти: 40-D вектор ознак, що складається з найефективніших функцій у YANG208 [25] , 4-D вектор нормальних ознак вершини та 8-D вектор ознак локальної кривизни форми. Комбіновані характеристики використовувалися як вхідні дані для ансамблю FLD і квадратичного класифікатора, щоб відрізнити 3D стегооб'єкти від обкладинок. Запропонований метод був протестований на наборі даних PSB, де стегооб'єкти створювалися за допомогою двох різних методів стеганографії, які приховують повідомлення в 3D-об'єктах. Результати показали, що метод забезпечує кращу продуктивність для процесу тривимірного стеганалізу, де локальні кривизни форми та нормалі вершин мають кращу

КПЗ-2025

здатність розрізняти.

Лі та Борс [27] запропонували алгоритм вибору ознак на основі надійності та релевантності (RRFS) як вирішення проблеми невідповідності покриття-джерело в 3D стеганалізі. Набір функцій (LAY252) витягується за допомогою комбінації функцій LFS52 [26] і YANG208 [25] . Запропонований алгоритм відбору вибирає ознаки на основі їх надійності та кореляції. Вибрані функції вводяться в ансамбль FLD. Запропонований алгоритм вибирає кращі функції, ніж інші алгоритми. Однак цей алгоритм обмежений набором перетворень у проблемі неузгодженості покриття-джерело.

Кім та ін. [28] запропонував локальний набір функцій (LFS64). Вони використовували середнє значення, загальну кривизну та нормаль краю на додаток до особливостей, представлених у [25] та [26] , і вони відобразили особливості за допомогою однорідної карти ядра, щоб допомогти класифікатору ансамблю FLD виявити сітки setgo. Запропонований метод перевершив LFS52 [26] .

Лі та Борс [29] запропонували метод (LFS76), розширений до функцій LFS52 [26] для визначення невеликих відмінностей між обкладинкою та стегографічними 3D-графічними об’єктами. Запропонований метод виділяє та поєднує різні тривимірні характеристики, такі як нормаль до вершини, локальна кривизна та локальне геометричне представлення вершини у сферичних координатах. Статистика наборів витягнутих ознак із вектором ознак 76-D подається в класифікатор SVM, ансамбль FLD і QDA. Автори використовували набір даних PSB, який містить 354 тривимірні сітчасті об’єкти покриття. Стегооб'єкти були створені за допомогою трьох різних стеганографічних методів приховування інформації. Експериментальні результати показали, що ансамбль FLD забезпечив найкращі результати для процесу стеганалізу, коли стеганографічний метод водяних знаків на основі середнього [2] використовувався для ідентифікації інформації, вбудованої в 3D-об’єкти.

КПЗ – 2025

Лі та ін. [30] запропонував техніку виділення 3D-об’єктів, яка використовує крайові вектори для захоплення локальних об’єктів, що призводить до 124-D вектора об’єктів (LFS124). Абсолютні різниці між довжинами ребер тривимірних компонентів вектора були обчислені в декартовій системі координат. Потім було обчислено норму різниці між двома векторами та визначено дві різні ознаки, отримані з абсолютних різниць і кута між ними. Нарешті, шість об’єктів були обчислені таким же чином у лапласівській системі координат, усі з яких разом утворили 12 об’єктів. Потім нещодавно витягнутий набір функцій об’єднали з існуючим набором функцій LFS76, щоб отримати вектор ознак 124-D. Ці функції були введені в ансамбль FLD. Запропонований метод перевірено на 354 об’єктах тривимірної сітки покриття з набору даних PSB. Тривимірні стего-сітки були виготовлені за допомогою шести методів приховування тривимірної інформації. Експеримент показав ефективність запропонованого методу в реалізації та зробив висновок про те, що крайовий вектор відіграє значну роль у стеганалізі.

Чжоу та ін. [45] запропонував спеціальний метод стеганалізу з використанням функції, націленої на перетворення PCA, щоб розрізнити стего та об’єкти 3D-сітки. Матриця трансформації стего-сітки близька до матриці ідентичності після перетворення PCA, тоді як матриця трансформації сітки-покриття в більшості випадків далека від матриці ідентичності. Одновимірна ознака визначається нормою між двома матрицями перетворення. Цей метод перевірено на наборах даних PMS і PMN. Запропонований метод стеганалізу був ефективним лише для стеганографічних методів, заснованих на перетворенні PCA.

Чжоу та ін. [31] представив 3D-стеганалітичну схему (NVT+), використовуючи модель тензорного голосування, яка збирає контекст локальної форми, щоб відрізнити стегооб’єкт 3D-сітки від об’єкта покриття. Спочатку було виконано три нормальних тензора голосування з різними визначеннями сусідів. По-друге, три власні значення були обчислені з кожного тензора, де абсолютне

КОПІЯ 2025

значення різниці між власними значеннями розглядалося як характеристика. Три тензорні моделі, кожна з яких виділяє три різниці власних значень, дали дев'ять ознак. По-третє, кілька статистичних моментів ознак, оброблених за допомогою нелінійного відображення, були виділені для формування 36 ознак. 36 отриманих характеристик були об'єднані з характеристиками методу LFS64 в [28] для отримання 100-D вектора ознак. Об'єднаний набір функцій було введено в ансамбль FLD для класифікації. Запропонований метод перевірено на наборах даних PMS та PMN. Експеримент показав, що запропонований метод підвищує ефективність виявлення. Однак час і складність цього методу дуже високі через обчислення кожної характеристики для суміжної грані.

Лі та Борс [32] запропонували WFS228, новий набір функцій стеганалізу 228-D, отриманих за допомогою тривимірного вейвлет-аналізу з багаторазовою роздільною здатністю [33] . Функції витягуються з перетворень між вхідною сіткою та її відповідною вищою та нижчою роздільною здатністю. Для вхідної сітки її відповідна вища та нижча роздільна здатність графіка обчислюється за допомогою тривимірного алгоритму вейвлету. Метод було навчено за допомогою ансамблю FLD, і експерименти показали, що функція тривимірного вейвлета забезпечила найкращу продуктивність для завдання стеганалізу. Таблиця. 6 містить більш детальну інформацію про роботу, виконану в стеганалізі 3D-зображень.

Методи стегааналізу двовимірного зображення на основі глибокого навчання

Протягом останніх кількох років глибоке навчання широко використовувалося в стеганалізі для виділення відповідних ознак для класифікації. Згорткові нейронні мережі (CNN) покращили продуктивність стеганалізу; однак обсяг пам'яті та вартість обчислювальної складності моделей все ще є перешкодами через велику кількість навчальних даних. У цьому розділі ми представляємо моделі глибокого навчання, спрямовані на зниження вартості навчання шляхом вилучення ключових функцій.

КОПЗ-2025

Ghosh та ін. [35] представив модель ШНМ, нову гібридну глибоку нейронну мережу ШНМ, засновану на власних значеннях (точніше PCA) і характеристиках Хараліка. Вони обчислили матрицю спільного появи вхідного зображення в градаціях сірого для чотирьох напрямків пар пікселів, а потім обчислили середнє значення. Потім застосовується двовимірне зменшення: PCA і Haralick. Їхній метод був багатообіцяючим і досяг підвищеної точності. Занг та ін. [36] запропонував вилучення ключових особливостей текстури за допомогою локального шару гістограми, який можна вивчати на основі багатоквадратичного моделювання ядра. Рівень гістограми використовував дві згортки, щоб дізнатися центр і ширину контейнера. Вони використовували нейронну мережу RBF для оновлення центру біну та ширини моделі, а власні значення використовувалися для визначення мінімального та максимального значень RBF. Метод показав значне покращення класифікації текстури. Абазар та ін. [37] представили нову структуру для зменшення вартості навчання за допомогою техніки «розділяй і володарюй». Набір даних розбивається на п’ять непересічних кластерів за допомогою k-середніх. Кожен кластер подається в окрему CNN. Мережі об’єднані, використовуючи швидкий процес зважування. Запропонована модель здатна зменшити розмір навчальних даних для кожної моделі. Експериментальні результати показали, що запропонована структура зменшує часову складність, зберігаючи при цьому точність.

У наступних розділах наведено короткий виклад найсучасніших робіт, які були виконані у стеганалізі двовимірних зображень із використанням методів глибокого навчання в просторовій та трансформаційній областях.

Стеганаліз просторової області

Як ми згадували раніше, у стеганографії просторової області біти корисного навантаження приховані в зображенні обкладинки шляхом зміни значень інтенсивності пікселів безпосередньо в просторовій області. Знаючи це, дослідники почали використовувати переваги застосування глибокого навчання для стеганалізу просторової області. Перша спроба використати

КОПІЯ 2025

неконтрольований метод глибокого навчання для стеганалізу була здійснена Таном і Лі [38] . Автори використовували стекові згорточні автокодери (SCAE) [39] . Вагові значення ядер і фільтрів у CNN були ініціалізовані випадковим чином. Автори вважали, що добре навчена CNN має працювати порівнянно з добре відомою та успішною SRM. Вони використовували дев'ятишаровий триступеневий CNN на основі сліпого стеганалізатора.

Qian та ін. [21] були першими, хто запропонував використовувати контрольоване навчання з CNN для стеганалізу. Їхня мережа складається з трьох етапів: високочастотного фільтра, який використовується як рівень попередньої обробки, згорткового рівня для виділення ознак, а потім повністю підключеного рівня для класифікації. Використовується шар високочастотного фільтра, оскільки стегограма має слабший сигнал, ніж вміст зображення. Ця модель досягла розумних результатів порівняно з традиційними моделями, які використовують функції ручної роботи. Ву та ін. [40] запропонував нову структуру вилучення ознак, яка може вивчати спільні ознаки з вхідних зображень та їхніх відповідних залишкових зображень. Процес об’єднання функцій у CNN абсолютно не контролюється. Щоб мінімізувати розмірність даних, метод вибирає карти функцій із середніх трьох прихованих шарів і об’єднує їх у 1D-вектор, який передається в повністю зв’язані шари для отримання результату класифікації. Мета полягає в тому, щоб зменшити негативний вплив фільтра високих частот, щоб гарантувати, що мережа залишається конвергентною.

Rezaei та ін. [17] протестували понад 40 архітектур CNN і виявили, що найкраща форма складається з двох згорткових шарів, за якими слідують три повністю з’єднані шари. Вхідне зображення CNN спочатку фільтрується високочастотним фільтром, як це зроблено в роботі Qian et al. [21] . CNN оцінюється за двома сценаріями, перший з яких є сценарієм ясновидіння, у якому передбачається, що той самий ключ вбудовування застосовано до різних зображень. Автори порівняли цей сценарій з ансамблевим класифікатором із функціями SRM і виявили, що CNN зменшив помилки класифікації втричі. У

КПЗ-2025

другому сценарії було припущено невідповідність обкладинки та джерела, за якої модель джерела, що використовується в стеганографії, відрізняється від моделі джерела, яка передбачається для стеганалізу. Помилки класифікації зменшилися порівняно з багатими моделями та класифікатором ансамблю.

Сю та ін. [41] запропонували використовувати CNN зі статистичним моделюванням, щоб уникнути конвергенції мережі. Вони використали високочастотний фільтр як шар, щоб отримати залишковий шум вихідних зображень, а потім подали їх у п’ять шарів згортки та об’єднання. 128-вимірні об’єкти подаються на повністю підключений рівень, а потім на шар softmax для класифікації вхідних даних. Основний внесок цієї техніки полягає в тому, що вона використовує абсолютний шар (ABS) після шару згортки для отримання позитивних значень. Потім вихідні дані передаються на рівень пакетної нормалізації, щоб гарантувати, що мережа не застрягла в локальних мінімумах. Нелінійна функція активації гіперболічного тангенсу (tanH) використовувалася в першій групі шарів згортки, а випрямлені лінійні одиниці (ReLU) використовувалися в інших шарах. Автори навчили свою модель CNN за допомогою міні-пакетного градієнтного спуску, і результати перевершили традиційний ансамблевий класифікатор SRM.

Є та ін. [42] представив YeNet, який має нову усічену лінійну одиницю (TLU), у моделі стеганалізу CNN. Мережа містить 10 згорткових рівнів, а 30 високочастотних ядер були ініційовані за допомогою SRM і використані як рівень попередньої обробки. У першому шарі згортки автори використовували TLU, а в решті шарів вони використовували функцію активації ReLU. Результати 144-вимірних векторів ознак подавалися в один повністю зв’язаний шар, за яким ішов шар softmax. YeNet досягнув нижчого рівня помилок виявлення в порівнянні зі стеганалізаторами SRM і maxSRMd2.

Yedroudj та ін. [43] представив модель CNN, включивши один рівень попередньої обробки, що складається з 30 високочастотних шарів із ядер SRM, за якими слідують п’ять згорткових шарів і, нарешті, один шар softmax у

КОПЗ_2025

просторовій області. Їхня модель CNN схожа на мережу Сю [41] та мережу Є [42] . YedroudjNet використовував пакетну нормалізацію та шари ABS як мережу Xu [41] , але вони використовували більш дрібні шари згортки порівняно з Ye's Net [42] . Нарешті, Yedroudj та ін. використовували три повністю з'єднані шари.

Таким чином, дослідження, проведені в рамках глибокого навчання, дійшли висновку, що врахування знань про предметну область у стеганалізі може покращити продуктивність CNN. Оскільки CNN використовують етап виділення функцій, під час проектування мережевих архітектур слід враховувати знання домену.

Стеганаліз домену трансформації

Підходи стеганографії до домену JPEG працюють у домені перетворення шляхом зміни коефіцієнтів, отриманих після застосування DCT. У минулому більшість методів стеганалізу JPEG витягували функції з розпакованих зображень. Проте зараз дослідники мотивовані вивчати стеганографічні алгоритми JPEG за допомогою CNN.

Xu [44] перетворив вхідні зображення JPEG у просторову область і подав їх у набір заданих фільтрів DCT розміром2 × 2 ,3 × 3 ,4 × 4 ,5 × 5 , і8 × 8 як етап попередньої обробки. Найкращий результат досягається, коли а4 × 4 використовується фільтр. Ці функції були використані в архітектурі CNN, що складається з 20 шарів згортки з пакетною нормалізацією та рівнями функцій ReLU. Вихідні дані – 384 вектори ознак – були подані в повністю зв'язаний шар, а потім у шар softmax. Результати показали, що запропонована мережа CNN мала меншу помилку класифікації.

Чен та ін. [45] розробили нову функцію визначення фази JPEG із двома архітектурами CNN для підвищення точності виявлення. Фаза JPEG – це статистична властивість, зібрана за допомогою8 × 8 околиці пікселів окремо шляхом отримання залишків шуму. Їхня модель CNN спиралася на модель Сю [41] ; однак вони включили визначення фази в XuNet [44] і вимкнули рівень

КБПЗ – 2025

об’єднання з перших двох рівнів. Кожна карта ознак, отримана від другого шару, виділяється на 64 підрешітках, а потім використовується в шарі з розділеним фазою. Залежно від поділу фаз вони реалізували дві мережі, які називаються PNet і VNet. У PNet вихідні карти об’єктів мають розмір16 × 16 поділені на 64 групи, що призводить до 16 карт функцій. Таким чином, кожна група має свої специфічні рівні обробки. Отримані карти об’єктів потім об’єднуються, щоб сформувати вектор об’єктів розміром 8192 D. (Цей підхід не виконується у VNet). Кінцевий вихідний вектор має розмірність 512. Експеримент показав, що PNet перевершує VNet.

Zeng та ін. [46] використовував гібридну структуру глибокого навчання для найсучасніших підходів стеганографії JPEG, J-UNIWARD [29] , UED [39] і UERD [17] , а також використовував ручні фази квантування та скорочення (Q & T) насичених моделей із CNN. Модель CNN має два етапи. На першому етапі, 25,5 × 5 База DCT була використана для обчислення 25 залишкових карт із нестиснутих і неурізаних зображень JPEG. Потім ці карти були передані на три фази Q & T. На другому етапі три вихідні сигнали фази Q&T подавалися в три незалежні субCNN. Карти вихідних ознак від кожної subCNN були зведені, і для кожної subCNN був отриманий вектор із 512 ознаками. Остаточний вихідний об’єкт довжиною 1536 було подано на чотири повністю з’єднані шари. Модель CNN було навчено за допомогою стохастичного градієнтного спуску (SGD).

Юсфі та ін. [47] виграв конкурс ALASKA steganalysis у 2019 році, використовуючи SRNet [48] для навчання різних комбінацій трьох вхідних каналів, яскравості Y і кольоровості Cr і Cb. SRNet використовував залишкові з'єднання пропуску, а розмір фільтра був3 × 3 . Усі згорткові шари супроводжувалися пакетною нормалізацією та функцією активації ReLU. Перші вісім згорткових рівнів не включали рівень об’єднання, оскільки передбачається, що середнє об’єднання є фільтром низьких частот, тоді як стеганаліз стосується вмісту високих частот, де знаходяться стегодані. Вихідні дані цих згорток подавалися на повністю пов’язаний рівень, який виробляв два виходи, і подавався

КОПІЮЄ 2025

на бінарний класифікатор.

Надихнувшись ідеєю використання передачі навчання нейронних мереж перед навчанням для непов’язаних завдань і вдосконалення стеганалізу, Юсфі та ін. [49] досліджували попередньо підготовлені мережі глибокого навчання, такі як EfficientNet [40] , MixNet [41] і ResNet [42] для стеганалізу. Вони прийшли до висновку, що видалення об'єднання та кроку в перших шарах дозволило підвищити продуктивність. Xiancheng Wu та ін. [43] досліджували ефекти застосування стиснення у восьмибітних обчисленнях і квантуванні з плаваючою комою в XuNet. Модель досягла вищої точності, ніж модель Сю. Їхні результати показали, що дві моделі CNN, засновані на схемах квантування, корисні в стеганалізі.

У стеганалізі двовимірних зображень із використанням класичного машинного навчання SVM і SRM є найпопулярнішими бінарними класифікаторами, тоді як FLD є найпопулярнішим ансамблевим класифікатором для стеганалізу тривимірних зображень. У глибокому навчанні дослідники зазвичай використовують 2D архітектури CNN для реалізації моделей стеганалізу для стеганалізу зображень. Відомо, що 3D-сітки мають більшу здатність до вбудовування, ніж 2D-зображення. Однак багато досліджень стеганоаналізу спрямовані на 2D-зображення. Тому важливо дослідити можливість виявлення тривимірної сітчастої стеганографії за допомогою методів глибокого навчання.

КОПЗ-2025

3.3 Розробка функціональної схеми

Функціонально система дозволяє реалізувати різні методи стегааналізу. Нижче розглянуто основні методи стегоаналізу, дослідження яких взяте з [15].

Візуальні методи стегоаналізу

Візуальні методи засновані на здатності зорової системи людини аналізувати графічні образи та виявляти відмінності в порівнюваних зображеннях. Візуальні атаки ефективні при повному заповненні контейнеру: чим

менше заповнений контейнер, тим складніше виявити факт прихованого повідомлення. Але частіше досліджується не саме зображення, а його бітові зрізи, тому що відмінності між порожнім та заповненим контейнером як правило візуально не проявляються. Якщо розглянути бітовий зріз, що містить найменші значимі біти, в деяких випадках можна побачити сліди прихованого повідомлення.

Статистичні методи стегоаналізу

Найбільш поширені та різноманітні методи статистичного стегоаналізу. Статистичні методи базуються на понятті «природного» контейнера. Суть методів полягає в оцінюванні ймовірності існування стегоповідомлення на основі критерію оцінки близькості досліджуваного контейнера до «природного». Основним недоліком методів цього класу є саме припущення про існування «природного» контейнера. Основні методи статистичного стегоаналізу найбільш повно розглянуті в [8], серед важливих методів не згадуються там лише RS-аналіз, огляд та дослідження якого можна зустріти в [4, 5, 6] та «аналіз пар», який розглянуто в [6,7].

Розглянемо методи описані [8] та розташовані у порядку убування ступеня довіри позитивним результатам їхнього застосування (виявлення прихованої інформації).

Метод оцінки числа переходів значень молодших біт у сусідніх елементах зображення

У методі використовується знання, що між молодшими бітами сусідніх елементів і між ними й іншими бітами природних контейнерів є кореляційні зв'язки. При аналізі графічних файлів формату BMP як елементи аналізованої послідовності вибираються найменш значущі біти складових кольору поруч розташованих пікселів зображення. При дослідженні файлів формату JPEG – молодші біти сусідніх дискретних косинусних коефіцієнтів, відмінних від 0 і 1.

КОПЗ – 2025

Залежність між бітами у відповідних розрядах елементів контейнера має марківский характер. При цьому параметри залежності визначаються номером розряду. Під «переходом» розуміють перехід значення i-го елемента послідовності в значення i + 1 елемента послідовності x, i = 1, 2, ..., n – 1, де n – довжина послідовності. Так як послідовності є двійковими, то аналізується чотири види переходів: з 0 в 0, з 0 в 1, з 1 в 0 і з 1 в 1. За отриманими результатами будується гістограма. Для кожного розряду перший стовбчик гістограми показує число переходів у потоці НЗБ із 0 в 0, другий стовбчик – з 0 в 1, третій стовбчик – з 1 в 0, четвертий стовбчик – з 1 в 1.

Для порожнього контейнера й контейнера, що містить вбудовану інформацію, число переходів у потоці НЗБ буде різним. Розподіл НЗБ стеганоконтейнера має, як правило, випадковий характер. Відповідно число переходів у потоці НЗБ для всіх станів буде приблизно однаковим, що не властиво порожньому контейнеру.

Метод оцінки частот появи k-бітових серій у потоці НЗБ елементів контейнера

Метод дозволяє оцінити рівномірність розподілу елементів у досліджуваній послідовності на основі аналізу частоти появи нулів і одиниць, і серій, що складаються з k біт. У бітовому представленні досліджуваної послідовності x підраховується, скільки разів зустрічаються нулі й одиниці (k = 1), серії-двійки (00, 01, 10, 11: k = 2), серії-трійки (000, 001, 010, 011, 100, 101, 110, 111: k = 3) і т.д. На основі результатів будується гістограма.

Для JPEG-зображень гістограма будується за значеннями частот появи бітових серій у потоці НЗБ дискретних косинусних коефіцієнтів, відмінних від -1, 0, 1.

Для незаповнених BMP і JPEG зображень не є характерним, щоб значення частот всіх компонентів перебували досить близько. При вбудовуванні інформації значення частот зближаються. Цей факт використовується при аналізі.

КБПЗ-2025

Результати роботи методу залежать від стеганографічного перетворення, застосованого для вбудовування приховуваних даних, а також від їхнього обсягу. Як правило, виявлення факту приховування можливе при заповненості контейнера на 60% і вище.

Метод аналізу розподілу пар значень на основі критерію χ^2

У методі використовується аналіз гістограми, отриманої за елементами зображення й оцінка розподілу пар значень цієї гістограми. Для BMP-файлів пари значень формуються значеннями пікселів зображення, для JPEG – квантуємими коефіцієнтами дискретного косинусного перетворення, які відрізняються за молодшим бітом. Молодші біти зображень не є випадковими. Частоти двох сусідніх елементів контейнера повинні перебувати досить далеко від значення частоти середнього арифметичного цих елементів. В «порожньому» зображенні ситуація, коли частоти елементів зі значеннями 2N і 2N + 1 близькі за значенням, зустрічається досить рідко. При вбудовуванні інформації дані частоти зближаються або стають рівними. **Ідея атаки χ^2** полягає в пошуку цих близьких значень і підрахунку ймовірності вбудовування на основі того, як близько розташовуються значення частот парних і непарних елементів аналізуємого контейнера. Особливістю алгоритму є послідовний аналіз усього зображення й, відповідно, нагромадження частот елементів.

Результати роботи методу за критерієм χ^2 значною мірою залежать від способу приховування даних. При послідовному записі в НЗБ елементів контейнера метод забезпечує гарні результати, а при псевдовипадковому виборі молодших біт і розсіюванні повідомлення по всій довжині контейнера метод не спрацьовує.

У роботі [12] автор запропонував «блоковий» варіант даного методу. Від класичного методу він відрізняється тим, що аналізоване зображення розбивається на блоки певного розміру, які можуть як перетинатися, так і не перетинатися, і для кожного блоку розраховуються свої набори частот елементів і свої ймовірності приховування. Крім того, існує можливість вибору окремих

КПЗ-2025

областей зображення для їхнього наступного аналізу. Такий підхід дозволяє виявляти наявність інформації, прихованої псевдовипадковим чином.

Метод аналізу гістограм, побудованих за частотами елементів зображення

Метод дозволяє оцінити рівномірність розподілу елементів аналізованого зображення, а також визначити частоту появи конкретного елемента.

Якщо розкид частот появи елементів у кольорових складових BMP-зображення прагне до нуля, то контейнер містить приховані дані. У протилежному випадку контейнер вважається порожнім.

Для зображень у JPEG-форматі будується гістограма частот квантованих дискретних косинусних коефіцієнтів. Експериментально виявлено, що огибаюча гістограми порожнього зображення має більш гладкий характер у порівнянні з гістограмами зображень, що містять стегоповідомлення. Звичайно, залежно від характеру й ступеня стиснення зображення, гістограми можуть змінюватися – у них можуть з'являтися скачки й провали, але важливо те, що приховування інформації міняє загальний вид гістограм. Більшість стеганографічних програм, що працюють із JPEG, приховують дані в молодші біти дискретних коефіцієнтів, відмінних від 0 і 1. Як наслідок, частоти 0-х і 1-х DCT не змінюються, у той час як всі інші частоти або зменшуються, або збільшуються залежно від алгоритму вбудовування. При значних обсягах приховуваної інформації гістограми часто приймають східчастий характер, що нетипово для звичайних JPEG-зображень.

Метод аналізу розподілу елементів зображення на площині

Метод призначений для визначення залежностей між елементами досліджуваної послідовності.

На площину (поле) розміром $(2^R - 1)(2^R - 1)$, де R – розрядність елемента послідовності, наносяться точки з координатами (x_i, x_{i+1}), x_i – елементи досліджуваної послідовності x, $i = 1, 2, \ldots, n - 1$, де n – довжина послідовності. За отриманою «картиною» проводиться аналіз

КОПЗ-2025

Якщо точки по всьому полю розташовані хаотично, то між елементами послідовності відсутні залежності, що характерно для контейнерів з вбудованими даними. У випадку незаповненого контейнера точки на полі будуть розташовані нерівномірно або утворювати «візерунки».

Метод перевірки розподілу елементів на монотонність

Метод дозволяє оцінити рівномірність розподілу елементів зображення за результатами аналізу довжин ділянок незростання й неубування елементів послідовності.

Досліджувана послідовність x графічно представляється у вигляді слідуючих один за одним непересічних ділянок незростання й неубування елементів послідовності.

Так як статистичні властивості стегоконтейнера близькі до властивостей випадкової послідовності, то ймовірність появи ділянки незростання (неубування) буде тим менше, чим більше його довжина n.

Метод «аналіз пар» [6,7]

Даний метод заснований на пошуку закономірності в ймовірностях появи значень яскравості в природних зображеннях і зображеннях з вбудованим стегоповідомленням. При заміні молодшого біта компонента кольору чергового пікселя зображення на черговий біт попередньо зашифрованого або стиснутого стегоповідомлення (тобто стегоповідомлення, що має властивості псевдовипадкової послідовності), значення яскравості пікселя модифікованого зображення або дорівнює значенню яскравості пікселя контейнера, або змінюється на одиницю з імовірністю ~1/2. Для пошуку слідів вбудовування відбувається аналіз закономірностей у частотах появи «сусідніх» значень яскравості. Такі пари значень («Pair of Values») розрізняються тільки значенням найменш значущого біта.

Значення яскравості, двійкове представлення якого закінчується нульовим бітом 1, називається «лівим» (L), а сусіднє з ним значення яскравості, двійкове представлення якого закінчується одиничним бітом – «правим» (R). Нехай

КПЗ-2025

кольорова гама вихідного контейнера включає 8 кольорів. Отже, при вбудовуванні повідомлення в НЗБ компонента кольору пікселів необхідно досліджувати статистичні характеристики в 4 парах номерів кольору.

Ймовірності появи лівих і правих номерів кольору в природних контейнерах, істотно відрізняються між собою у всіх парах, а в зображенні з убудованим стегоповідомлення ці ймовірності рівні. Це є явною ознакою наявності приховуваної інформації. Ступінь розходження між ймовірнісними розподілами елементів природних контейнерів і зображень із вбудованим стегоповідомлення може бути використана для оцінки ймовірності наявності стегоповідомлення у зображенні. Дану ймовірність зручно визначати з використанням критерію згоди χ^2.

Метод RS-аналізу [4, 5, 6]

Одним з оригінальних методів статистичного стегоаналізу є метод RS, вперше опублікований в 2001 р. колективом учених під керівництвом Дж. Фрідріх. Скорочення в назві розшифровується як Regular-Singular, тобто «регулярно-сингулярний».

Суть методу. Все зображення розбивається на групи по n пікселів $G(x_1, x_2,...,x_n)$ де n парне число, наприклад по 2 пікселя, що перебувають поруч по горизонталі. Для групи пікселів визначається функція регулярності або «гладкості» f(G), в якості такої функції можна вибрати, наприклад, дисперсію значень всередині групи, або просто суму перепадів значень суміжних пікселів. Під значенням пікселя розуміється ціле число від 0 до 255.

$$f(x_1, x_2,..., x_n) = \sum_{i=1}^{n-1} \left| x_{i+1} - x_i \right|. \qquad (3.1)$$

Функція F(x) називається фліпінгом і має властивість $F(F(x)) = x$. Визначаються дві функції фліпінгу – F_1, відповідає інверсії молодшого біта пікселя, і F_2, що представляє собою інверсію з переносом у старший біт (додавання одиниці):

$$F_1: 0 \leftrightarrow 1,\ 2 \leftrightarrow 3,\ \ldots,\ 254 \leftrightarrow 255,$$

КПЗ – 2025

F_2: 255↔0, 1↔2, 3↔4, ... , 253↔254, 255↔0.

При застосуванні фліпінгу до групи одержуємо перетворену групу пікселів. Далі, всі групи пікселів розділяються на класи в такий спосіб:

(1) Регулярні групи: $G \in R \Leftrightarrow f(F(G)) > f(G)$,

(2) Сингулярні групи: $G \in S \Leftrightarrow f(F(G)) < f(G)$,

(3) Невикористовувані групи: $G \in U \Leftrightarrow f(F(G)) > f(G)$.

Метод ґрунтується на статистичному припущенні, що для природного зображення, тобто незаповненого контейнера, характерно наступне:

$$R_M \cong R_{-M} \text{ та } S_M \cong S_{-M}. \quad (3.2)$$

Припущення засноване на тому, що застосування F_{-1} дасть той же розподіл, що й F_1 на зображенні, значення пікселів якого зсунуті на одиницю. Для звичайного зображення співвідношення між групами не повинне істотно змінюватися. Значна розбіжність між значеннями свідчить про застосування LSB-стеганографії для молодших біт зображення.

Розглянемо зміни молодших біт зображення при 100% перезаписі їх бітами повідомлення. Вбудовування випадкового повідомлення довжиною, рівною розміру зображення, призведе до того, що 50% молодших біт будуть інвертовані. Це, у свою чергу зведе до нуля різницю між значеннями R_M і S_M. Однак на R_{-M} і S_{-M} вбудовування повідомлення буде впливати прямо протилежно, і різниця цих величин буде пропорційна ступеню заповнювання контейнера, іншими словами довжині повідомлення.

Припускаючи, що в зображення внесене повідомлення довжиною p біт, і при цьому 50% молодших біт, використаних для запису, будуть інвертовані, значення статистик буде одержане у точці p/2. Потім, якщо інвертувати всі молодші біти зображення й перерахувати статистики, на діаграмі вони будуть відповідати точкам кривих при x = 100-p/2. Повній рандомізації молодшої бітової площини відповідає точка 1/2**.** Тепер, якщо прийняти p/2 за нуль, а 100-p/2 за одиницю, а також використовувати апроксимацію кривих R_{-M} і S_{-M} прямими, а

КПІВ_2025

R_M і S_M параболами, можна вивести квадратне рівняння для знаходження координати точки перетину кривих R_M і S_M:

$$2(d_1 + d_0)x^2 + (d_{-0} - d_{-1} - d_1 - 3d_0)x + d_0 - d_{-0} = 0 .$$

Потім, довжина повідомлення p обчислюється як p = x/(x-1/2). Таким чином, вихідне значення довжини є відповіддю для даного методу.

Для дуже зашумлених і дрібнотекстурованих зображень різниця між кількістю регулярних і сингулярних груп контейнера мала. Відповідно, лінії в RS-діаграмі перетнуться під малим кутом і точність зменшиться.

Методика RS-стегоаналізу точніша для повідомлень, стегоповідомлення-біти яких випадково розміщені в площині стегоконтейнера, чим для повідомлень, вбудованих локально.

Методи стегоаналізу заснованих на стисненні даних

В [9, 10] запропоновано метод стегоаналізу цифрових зображень на основі стиснення даних. В даному методі можуть застосовуватися широко розповсюджені програми-архіватори.

Ідея методу полягає в наступному: потік випадкових даних стискається гірше, ніж потік, де зустрічаються повторювані послідовності. Інформація, що включається в молодші біти контейнера, як правило, попередньо шифрується й, можливо, стискається, тому є псевдовипадковою. Ступінь стиснення контейнерів використовується для визначення наявності в них прихованої інформації.

Формально даний алгоритм виглядає в такий спосіб. Нехай $X = \{x_1, ..., x_N\}$ – послідовність байтів у полі даних зображення BMP, де $|X| = N$ – довжина послідовності.

Послідовність X розбивається на d рівних відрізків, а кожний відрізок позначається X_i, де i = 1, 2, ..., d. Нехай $\psi(X)$ – алгоритм стиснення, застосований до послідовності X. Далі визначається коефіцієнт стиснення відрізка n послідовності X алгоритмом ψ за наступною формулою:

$$f(X, n) = \frac{|\psi(X_n)|}{|X_n|} . \qquad (3.3)$$

КПЗ-2025

Нехай φ(X) – псевдовипадкова зміна молодших біт всіх байтів послідовності X, що подається на вхід програми, а Y = φ(X) – отримана з неї нова послідовність ("заповнений" контейнер). Початкова послідовність X повинна стискатися "сильніше" у порівнянні зі зміненою послідовністю Y.

Якщо відрізок X_i послідовності X містить "приховану" інформацію, то коефіцієнт f(X,i) і відповідний йому f(Y,i) відрізняються несуттєво, і навпроти, "порожня" ділянка стискається краще "заповненої". Для визначення факту вбудовування інформації обчислюється різниця коефіцієнтів стиснення:

$$\delta(X, n) = \left| f(X, n) - f(Y, n) \right|, \tag{3.4}$$

та вибирається граничне значення для величини δ і здійснюється оцінка кількості відрізків, на яких значення величини не перевищує поріг. Якщо таких відрізків більше d/2, то вважається, що вхідна послідовність X містила приховані дані, у протилежному випадку послідовність X вважається "порожньою". Поріг можна варіювати, регулюючи тим самим частоту помилок програми на порожніх і непустих контейнерах.

Методи стегоаналізу заснованих на використанні нейронних мереж

Методи стегоаналізу засновані на використанні нейронних мереж малодосліджені. Вони використовують статистичні методи стегоаналізу в поєднанні з можливостями нейронних мереж до навчання та класифікації. Існуючі методи засновані на алгоритмах навчання з учителем. В якості векторів вхідних даних використовуються частотні форми представлення зображень, отримані за допомогою вейвлет-перетворень.

В [11] було запропоновано використати для стегоаналізу нейронні мережі RBF. Даний метод відноситься до «сліпих» методів виявлення вбудованої інформації. Одним з найважливіших етапів алгоритмів стегоаналізу заснованих на використанні нейронних мереж є вибір ознак за якими нейромережа буде робити висновок про наявність стеговставки. Простір пікселів зображення перетворюється в простір ознак і визначення наявності вбудованого повідомлення відбувається вже в просторі ознак. В [11] як ознаки були

КОПЗ_2025

використані статистичні моменти в частотній області гістограм вейвлет-коефіцієнтів.

Використання статистичних характеристик гістограм зображень пояснюється легкістю моделювання гістограм за допомогою суми, як правило, двох випадкових нормальних змінних і повнотою представлення зображення. В частотній області відмінність стегоповідомлення від контейнера легше розрізнити.

Статистичні моменти в частотній області гістограм визначаються в такий спосіб:

$$M_n = \sum_{k=-N/2}^{N/2} |f_k|^n p(f_k), \tag{3.5}$$

де n – порядок моменту, N – кількість відліків коефіцієнтів дискретного перетворення Фур'є (ДПФ) для гістограми, $f_к$ – k-та частота в ДПФ (k = -N/2,...-1,0,1,..N/2).

$$p(f_k) = \frac{|H(f_k)|}{\sum_{k=-N/2}^{N/2} |H(f_k)|}, \tag{3.6}$$

де, $|H(f_k)|$ – амплітуда ДПФ гістограми $h(x_k)$,

$$H(f) = \int_{-\infty}^{\infty} h(x) e^{-j2\pi fx} dx, \tag{3.7}$$

де h(x) – гістограма зображення, або інакше кажучи, кількість пікселів, що приймають значення x.

Кожний з n компонентів вхідного вектора подається на вхід m базисних функцій RBF-мережі і їхні виходи лінійно підсумовуються з вагами:

$$\{w_j\}_{j=1}^{m}.$$

Вихід RBF-мережі є лінійною комбінацією набору базисних функцій:

КПЗ 2025

$$f(\bar{x}) = \sum_{j=1}^{m} w_j h_j(\bar{x}).$$

Якщо припустити, що параметри функції, зсув *c* і радіус *r* фіксовані, то завдання знаходження ваг вирішується методами лінійної алгебри. Цей метод називається методом псевдообернених матриць і він мінімізує середній квадрат помилки. Суть цього методу полягає в наступному.

Знаходиться інтерполяційна матриця H:

$$H = \begin{bmatrix} h_1(\overline{x_1}) \ldots . h_m(\overline{x_1}) \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ h_1(\overline{x_p}) \ldots . h_m(\overline{x_p}) \end{bmatrix},$$

де m – число нейронів у прихованому шарі, p – розмір навчальної вибірки, n – число входів мережі.

На наступному етапі обчислюється інверсія добутку матриці H на транспортовану матрицю H^T:

$$A^{-1} = (H^T H)^{-1}.$$

Вектор ваг:

$$\overline{W} = A^{-1} H^T \bar{y}.$$

Якщо припущення про фіксовані параметри функції не виконуються, тобто крім ваг необхідно налаштувати параметри активаційної функції кожного нейрона (зсув функції й радіус) і задача стає нелінійною. Вирішувати її доводиться з використанням ітеративних чисельних методів оптимізації, зокрема, градієнтних методів.

Для навчання нейронних мереж в [11] було використано алгоритм навчання з учителем. Навчання без вчителя не було досліджене.

Для адекватного навчання RBF-мережі необхідно підготувати вхідні дані – провести аналіз за допомогою методу головних компонентів і стиснути діапазон по кожній ознаці до інтервалу [0,1].

КОПЗ_2025

Рисунок 3.2 – Функціональна схема системи

Функціональна схема системи зображена на рисунку 3.2. Вона складається з наступних блоків:

1. Інтерфейс користувача системи виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

КБПЗ – 2025

2. Статистичні методи стегоаналізу:

– Метод оцінки числа переходів значень молодших біт у сусідніх елементах зображення.

– Метод оцінки частот появи k-бітових серій у потоці НЗБ елементів контейнера.

– Метод аналізу розподілу пар значень на основі критерію χ^2.

– Метод аналізу гістограм, побудованих за частотами елементів зображення.

– Метод аналізу розподілу елементів зображення на площині.

– Метод перевірки розподілу елементів на монотонність.

– Метод «аналіз пар».

– Метод RS-аналізу.

3. Методи стегоаналізу заснованих на стисненні даних.

4. Методи стегоаналізу заснованих на використанні нейронних мереж.

5. Візуальні методи стегоаналізу.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Головне вікно ПЗ.

– Налаштування ПЗ.

– Налаштування функціоналу ПЗ.

– Параметри зберігання даних.

– Сканування дисків та формування пакетів даних.

– Вибір пакету даних для аналізу.

– Виявлення несанкціонованого витоку даних.

КбПЗ_2025

– Аналіз даних на основі гістограм частот елементів.

– Аналіз даних на основі стиснення даних.

– Аналіз на основі нейронних мереж.

– Результати статистичного аналізу.

– Очищення файлів від вкладень.

– Список файлів зі вкладеннями.

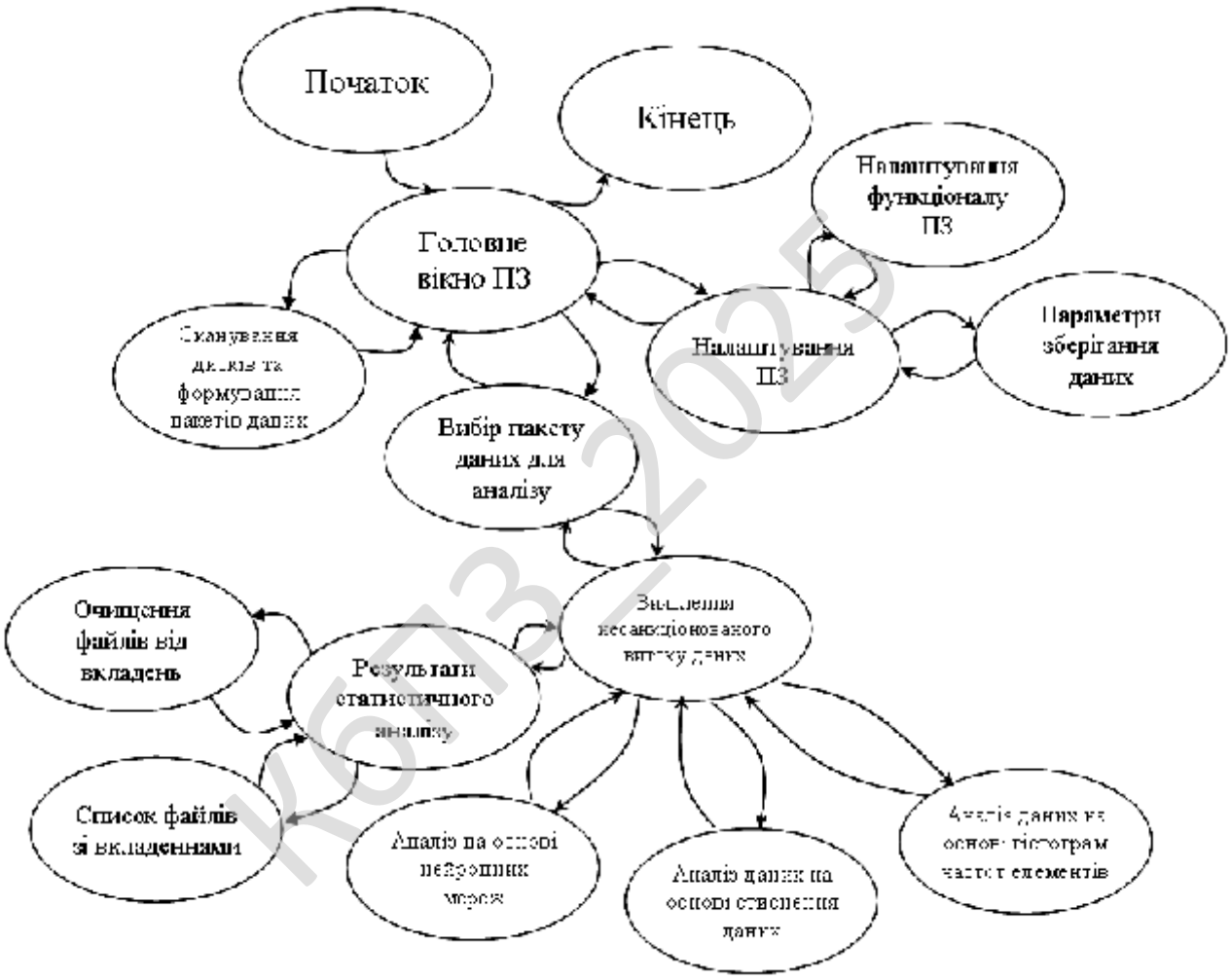

Рисунок 3.3 – Діаграма взаємодії процесів

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

КПЗ 2025

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

КбПЗ_2025

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

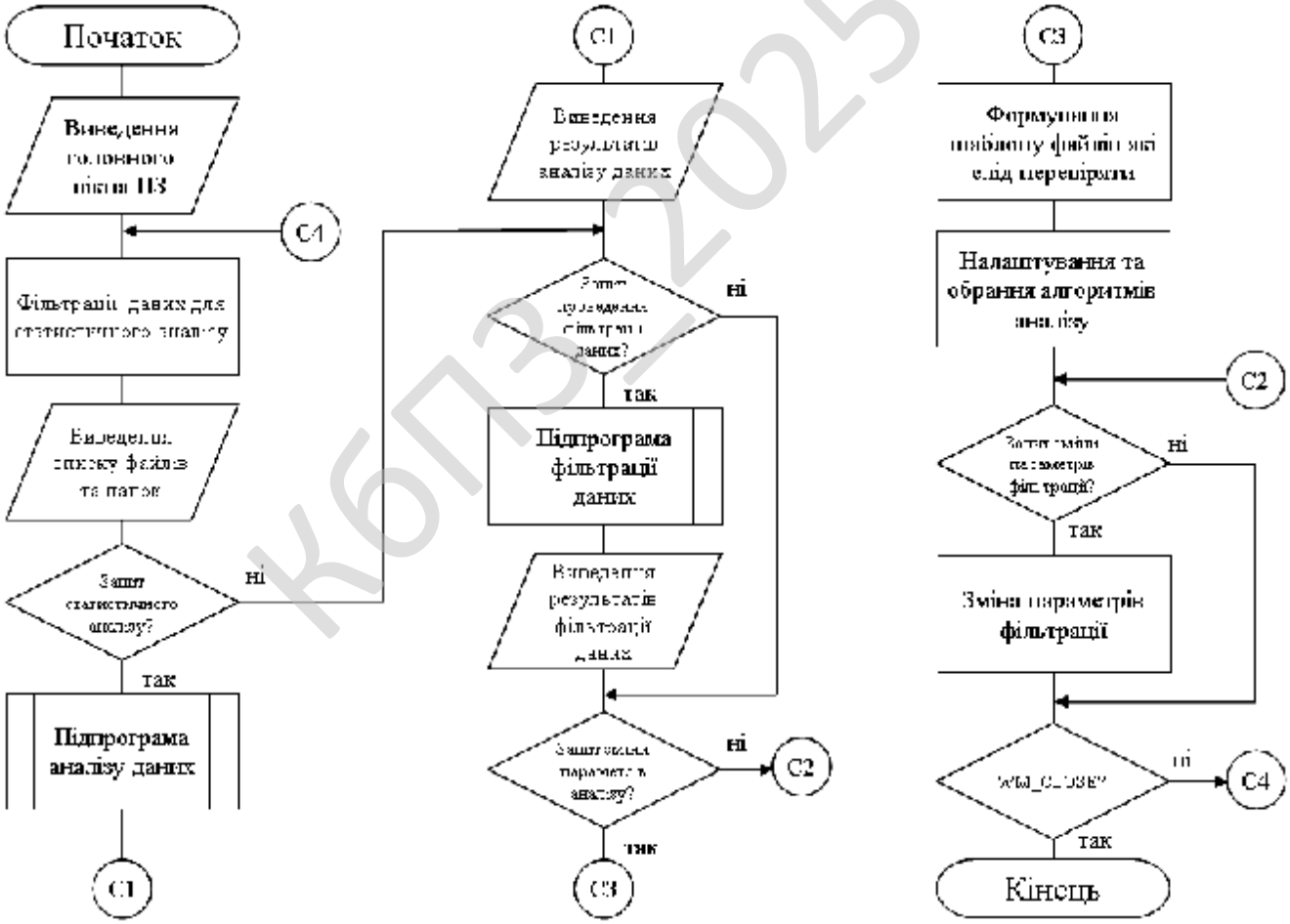

Рисунок 4.1 – Блок схема основної програми

КбПЗ-2025

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

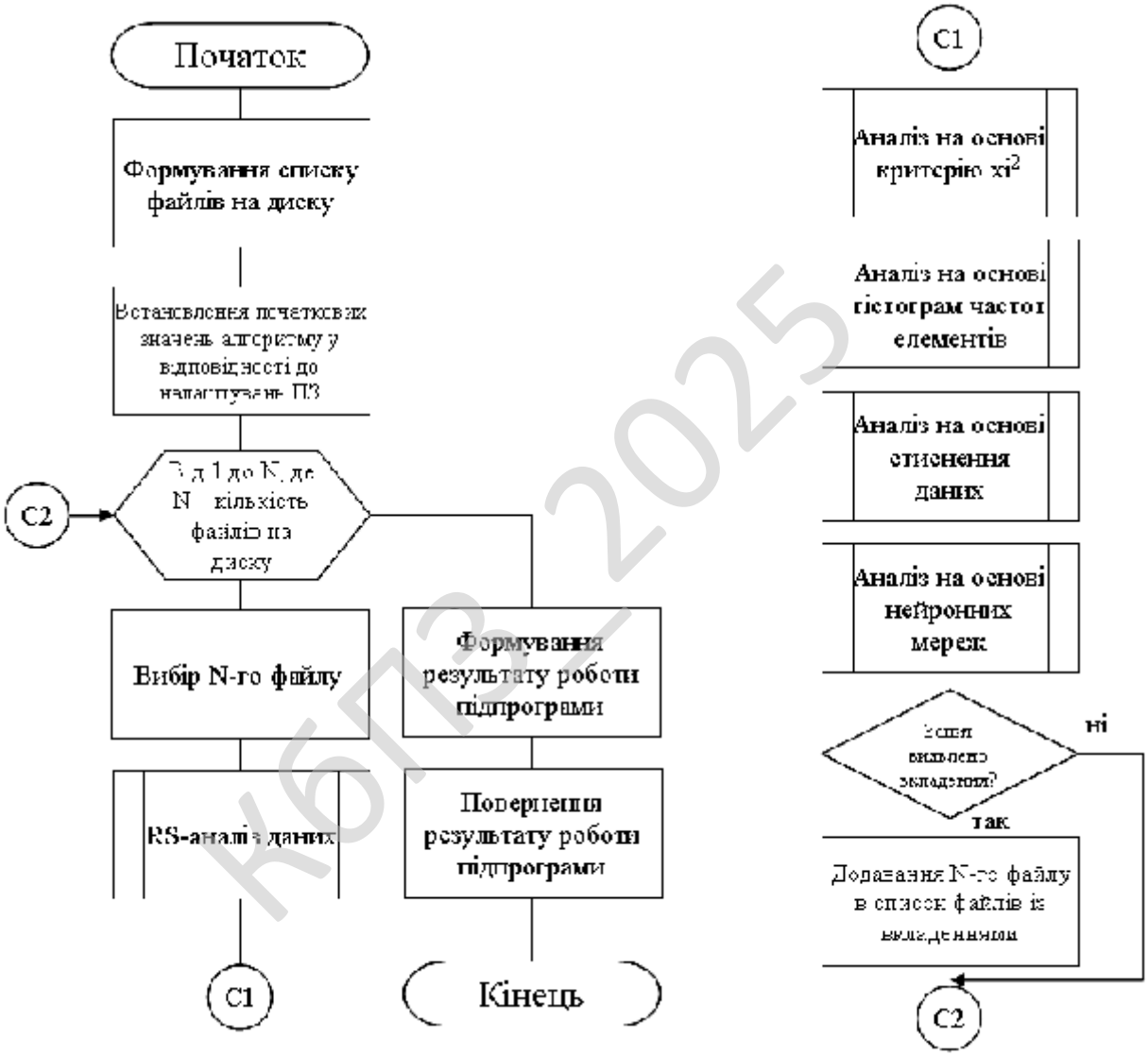

Рисунок 4.2 – Блок схема підпрограм

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

КБПЗ-2025

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю обробки масивів даних.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

Розглянемо роботу процедури статистичного аналізу з використанням нейронної мережі:

```
private static void a1(multilayerperceptron network,
            ref int connidx,
            ref int neuroidx,
            ref int structinfoidx,
            ref int weightsidx,
            int k,
            int nprev,
            int nout,
            bool iscls,
            bool islinearout)
        {
            int i = 0;
            int j = 0;
            int neurooffs = 0;
            int connoffs = 0;
ap.assert((iscls & islinearout) | !iscls, "A1: internal error");
            neurooffs = hlnfieldwidth*neuroidx;
            connoffs = hlconnfieldwidth*connidx;
            if(!iscls)
            {

                //
// регрес нейромережі
```

КбПЗ_2025

```
                //
                for(i=0; i<=nout-1; i++)
                {
                    network.hlneurons[neurooffs+0] = k;
                    network.hlneurons[neurooffs+1] = i;
                    network.hlneurons[neurooffs+2] = structinfoidx+1+nout+i;
                    network.hlneurons[neurooffs+3] = weightsidx+nprev+(nprev+1)*i;
                    neurooffs = neurooffs+hlnfieldwidth;
                }
                for(i=0; i<=nprev-1; i++)
                {
                    for(j=0; j<=nout-1; j++)
                    {
                        network.hlconnections[connoffs+0] = k-1;
                        network.hlconnections[connoffs+1] = i;
                        network.hlconnections[connoffs+2] = k;
                        network.hlconnections[connoffs+3] = j;
network.hlconnections[connoffs+4] = weightsidx+i+j*(nprev+1);
                        connoffs = connoffs+hlconnfieldwidth;
                    }
                }
                connidx = connidx+nprev*nout;
                neuroidx = neuroidx+nout;
                structinfoidx = structinfoidx+2*nout+1;
                weightsidx = weightsidx+nout*(nprev+1);
            }
            else
            {


//
//Класифікация нейромережею
//
                for(i=0; i<=nout-2; i++)
                {
                    network.hlneurons[neurooffs+0] = k;
                    network.hlneurons[neurooffs+1] = i;
                    network.hlneurons[neurooffs+2] = -1;
                    network.hlneurons[neurooffs+3] = weightsidx+nprev+(nprev+1)*i;
                    neurooffs = neurooffs+hlnfieldwidth;
                }
                network.hlneurons[neurooffs+0] = k;
```

КбПЗ_2025

```
                network.hlneurons[neurooffs+1] = i;
                network.hlneurons[neurooffs+2] = -1;
                network.hlneurons[neurooffs+3] = -1;
                for(i=0; i<=nprev-1; i++)
                {
                    for(j=0; j<=nout-2; j++)
                    {
                        network.hlconnections[connoffs+0] = k-1;
                        network.hlconnections[connoffs+1] = i;
                        network.hlconnections[connoffs+2] = k;
                        network.hlconnections[connoffs+3] = j;
network.hlconnections[connoffs+4] = weightsidx+i+j*(nprev+1);
                        connoffs = connoffs+hlconnfieldwidth;
                    }
                }
                connidx = connidx+nprev*(nout-1);
                neuroidx = neuroidx+nout;
                structinfoidx = structinfoidx+nout+2;
                weightsidx = weightsidx+(nout-1)*(nprev+1);
            }
        }
private static void hladdhiddenlayer(multilayerperceptron network,
            ref int connidx,
            ref int neuroidx,
            ref int structinfoidx,
            ref int weightsidx,
            int k,
            int nprev,
            int ncur)
        {
            int i = 0;
            int j = 0;
            int neurooffs = 0;
            int connoffs = 0;
            neurooffs = hlnfieldwidth*neuroidx;
            connoffs = hlconnfieldwidth*connidx;
            for(i=0; i<=ncur-1; i++)
            {
                network.hlneurons[neurooffs+0] = k;
                network.hlneurons[neurooffs+1] = i;
                network.hlneurons[neurooffs+2] = structinfoidx+1+ncur+i;
                network.hlneurons[neurooffs+3] = weightsidx+nprev+(nprev+1)*i;
```

КбПЗ_2025

```
                neurooffs = neurooffs+hlnfieldwidth;
            }
            for(i=0; i<=nprev-1; i++)
            {
                for(j=0; j<=ncur-1; j++)
                {
                    network.hlconnections[connoffs+0] = k-1;
                    network.hlconnections[connoffs+1] = i;
                    network.hlconnections[connoffs+2] = k;
                    network.hlconnections[connoffs+3] = j;
                    network.hlconnections[connoffs+4] = weightsidx+i+j*(nprev+1);
                    connoffs = connoffs+hlconnfieldwidth;
                }
            }
            connidx = connidx+nprev*ncur;
            neuroidx = neuroidx+ncur;
            structinfoidx = structinfoidx+2*ncur+1;
            weightsidx = weightsidx+ncur*(nprev+1);
        }
            ap.assert((iscls  &  islinearout)  |  !iscls,  "FillHighLevelInformation:
internal error");
            //
            // Ініціалізація
            //
            idxweights = 0;
            idxneuro = 0;
            idxstruct = 0;
            idxconn = 0;
            network.hlnetworktype = 0;
//
// мережа без прихованих шарів
//
            if(nhid1==0)
            {
                network.hllayersizes = new int[2];
                network.hllayersizes[0] = nin;
                network.hllayersizes[1] = nout;
                if(!iscls)
                {
                    network.hlconnections = new int[hlconnfieldwidth*nin*nout];
                    network.hlneurons = new int[hlnfieldwidth*(nin+nout)];
                    network.hlnormtype = 0;
```

КОПЗ_2025

```
                }
                else
                {
                network.hlconnections = new int[hlconnfieldwidth*nin*(nout-1)];
                network.hlneurons = new int[hlnfieldwidth*(nin+nout)];
                network.hlnormtype = 1;
                }
hladdinputlayer(network, ref idxconn, ref idxneuro, ref idxstruct, nin);
a1(network, ref idxconn, ref idxneuro, ref idxstruct, ref idxweights, 1, nin,
nout, iscls, islinearout);
                return;
            }
//
// мережі з одним прихованим шаром
//
            if(nhid2==0)
            {
                network.hllayersizes = new int[3];
                network.hllayersizes[0] = nin;
                network.hllayersizes[1] = nhid1;
                network.hllayersizes[2] = nout;
                if(!iscls)
                {
network.hlconnections = new int[hlconnfieldwidth*(nin*nhid1+nhid1*nout)];
                    network.hlneurons = new int[hlnfieldwidth*(nin+nhid1+nout)];
                    network.hlnormtype = 0;
                }
                else
                {
network.hlconnections = new int[hlconnfieldwidth*(nin*nhid1+nhid1*(nout-1))];
                    network.hlneurons = new int[hlnfieldwidth*(nin+nhid1+nout)];
                    network.hlnormtype = 1;
                }
hladdinputlayer(network, ref idxconn, ref idxneuro, ref idxstruct, nin);
hladdhiddenlayer(network, ref idxconn, ref idxneuro, ref idxstruct, ref
idxweights, 1, nin, nhid1);
a1(network, ref idxconn, ref idxneuro, ref idxstruct, ref idxweights, 2, nhid1,
nout, iscls, islinearout);
                return;
            }

//
```

КОПІЯ - 2025

```
// 2 прихованих шари
//
            network.hllayersizes = new int[4];
            network.hllayersizes[0] = nin;
            network.hllayersizes[1] = nhid1;
            network.hllayersizes[2] = nhid2;
            network.hllayersizes[3] = nout;
            if(!iscls)
            {
                network.hlconnections = new
int[hlconnfieldwidth*(nin*nhid1+nhid1*nhid2+nhid2*nout)];
                network.hlneurons = new int[hlnfieldwidth*(nin+nhid1+nhid2+nout)];
                network.hlnormtype = 0;
            }
            else
            {
                network.hlconnections = new
int[hlconnfieldwidth*(nin*nhid1+nhid1*nhid2+nhid2*(nout-1))];
                network.hlneurons = new int[hlnfieldwidth*(nin+nhid1+nhid2+nout)];
                network.hlnormtype = 1;
            }
hladdinputlayer(network, ref idxconn, ref idxneuro, ref idxstruct, nin);
hladdhiddenlayer(network, ref idxconn, ref idxneuro, ref idxstruct, ref
idxweights, 1, nin, nhid1);
hladdhiddenlayer(network, ref idxconn, ref idxneuro, ref idxstruct, ref
idxweights, 2, nhid1, nhid2);
a1(network, ref idxconn, ref idxneuro, ref idxstruct, ref idxweights, 3, nhid2,
nout, iscls, islinearout);

}
```

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю.

UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою

КОПЗ_2025

програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об'єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації. UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

КПЗ-2025

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Розглянемо роботу процедури статистичного аналізу з використанням критерію χ^2:

```
public static bool a2(int[] val, int[] test_val)
    {
        int i;
        if(st_analysis.ap.len(val)!=st_analysis.ap.len(test_val))
            return false;
        for(i=0; i<st_analysis.ap.len(val); i++)
            if(val[i]!=test_val[i])
                return false;
        return true;
    }
    public static bool doc_st_analysisnt_matrix(int[,] val, int[,] test_val)
    {
        int i, j;
        if(st_analysis.ap.rows(val)!=st_analysis.ap.rows(test_val))
            return false;
        if(st_analysis.ap.cols(val)!=st_analysis.ap.cols(test_val))
            return false;
        for(i=0; i<st_analysis.ap.rows(val); i++)
            for(j=0; j<st_analysis.ap.cols(val); j++)
                if(val[i,j]!=test_val[i,j])
                    return false;
        return true;
    }
public static bool doc_test_real_vector(double[] val, double[] test_val, double
_threshold)
    {
        int i;
        if(st_analysis.ap.len(val)!=st_analysis.ap.len(test_val))
            return false;
        for(i=0; i<st_analysis.ap.len(val); i++)
        {
```

КОПЗ_2025

```
            double s = _threshold>=0 ? 1.0 : Math.Abs(test_val[i]);
            double threshold = Math.Abs(_threshold);
            if(Math.Abs(val[i]-test_val[i])/s>threshold)
                return false;
        }
        return true;
    }
    public static bool doc_test_real_matrix(double[,] val, double[,] test_val,
double _threshold)
    {
        int i, j;
        if(st_analysis.ap.rows(val)!=st_analysis.ap.rows(test_val))
            return false;
        if(st_analysis.ap.cols(val)!=st_analysis.ap.cols(test_val))
            return false;
        for(i=0; i<st_analysis.ap.rows(val); i++)
            for(j=0; j<st_analysis.ap.cols(val); j++)
            {
                double s = _threshold>=0 ? 1.0 : Math.Abs(test_val[i,j]);
                double threshold = Math.Abs(_threshold);
                if(Math.Abs(val[i,j]-test_val[i,j])/s>threshold)
                    return false;
            }
        return true;
    }
    public static bool doc_test_complex_vector(st_analysis.complex[] val,
st_analysis.complex[] test_val, double _threshold)
    {
        int i;
        if(st_analysis.ap.len(val)!=st_analysis.ap.len(test_val))
            return false;
        for(i=0; i<st_analysis.ap.len(val); i++)
        {
double s = _threshold>=0 ? 1.0 : st_analysis.math.abscomplex(test_val[i]);
            double threshold = Math.Abs(_threshold);
            if(st_analysis.math.abscomplex(val[i]-test_val[i])/s>threshold)
                return false;
        }
        return true;
    }
    public static bool doc_test_complex_matrix(st_analysis.complex[,] val,
st_analysis.complex[,] test_val, double _threshold)
```

КОПЗ_2025

```
    {
        int i, j;
        if(st_analysis.ap.rows(val)!=st_analysis.ap.rows(test_val))
            return false;
        if(st_analysis.ap.cols(val)!=st_analysis.ap.cols(test_val))
            return false;
        for(i=0; i<st_analysis.ap.rows(val); i++)
            for(j=0; j<st_analysis.ap.cols(val); j++)
            {
                double s = _threshold>=0 ? 1.0 :
st_analysis.math.abscomplex(test_val[i,j]);
                double threshold = Math.Abs(_threshold);
                if(st_analysis.math.abscomplex(val[i,j]-
test_val[i,j])/s>threshold)
                    return false;
            }
        return true;
    }
public static void spoil_vector_by_adding_element<T>(ref T[]x) where T:new()
    {
        int i;
        T[] y = x;
        x = new T[y.Length+1];
        for(i=0; i<y.Length; i++)
            x[i] = y[i];
        x[y.Length] = new T();
    }
public static void spoil_vector_by_deleting_element<T>(ref T[]x) where T:new()
    {
        int i;
        T[] y = x;
        x = new T[y.Length-1];
        for(i=0; i<y.Length-1; i++)
            x[i] = y[i];
    }
```

Розглянемо роботу процедури статистичного RS-аналізу.

```
public static void a3<T>(ref T[,] x) where T : new()
    {
        int i, j;
        T[,] y = x;
        x = new T[y.GetLength(0)+1,y.GetLength(1)];
```

КОПЗ_2025

```
        for(i=0; i<y.GetLength(0); i++)
            for(j=0; j<y.GetLength(1); j++)
                x[i,j] = y[i,j];
        for(j=0; j<y.GetLength(1); j++)
            x[y.GetLength(0),j] = new T();
    }
    public static void spoil_matrix_by_deleting_row<T>(ref T[,] x) where T : new()
    {
        int i, j;
        T[,] y = x;
        x = new T[y.GetLength(0)-1,y.GetLength(1)];
        for(i=0; i<y.GetLength(0)-1; i++)
            for(j=0; j<y.GetLength(1); j++)
                x[i,j] = y[i,j];
    }
    public static void spoil_matrix_by_adding_col<T>(ref T[,] x) where T : new()
    {
        int i, j;
        T[,] y = x;
        x = new T[y.GetLength(0), y.GetLength(1)+1];
        for(i=0; i<y.GetLength(0); i++)
            for(j=0; j<y.GetLength(1); j++)
                x[i,j] = y[i,j];
        for(i=0; i<y.GetLength(0); i++)
            x[i,y.GetLength(1)] = new T();
    }
public static void spoil_matrix_by_deleting_col<T>(ref T[,] x) where T : new()
    {
        int i, j;
        T[,] y = x;
        x = new T[y.GetLength(0), y.GetLength(1)-1];
        for(i=0; i<y.GetLength(0); i++)
            for(j=0; j<y.GetLength(1)-1; j++)
                x[i,j] = y[i,j];
    }
public static void spoil_vector_by_value<T>(ref T[] x, T val)
    {
if(x.Length!=0) x[st_analysis.math.randominteger(x.Length)] = val;
    }
public static void spoil_matrix_by_value<T>(ref T[,] x, T val)
    {
        if(x.GetLength(0)!=0 && x.GetLength(1)!=0)
```

КОПЗ_2025

```
x[st_analysis.math.randominteger(x.GetLength(0)),st_analysis.math.randominteger(x.
GetLength(1))] = val;
    }
    public static void function1_func(double[] x, ref double func, object obj)
    {
// цей зворотній виклик обчислює f(x0,x1) = 100*(x0+3)^4 + (x1-3)^4
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
    }

public static void function1_grad(double[] x, ref double func, double[] grad,
object obj)
    {
        // цей зворотній виклик обчислює f(x0,x1) = 100*(x0+3)^4 + (x1-3)^4
        // і її похідні df/d0 and df/dx1
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
        grad[0] = 400*System.Math.Pow(x[0]+3,3);
        grad[1] = 4*System.Math.Pow(x[1]-3,3);
    }

public static void function1_hess(double[] x, ref double func, double[] grad,
double[,] hess, object obj)
    {
// цей зворотній виклик обчислює f(x0,x1) = 100*(x0+3)^4 + (x1-3)^4
// її похідні df/d0 and df/dx1
// і її гессіан.
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
        grad[0] = 400*System.Math.Pow(x[0]+3,3);
        grad[1] = 4*System.Math.Pow(x[1]-3,3);
        hess[0,0] = 1200*System.Math.Pow(x[0]+3,2);
        hess[0,1] = 0;
        hess[1,0] = 0;
        hess[1,1] = 12*System.Math.Pow(x[1]-3,2);
    }
    public static void  function1_fvec(double[] x, double[] fi, object obj)
    {
//
// цей зворотній виклик обчислює
// f0(x0,x1) = 100*(x0+3)^4,
// f1(x0,x1) = (x1-3)^4
//
        fi[0] = 10*System.Math.Pow(x[0]+3,2);
```

КбПЗ_2025

```
        fi[1] = System.Math.Pow(x[1]-3,2);
    }

public static void  function1_jac(double[]x, double[]fi, double[,]jac, object obj)
    {
// цей зворотній виклик обчислює
// f0(x0,x1) = 100*(x0+3)^4,
// f1(x0,x1) = (x1-3)^4
// і матриці Якобі J = [dfi/dxj]
        fi[0] = 10*System.Math.Pow(x[0]+3,2);
        fi[1] = System.Math.Pow(x[1]-3,2);
        jac[0,0] = 20*(x[0]+3);
        jac[0,1] = 0;
        jac[1,0] = 0;
        jac[1,1] = 2*(x[1]-3);
    }
    public static void function2_func(double[] x, ref double func, object obj)
    {
//
// цей зворотній виклик обчислює f(x0,x1) = (x0^2+1)^2 + (x1-1)^2
//
        func = System.Math.Pow(x[0]*x[0]+1,2) + System.Math.Pow(x[1]-1,2);
    }
    public static void function2_grad(double[] x, ref double func, double[] grad,
object obj)
    {
//
// цей зворотній виклик обчислює f(x0,x1) = (x0^2+1)^2 + (x1-1)^2
// і її похідні df/d0 and df/dx1
//
        func = System.Math.Pow(x[0]*x[0]+1,2) + System.Math.Pow(x[1]-1,2);
        grad[0] = 4*(x[0]*x[0]+1)*x[0];
        grad[1] = 2*(x[1]-1);
    }
public static void function2_hess(double[]x, ref double func, double[] grad,
double[,] hess, object obj)
    {
//
// цей зворотній виклик обчислює f(x0,x1) = (x0^2+1)^2 + (x1-1)^2
// градієнт і гессіан
//
        func = System.Math.Pow(x[0]*x[0]+1,2) + System.Math.Pow(x[1]-1,2);
```

КбПЗ-2025

```
        grad[0] = 4*(x[0]*x[0]+1)*x[0];
        grad[1] = 2*(x[1]-1);
        hess[0,0] = 12*x[0]*x[0]+4;
        hess[0,1] = 0;
        hess[1,0] = 0;
        hess[1,1] = 2;
    }
public static void  function2_fvec(double[] x, double[] fi, object obj)
{
//
// цей зворотній виклик обчислює
// f0(x0,x1) = 100*(x0+3)^4,
// f1(x0,x1) = (x1-3)^4
//
        fi[0] = x[0]*x[0]+1;
        fi[1] = x[1]-1;
}
public static void  function2_jac(double[] x, double[] fi, double[,] jac, object
obj)
{
//
// цей зворотній виклик обчислює
// f0(x0,x1) = x0^2+1
// f1(x0,x1) = x1-1
// і матриці Якобі J = [dfi/dxj]
        fi[0] = x[0]*x[0]+1;
        fi[1] = x[1]-1;
        jac[0,0] = 2*x[0];
        jac[0,1] = 0;
        jac[1,0] = 0;
        jac[1,1] = 1;
    }
```

КОПЗ_2025

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Twofish, який є симетричним алгоритмом блочного шифрування з розміром блоку 128 біт і довжиною ключа до 256 біт. Число раундів 16. Розроблено групою фахівців на чолі з Брюсом Шнайером. Був одним з п'яти

фіналістів другого етапу конкурсу AES. Алгоритм розроблений на основі алгоритмів Blowfish, SAFER і Square.

Відмінними особливостями алгоритму є використання попередньо обчислюваних та залежних від ключа S-box'ів і складна схема розгортки підключення шифрування. Половина n-бітного ключа шифрування використовується як власне ключ шифрування, інша – для модифікації алгоритму (від неї залежать S-box'и).

Twofish розроблявся спеціально з урахуванням вимог та рекомендацій NIST для конкурсу AES [1]:

– 128-бітний блочний симетричний шифр.

– Довжина ключів 128, 192 і 256 біт.

– Відсутність слабких ключів.

– Ефективна програмна (в першу чергу на 32-бітних процесорах) та апаратна реалізація.

– Гнучкість (можливість використання додаткових довжин ключа, використання в поточному шифруванні, хеш-функціях і т.д.).

– Простота алгоритму – для можливості його ефективного аналізу.

Однак саме складність структури алгоритму і, відповідно, складність його аналізу на предмет слабких ключів або прихованих зв'язків, а також досить повільне час шифрування порівняно з Rijndael на більшості платформ, зіграло не на його користь.[2]

Алгоритм Twofish виник в результаті спроби модифіковані алгоритм Blowfish для 128-бітового вхідного блоку. Новий алгоритм повинен був бути легко реалізованим апаратно (у тому числі використовувати таблиці меншого розміру), мати досконалішу систему розширення ключа (key schedule) і мати однозначну функцію F.

В результаті, алгоритм був реалізований у вигляді змішаної мережі Фейстеля з чотирма гілками, які модифікують один одну з використанням криптоперетворень Адамара (Pseudo-Hadamar Transform, PHT).

КБПЗ-2025

Можливість ефективної реалізації на сучасних (для того часу) 32-бітних процесорах (а також в смарт-картах і подібних пристроях) – один з ключових принципів, яким керувалися розробники Twofish.

Наприклад, у функції F при обчисленні PHT і складання з частиною ключа K навмисно використовується додавання, замість традиційного xor. Це дає можливість використовувати команду LEA сімейства процесорів Pentium, яка за один такт дозволяє обчислити перетворення Адамара. (Правда в такому випадку код доводиться компілювати під конкретне значення ключа).

Алгоритм Twofish не запатентований і може бути використаний ким завгодно без будь-якої плати або відрахувань. Він використовується в багатьох програмах шифрування, хоча і отримав менше поширення, ніж Blowfish.

Опис алгоритму

Twofish розбиває вхідний 128-бітний блок даних на чотири 32-бітних підблоки, над якими, після процедури вхідного відбілювання (input whitening), проводиться 16 раундів перетворень. Після останнього раунду виконується вихідна відбілювання (output whitening).

Відбілювання (whitening)

Відбілювання – це процедура xor'а даних з підключами перед першим раундом і після останнього раунду. Вперше ця техніка була використана в шифрі Khufu/Khare і, незалежно, Рональдом Рівестом (англ. Ron Rivest) в алгоритмі шифрування DESX. Joe Killian (NEC) і Phillip Rogaway (Каліфорнійський університет) показали, що відбілювання дійсно ускладнює завдання пошуку ключа повним перебором (exhaustive key-search) в DESX[3] . Розробники Twofish стверджують[4], що в Twofish відбілювання також істотно ускладнює завдання підбору ключа, оскільки криптоаналітика не може дізнатися, які дані потрапляють на вхід функції F першого раунду.

Тим не менш, виявилися і негативні сторони. Цікаве дослідження провели фахівці дослідницького центру компанії IBM.[5] Вони виконали реалізацію алгоритму Twofish для типової смарт-карти з CMOS-архітектурою і

КОПЗ_2025

проаналізували можливість атаки за допомогою диференціального аналізу споживаної потужності (DPA – Differential Power Analysis). Атаці піддавалася саме процедура вхідного вибілювання, оскільки вона безпосередньо використовує xor підключів з вхідними даними. У результаті дослідники показали, що можна повністю обчислити 128-бітовий ключ проаналізувавши всього 100 операцій шифрування довільних блоків.

Функція g

Функція g – основа алгоритму Twofish. На вхід функції подається 32-бітове число X, яке потім розбивається на чотири байти x0, x1, x2, x3. Кожен з вийшов байтів пропускається через свій S-box. (Слід зазначити, що S-box'и в алгоритмі не фіксовані, а залежать від ключа). Отримані 4 байти на виходах S-box'ов інтерпретуються як вектор з чотирма компонентами. Цей вектор множиться на фіксовану матрицю MDS (maximum distance separable) розміром 4x4, причому обчислення проводяться в скінченному полі по модулю непривідного многочлена

MDS матриця – це така матриця над кінцевим полем K, що якщо взяти її як матрицю лінійного перетворення з простору у простір , то будь-які два вектори з простору виду (x, f (x)) будуть мати як мінімум m+1 відмінностей в компонентах. Тобто набір векторів вигляду (x, f (x)) утворює код, що володіє властивістю максимального рознесення (maximum distance separable code). Таким кодом, наприклад, є код Ріда-Соломона.

В Twofish властивість максимальної рознесеність матриці MDS означає, що загальна кількість змінних байт вектора a і вектора не менше п'яти. Іншими словами, будь-яка зміна тільки одного байта в a призводить до зміни всіх чотирьох байтів в b.

Криптоперетворення Адамара (Pseudo-Hadamar Transform, PHT)

Криптоперетворення Адамара – оборотне перетворення бітового рядка довжиною 2n. Рядок розбивається на дві частини a і b однакової довжини в n біт. Перетворення обчислюється таким чином:

КОПЗ-2025

)

)

Ця операція часто використовується для «розсіювання» коду (наприклад в шифрі SAFER).

В Twofish це перетворення використовується при змішуванні результатів двох g-функцій (n = 32).

Циклічний зсув на 1 біт

У кожному раунді два правих 32-бітових блоки, які xor-яться з результатами функції F, додатково циклічно зрушуються на один біт. Третій блок зрушується до операції xor, четвертий блок – після. Ці зрушення спеціально додані, щоб порушити вирівнювання по байтах, яке властиво S-box'ам та операції множення на MDS-матрицю. Проте шифр перестає бути повністю симетричним, так як при шифруванні й розшифровці зрушення слід здійснювати в протилежні сторони.

Генерація ключів

Twofish розрахований на роботу з ключами довжиною 128, 192 і 256 біт. З вихідного ключа генерується 40 32-бітних підключів, перші вісім з яких використовуються тільки в операціях вхідного і вихідного вибілювання, а решта 32 – в раундах шифрування, по два підключі на раунд. Особливістю Twofish є те, що вихідний ключ використовується також і для зміни самого алгоритму шифрування, так як використовуються у функції g S-box'и не фіксовані, а залежать від ключа.

Для формування раундових підключів вихідний ключ M розбивається з перестановкою байт на два однакові блоки M_o і M_e. Потім за допомогою блоку M_o і функції h шифрується значення 2 * i, а за допомогою блоку M_e шифрується значення 2*i+1, де i – номер поточного раунду (0 – 15). Отримані зашифровані блоки змішуються криптоперетвореням Адамара, і потім використовуються як раундові підключі.

КОПЗ – 2025

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ КІБЕРБЕЗПЕКИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

– Меню ПЗ, пункти: Файл; Аналіз; Фільтрація; Параметри; Довідка.

– Розділ обрання списку файлів для перевірки.

– Розділ виведення результатів перевірки.

– Розділ фільтрація даних.

– Функціональні дії.

КБПЗ – 2025

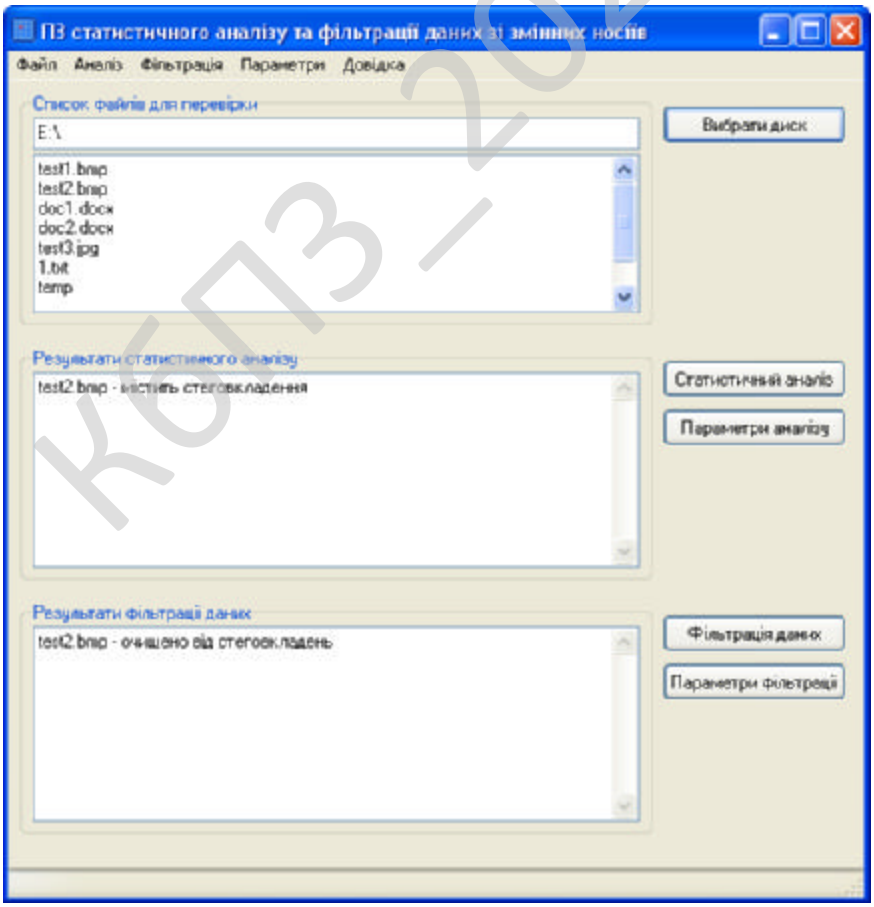

Рисунок 5.1 – Головне вікно програми

У вищезазначених розділах реалізується наступний функціонал:

– Налаштування ПЗ.

– Видрати диск.

– Параметри аналізу.

– Аналіз.

– Фільтрація даних

– Параметри фільтрації.

Для початку роботи з програмою користувач має під'єднати до комп'ютера змінний носій (флеш-диск, DVD-диск або ін.) для перевірки на наявність стеговкладень у його файлах. Після цього у програмі за допомогою кнопки «Вибрати диск» відкривається змінний диск, файли якого слід перевірити.

По натисненні на кнопку «Статистичний аналіз» здійснюється пошук стеговкладень у наявних на диску файлах методами статистичного стегоаналізу. Результати пошуку виводяться у текстовому полі «Результати статистичного аналізу» у вигляді списку файлів, що містять стеговкладення. По натисненні на кнопку «Параметри аналізу» з'являється діалогове вікно вибору параметрів статистичного аналізу.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

КПЗ_2025

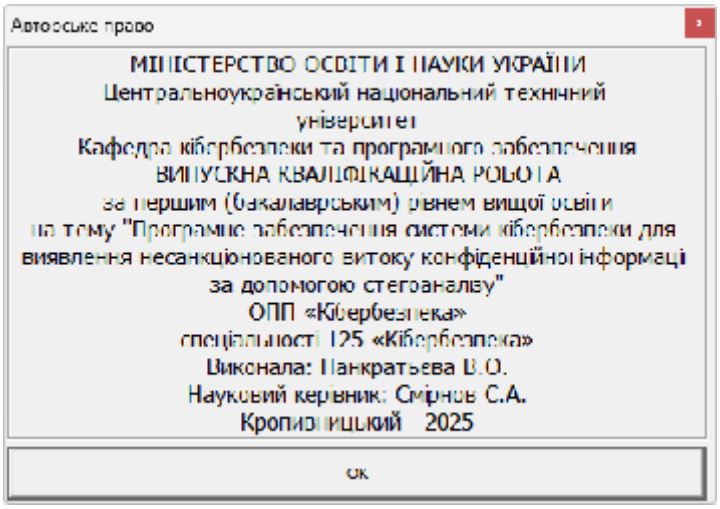

Рисунок 5.2 – Вікно авторського права

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості. Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму. В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

КПЗ-2025

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

– Досліджена система для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

– На основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Visual C#. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

КПЗ-2025

системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи кібербезпеки й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Twofish.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

КбПЗ-2025

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.

2. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.

3. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.

4. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.

5. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p

6. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.

7. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

8. Alex Matrosov, Eugene Rodionov, Sergey Bratus. Rootkits and Bootkits. No Starch Press. 2019. 450 p.

9. Lakhno, V., Malyukov, V., Smirnov, O., Bebeshko, B., Chubaievskiy, V., Zhumadilova, M., Malyukova, I., Smirnov, S. «Multifactorial Model for Targeted Attacks Counteracting Within the Framework of a Multi-Step Quality Game with Fuzzy Information». *8th International Symposium on Intelligent Informatics, ISI 2023*, 2025. vol 389. pp 377-389. Springer, Singapore.

10. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

11. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447.

КПЗ – 2025

12. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnova, T., Prokopov, S., Bilanovych, A. «New Cost Function for S-boxes Generation by Simulated Annealing Algorithm». Lecture Notes on Data Engineering and Communications Technologies, 2023. vol 180. pp. 310-320. Springer, Cham.

13. Kuznetsov, O., Frontoni, E., Kandiy, S., Smirnov, O., Ulianovska, Y., Kobylianska, O. «Heuristic Search for Nonlinear Substitutions for Cryptographic Applications». *Lecture Notes on Data Engineering and Communications Technologies,* 2023. vol 180. Springer, Cham. pp. 288-298.

14. Kuznetsov, O., Kuznetsova, Y., Smirnov, O., Kostenko, O., Zvieriev, V. «Evaluating Hashing Algorithms in the Age of ASIC Resistance». *CEUR Workshop Proceedings*, 2023, 3628, pp. 93-105.

15. Kuznetsov O., Frontoni E., Kuznetsova Ye., Smirnov O., Chevardin V. «Achieving Enhanced Security in Biometric Authentication: A Rigorous Analysis of Code-Based Fuzzy Extractor». *CEUR Workshop Proceedings,* Volume 3624, 2023, pp. 330-339.

16. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

17. Kuznetsov, O., Kandiy, S., Frontoni, E., Smirnov, O. «Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece». *CEUR Workshop Proceedings,* Volume 3504, 2023, pp. 1-11.

18. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022,

19. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile*

КБПЗ-2025

Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

20. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. Springer, Cham. 2022, pp. 2463-2477.

21. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». *11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021,* Cracow, Poland, 22-25 September 2021. P. 414-418

22. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». *4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021*, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

23. Smirnov O., Kuznetsov A., Girzheva O., Kiian A., Nakisko O., Kuznetsova T. «Advanced Code-Based Electronic Digital Signature Scheme». *2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020,* Kharkiv, 6 October 2020-9 October 2020, P. 358-362.

24. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.*

25. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

26. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

КОПІЯ 2025

27. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

28. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131.

29. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14.

30. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions». *Lecture Notes in Networks and Systems*, vol 152. Springer, Cham. 2021, pp 66-84.

31. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

32. Smirnov O., Kuznetsov A., Kiian A., Babenko V., Perevozova I.,Chepurko I. «New Approach to the Implementation of Post-Quantum Digital Signature Scheme». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 166-171.

33. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

34. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In:

КОПЭ 2025

Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

35. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

36. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43.

37. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660.

38. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

39. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

40. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019.

41. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

КОПЗ - 2025

42. Smirnov, O., Kuznetsov, A., Kiian, A., Gorbenko, Y., Cherep, O., Bexhter L. «Code-based Pseudorandom Generator for the Post-Quantum Period», *2019 IEEE International Conference on Advanced Trends in Information Theory (IEEE ATIT 2019).* 18.12.19-20.12.19 Kyiv Ukraine. P. 204 – 209.

43. Smirnov, O., Kuznetsov, A., Nariezhnii, O., Stelnyk, S., Kokhanovska, T., Kuznetsova T., «Side Channel Attack on a Quantum Random Number Generator», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18 - 21 September 2019. P.713-718.

44. Kuznetsova, T., «Code-Based Schemes for Post-Quantum Digital Signatures», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019;* Metz; France; 18-21 September 2019. P. 707-712.

45. Smirnov, O., Kuznetsov, A., Stefanovych, O., Gorbenko, Y., Krasnobaev, V., Kuznetsova K. «Information Hiding Using 3D-Printing Technology», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.701-706.

46. Smirnov, O., Hu, Z., Vasiliu, Y., Sydorenko, V., Polishchuk, Y., «Abstract Model of Eavesdropper and Overview on Attacks in Quantum Cryptography Systems», *10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2019*; Metz; France; 18-21 September 2019. P.399-405.

47. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

48. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes»,

КОПІЯ 2025

2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

49. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

50. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

51. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

52. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884.

53. Smirnov A.A., Kuznetsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

54. Ткаченко, О., Ільєнко, А., Улічев, О., Мелешко, Є., Смірнов, О. «Правові засади поширення інформаційних впливів в соціальних мережах». *Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка»,* 2024. № 2(26), С. 170–188.

КОПІЯ 2025

Додаток А
(обов’язковий)

Технічне завдання

Зміст

КОПЗ–2025

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 57-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу.

КбПЗ_2025

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи кібербезпеки з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

КБПЗ_2025

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Visual C#.

КбПЗ – 2025

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 85 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

КБПЗ_2025

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 3.06.2025 р.

КбПЗ_2025

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Смірнов С.А.

КбПЗ_2025

Програмне забезпечення системи кібербезпеки для виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 98

Літера: РП

Кропивницький – 2025 року

// st_analysis.cs – головний модуль програми, що реалізує виявлення несанкціонованого витоку конфіденційної інформації за допомогою стегоаналізу

```
#pragma warning disable 219
#pragma warning disable 162
using System;
public class MainTest
{

    public static bool doc_test_bool(bool val, bool test_val)
    { return (val && test_val) || (!val && !test_val); }

    public static bool doc_st_analysisnt(int val, int test_val)
    { return val==test_val; }

    public static bool doc_test_real(double val, double test_val, double
_threshold)
    {
        double s = _threshold>=0 ? 1.0 : Math.Abs(test_val);
        double threshold = Math.Abs(_threshold);
        return Math.Abs(val-test_val)/s<=threshold;
    }

    public static bool doc_test_complex(st_analysis.complex val,
st_analysis.complex test_val, double _threshold)
    {
        double s = _threshold>=0 ? 1.0 : st_analysis.math.abscomplex(test_val);
        double threshold = Math.Abs(_threshold);
        return st_analysis.math.abscomplex(val-test_val)/s<=threshold;
    }


    public static bool doc_test_bool_vector(bool[] val, bool[] test_val)
    {
        int i;
        if( st_analysis.ap.len(val)!=st_analysis.ap.len(test_val) )
            return false;
        for(i=0; i<st_analysis.ap.len(val); i++)
            if( val[i]!=test_val[i] )
                return false;
        return true;
    }

    public static bool doc_test_bool_matrix(bool[,] val, bool[,] test_val)
    {
        int i, j;
        if( st_analysis.ap.rows(val)!=st_analysis.ap.rows(test_val) )
            return false;
        if( st_analysis.ap.cols(val)!=st_analysis.ap.cols(test_val) )
            return false;
        for(i=0; i<st_analysis.ap.rows(val); i++)
            for(j=0; j<st_analysis.ap.cols(val); j++)
                if( val[i,j]!=test_val[i,j] )
                    return false;
        return true;
    }

    public static bool doc_st_analysisnt_vector(int[] val, int[] test_val)
    {
        int i;
        if( st_analysis.ap.len(val)!=st_analysis.ap.len(test_val) )
            return false;
        for(i=0; i<st_analysis.ap.len(val); i++)
            if( val[i]!=test_val[i] )
                return false;
        return true;
    }
```

КОПІЯ 2025

```
    public static bool doc_st_analysisnt_matrix(int[,] val, int[,] test_val)
    {
        int i, j;
        if( st_analysis.ap.rows(val)!=st_analysis.ap.rows(test_val) )
            return false;
        if( st_analysis.ap.cols(val)!=st_analysis.ap.cols(test_val) )
            return false;
        for(i=0; i<st_analysis.ap.rows(val); i++)
            for(j=0; j<st_analysis.ap.cols(val); j++)
                if( val[i,j]!=test_val[i,j] )
                    return false;
        return true;
    }

    public static bool doc_test_real_vector(double[] val, double[] test_val,
double _threshold)
    {
        int i;
        if( st_analysis.ap.len(val)!=st_analysis.ap.len(test_val) )
            return false;
        for(i=0; i<st_analysis.ap.len(val); i++)
        {
            double s = _threshold>=0 ? 1.0 : Math.Abs(test_val[i]);
            double threshold = Math.Abs(_threshold);
            if( Math.Abs(val[i]-test_val[i])/s>threshold )
                return false;
        }
        return true;
    }

    public static bool doc_test_real_matrix(double[,] val, double[,] test_val,
double _threshold)
    {
        int i, j;
        if( st_analysis.ap.rows(val)!=st_analysis.ap.rows(test_val) )
            return false;
        if( st_analysis.ap.cols(val)!=st_analysis.ap.cols(test_val) )
            return false;
        for(i=0; i<st_analysis.ap.rows(val); i++)
            for(j=0; j<st_analysis.ap.cols(val); j++)
            {
                double s = _threshold>=0 ? 1.0 : Math.Abs(test_val[i,j]);
                double threshold = Math.Abs(_threshold);
                if( Math.Abs(val[i,j]-test_val[i,j])/s>threshold )
                    return false;
            }
        return true;
    }

    public static bool doc_test_complex_vector(st_analysis.complex[] val,
st_analysis.complex[] test_val, double _threshold)
    {
        int i;
        if( st_analysis.ap.len(val)!=st_analysis.ap.len(test_val) )
            return false;
        for(i=0; i<st_analysis.ap.len(val); i++)
        {
            double s = _threshold>=0 ? 1.0 :
st_analysis.math.abscomplex(test_val[i]);
            double threshold = Math.Abs(_threshold);
            if( st_analysis.math.abscomplex(val[i]-test_val[i])/s>threshold )
                return false;
        }
        return true;
    }

    public static bool doc_test_complex_matrix(st_analysis.complex[,] val,
st_analysis.complex[,] test_val, double _threshold)
```

KGTU 2025

```
    {
        int i, j;
        if( st_analysis.ap.rows(val)!=st_analysis.ap.rows(test_val) )
            return false;
        if( st_analysis.ap.cols(val)!=st_analysis.ap.cols(test_val) )
            return false;
        for(i=0; i<st_analysis.ap.rows(val); i++)
            for(j=0; j<st_analysis.ap.cols(val); j++)
            {
                double s = _threshold>=0 ? 1.0 :
st_analysis.math.abscomplex(test_val[i,j]);
                double threshold = Math.Abs(_threshold);
                if( st_analysis.math.abscomplex(val[i,j]-
test_val[i,j])/s>threshold )
                    return false;
            }
        return true;
    }

    public static void spoil_vector_by_adding_element<T>(ref T[] x) where T :
new()
    {
        int i;
        T[] y = x;
        x = new T[y.Length+1];
        for(i=0; i<y.Length; i++)
            x[i] = y[i];
        x[y.Length] = new T();
    }

    public static void spoil_vector_by_deleting_element<T>(ref T[] x) where T :
new()
    {
        int i;
        T[] y = x;
        x = new T[y.Length-1];
        for(i=0; i<y.Length-1; i++)
            x[i] = y[i];
    }

    public static void spoil_matrix_by_adding_row<T>(ref T[,] x) where T : new()
    {
        int i, j;
        T[,] y = x;
        x = new T[y.GetLength(0)+1,y.GetLength(1)];
        for(i=0; i<y.GetLength(0); i++)
            for(j=0; j<y.GetLength(1); j++)
                x[i,j] = y[i,j];
        for(j=0; j<y.GetLength(1); j++)
            x[y.GetLength(0),j] = new T();
    }

    public static void spoil_matrix_by_deleting_row<T>(ref T[,] x) where T :
new()
    {
        int i, j;
        T[,] y = x;
        x = new T[y.GetLength(0)-1,y.GetLength(1)];
        for(i=0; i<y.GetLength(0)-1; i++)
            for(j=0; j<y.GetLength(1); j++)
                x[i,j] = y[i,j];
    }

    public static void spoil_matrix_by_adding_col<T>(ref T[,] x) where T : new()
    {
        int i, j;
        T[,] y = x;
        x = new T[y.GetLength(0), y.GetLength(1)+1];
        for(i=0; i<y.GetLength(0); i++)
```

КбПЗ_2025

```
            for(j=0; j<y.GetLength(1); j++)
                x[i,j] = y[i,j];
        for(i=0; i<y.GetLength(0); i++)
            x[i,y.GetLength(1)] = new T();
    }

    public static void spoil_matrix_by_deleting_col<T>(ref T[,] x) where T :
new()
    {
        int i, j;
        T[,] y = x;
        x = new T[y.GetLength(0), y.GetLength(1)-1];
        for(i=0; i<y.GetLength(0); i++)
            for(j=0; j<y.GetLength(1)-1; j++)
                x[i,j] = y[i,j];
    }

    public static void spoil_vector_by_value<T>(ref T[] x, T val)
    {
        if( x.Length!=0 )
            x[st_analysis.math.randominteger(x.Length)] = val;
    }
    public static void spoil_matrix_by_value<T>(ref T[,] x, T val)
    {
        if( x.GetLength(0)!=0 && x.GetLength(1)!=0 )

x[st_analysis.math.randominteger(x.GetLength(0)),st_analysis.math.randominteger(
x.GetLength(1))] = val;
    }

    public static void function1_func(double[] x, ref double func, object obj)
    {
        // цей зворотній виклик обчислює f(x0,x1) = 100*(x0+3)^4 + (x1-3)^4
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
    }
    public static void function1_grad(double[] x, ref double func, double[]
grad, object obj)
    {
        // цей зворотній виклик обчислює f(x0,x1) = 100*(x0+3)^4 + (x1-3)^4
        // і її похідні df/d0 and df/dx1
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
        grad[0] = 400*System.Math.Pow(x[0]+3,3);
        grad[1] = 4*System.Math.Pow(x[1]-3,3);
    }
    public static void function1_hess(double[] x, ref double func, double[]
grad, double[,] hess, object obj)
    {
        // цей зворотній виклик обчислює f(x0,x1) = 100*(x0+3)^4 + (x1-3)^4
        // її похідні df/d0 and df/dx1
        // і її гессіан.
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
        grad[0] = 400*System.Math.Pow(x[0]+3,3);
        grad[1] = 4*System.Math.Pow(x[1]-3,3);
        hess[0,0] = 1200*System.Math.Pow(x[0]+3,2);
        hess[0,1] = 0;
        hess[1,0] = 0;
        hess[1,1] = 12*System.Math.Pow(x[1]-3,2);
    }
    public static void  function1_fvec(double[] x, double[] fi, object obj)
    {
        //
        // цей зворотній виклик обчислює
        // f0(x0,x1) = 100*(x0+3)^4,
        // f1(x0,x1) = (x1-3)^4
        //
        fi[0] = 10*System.Math.Pow(x[0]+3,2);
        fi[1] = System.Math.Pow(x[1]-3,2);
    }
```

КПЗ_2025

```
    public static void  function1_jac(double[] x, double[] fi, double[,] jac,
object obj)
    {
        // цей зворотній виклик обчислює
        // f0(x0,x1) = 100*(x0+3)^4,
        // f1(x0,x1) = (x1-3)^4
        // і матриці Якобі J = [dfi/dxj]
        fi[0] = 10*System.Math.Pow(x[0]+3,2);
        fi[1] = System.Math.Pow(x[1]-3,2);
        jac[0,0] = 20*(x[0]+3);
        jac[0,1] = 0;
        jac[1,0] = 0;
        jac[1,1] = 2*(x[1]-3);
    }
    public static void function2_func(double[] x, ref double func, object obj)
    {
        //
        // цей зворотній виклик обчислює f(x0,x1) = (x0^2+1)^2 + (x1-1)^2
        //
        func = System.Math.Pow(x[0]*x[0]+1,2) + System.Math.Pow(x[1]-1,2);
    }
    public static void function2_grad(double[] x, ref double func, double[]
grad, object obj)
    {
        //
        // цей зворотній виклик обчислює f(x0,x1) = (x0^2+1)^2 + (x1-1)^2
        // і її похідні df/d0 and df/dx1
        //
        func = System.Math.Pow(x[0]*x[0]+1,2) + System.Math.Pow(x[1]-1,2);
        grad[0] = 4*(x[0]*x[0]+1)*x[0];
        grad[1] = 2*(x[1]-1);
    }
    public static void function2_hess(double[] x, ref double func, double[]
grad, double[,] hess, object obj)
    {
        //
        // цей зворотній виклик обчислює f(x0,x1) = (x0^2+1)^2 + (x1-1)^2
        // градієнт і гессіан
        //
        func = System.Math.Pow(x[0]*x[0]+1,2) + System.Math.Pow(x[1]-1,2);
        grad[0] = 4*(x[0]*x[0]+1)*x[0];
        grad[1] = 2*(x[1]-1);
        hess[0,0] = 12*x[0]*x[0]+4;
        hess[0,1] = 0;
        hess[1,0] = 0;
        hess[1,1] = 2;
    }
    public static void  function2_fvec(double[] x, double[] fi, object obj)
    {
        //
        // цей зворотній виклик обчислює
        // f0(x0,x1) = 100*(x0+3)^4,
        // f1(x0,x1) = (x1-3)^4
        //
        fi[0] = x[0]*x[0]+1;
        fi[1] = x[1]-1;
    }
    public static void  function2_jac(double[] x, double[] fi, double[,] jac,
object obj)
    {
        //
        // цей зворотній виклик обчислює
        // f0(x0,x1) = x0^2+1
        // f1(x0,x1) = x1-1
        // і матриці Якобі J = [dfi/dxj]
        //
        fi[0] = x[0]*x[0]+1;
        fi[1] = x[1]-1;
        jac[0,0] = 2*x[0];
```

КПЗ_2025

```
        jac[0,1] = 0;
        jac[1,0] = 0;
        jac[1,1] = 1;
    }
    public static void bad_func(double[] x, ref double func, object obj)
    {
        //
        // цей зворотній виклик обчислює «погану» функцію,
        // тобто функцію з неправильно розрахованими похідними
        //
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
    }
    public static void bad_grad(double[] x, ref double func, double[] grad,
object obj)
    {
        //
        // цей зворотній виклик обчислює «погану» функцію,
        // тобто функцію з неправильно розрахованими похідними
        //
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
        grad[0] = 40*System.Math.Pow(x[0]+3,3);
        grad[1] = 40*System.Math.Pow(x[1]-3,3);
    }
    public static void bad_hess(double[] x, ref double func, double[] grad,
double[,] hess, object obj)
    {
        //
        // цей зворотній виклик обчислює «погану» функцію,
        // тобто функцію з неправильно розрахованими похідними
        //
        func = 100*System.Math.Pow(x[0]+3,4) + System.Math.Pow(x[1]-3,4);
        grad[0] = 40*System.Math.Pow(x[0]+3,3);
        grad[1] = 40*System.Math.Pow(x[1]-3,3);
        hess[0,0] = 120*System.Math.Pow(x[0]+3,2);
        hess[0,1] = 1;
        hess[1,0] = 1;
        hess[1,1] = 120*System.Math.Pow(x[1]-3,2);
    }
    public static void  bad_fvec(double[] x, double[] fi, object obj)
    {
        //
        // цей зворотній виклик обчислює «погану» функцію,
        // тобто функцію з неправильно розрахованими похідними
        //
        fi[0] = 10*System.Math.Pow(x[0]+3,2);
        fi[1] = System.Math.Pow(x[1]-3,2);
    }
    public static void  bad_jac(double[] x, double[] fi, double[,] jac, object
obj)
    {
        //
        // цей зворотній виклик обчислює «погану» функцію,
        // тобто функцію з неправильно розрахованими похідними
        //
        fi[0] = 10*System.Math.Pow(x[0]+3,2);
        fi[1] = System.Math.Pow(x[1]-3,2);
        jac[0,0] = 20*(x[0]+3);
        jac[0,1] = 0;
        jac[1,0] = 1;
        jac[1,1] = 20*(x[1]-3);
    }
    public static void function_cx_1_func(double[] c, double[] x, ref double
func, object obj)
    {
        // цей зворотній виклик обчислює f(c,x)=exp(-c0*sqr(x0))
        // де x позиції по осі X і C регульований параметр
        func = System.Math.Exp(-c[0]*x[0]*x[0]);
    }
```

КОПІЯ - 2025

```
    public static void function_cx_1_grad(double[] c, double[] x, ref double
func, double[] grad, object obj)
    {
        // цей зворотній виклик обчислює f(c,x)=exp(-c0*sqr(x0)) і градієнт
G={df/dc[i]}
        // де x позиції по осі X і C регульований параметр.
        // ВАЖЛИВО: градієнт розраховується по відношенню до C, а не X
        func = System.Math.Exp(-c[0]*System.Math.Pow(x[0],2));
        grad[0] = -System.Math.Pow(x[0],2)*func;
    }
    public static void function_cx_1_hess(double[] c, double[] x, ref double
func, double[] grad, double[,] hess, object obj)
    {
        // цей зворотній виклик обчислює f(c,x)=exp(-c0*sqr(x0)), gradient
G={df/dc[i]} and Hessian H={d2f/(dc[i]*dc[j])}
        // де x позиції по осі X і C регульований параметр.
        // IMPORTANT: gradient/Hessian are calculated with respect to C, not to
X
        func = System.Math.Exp(-c[0]*System.Math.Pow(x[0],2));
        grad[0] = -System.Math.Pow(x[0],2)*func;
        hess[0,0] = System.Math.Pow(x[0],4)*func;
    }
    public static void ode_function_1_diff(double[] y, double x, double[] dy,
object obj)
    {
        // цей зворотній виклик обчислює f(y[],x)=-y[0]
        dy[0] = -y[0];
    }
    public static void int_function_1_func(double x, double xminusa, double
bminusx, ref double y, object obj)
    {
        // цей зворотній виклик обчислює f(x)=exp(x)
        y = Math.Exp(x);
    }
    public static void function_debt_func(double[] c, double[] x, ref double
func, object obj)
    {
        //
        // цей зворотній виклик обчислює f(c,x)=c[0]*(1+c[1]*(pow(x[0]-
1999,c[2])-1))
        //
        func = c[0]*(1+c[1]*(System.Math.Pow(x[0]-1999,c[2])-1));
    }
    public static void s1_grad(double[] x, ref double func, double[] grad,
object obj)
    {
        //
        // цей зворотній виклик обчислює f(x) = (1+x)^(-0.2) + (1-x)^(-0.3) +
1000*x and its gradient.
        //
        // Функція обрізається, коли обчислюється поблизу особливих точок або
поза [-1,+1].
        // !!! в цьому випадку градієнт не розраховується.
        //
        if( (x[0]<=-0.999999999999) || (x[0]>=+0.999999999999) )
        {
            func = 1.0E+300;
            return;
        }
        func = System.Math.Pow(1+x[0],-0.2) + System.Math.Pow(1-x[0],-0.3) +
1000*x[0];
        grad[0] = -0.2*System.Math.Pow(1+x[0],-1.2) +0.3*System.Math.Pow(1-
x[0],-1.3) + 1000;
    }

    public static int Main(string[] args)
    {
        bool _TotalResult = true;
        bool _TestResult;
```

КБТЗ_2025

```
        int _spoil_scenario;
        System.Console.WriteLine("Будь-ласка зачекайте...");
        try
        {
            //
            // Статистичний аналіз nneighbor_d_1
            //      Найближчий сусід пошуку, KNN запитів
            //
            System.Console.WriteLine("0/91");
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {
                    double[,] a = new double[,]{{0,0},{0,1},{1,0},{1,1}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.NegativeInfinity);
                    int nx = 2;
                    int ny = 0;
                    int normtype = 2;
                    st_analysis.kdtree kdt;
                    double[] x;
                    double[,] r = new double[0,0];
                    int k;
                    st_analysis.kdtreebuild(a, nx, ny, normtype, out kdt);
                    x = new double[]{-1,0};
                    k = st_analysis.kdtreequeryknn(kdt, x, 1);
                    _TestResult = _TestResult && doc_st_analysisnt(k, 1);
                    st_analysis.kdtreequeryresultsx(kdt, ref r);
                    _TestResult = _TestResult && doc_test_real_matrix(r, new
double[,]{{0,0}}, 0.05);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка, "nneighbor_d_1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз nneighbor_t_2
            //    При наступних запитах буфер функції треба використовувати
раніше виділеної пам'яті (якщо достатньо великий), так що буфер може містити
деяку інформацію від попереднього виклику           //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {
                    double[,] a = new double[,]{{0,0},{0,1},{1,0},{1,1}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.NegativeInfinity);
                    int nx = 2;
```

Копія 2025

```
                    int ny = 0;
                    int normtype = 2;
                    st_analysis.kdtree kdt;
                    double[] x;
                    double[,] rx = new double[0,0];
                    int k;
                    st_analysis.kdtreebuild(a, nx, ny, normtype, out kdt);
                    x = new double[]{+2,0};
                    k = st_analysis.kdtreequeryknn(kdt, x, 2, true);
                    _TestResult = _TestResult && doc_st_analysisnt(k, 2);
                    st_analysis.kdtreequeryresultsx(kdt, ref rx);
                    _TestResult = _TestResult && doc_test_real_matrix(rx, new
double[,]{{1,0},{1,1}}, 0.05);
                    x = new double[]{-2,0};
                    k = st_analysis.kdtreequeryknn(kdt, x, 1, true);
                    _TestResult = _TestResult && doc_st_analysisnt(k, 1);
                    st_analysis.kdtreequeryresultsx(kdt, ref rx);
                    _TestResult = _TestResult && doc_test_real_matrix(rx, new
double[,]{{0,0},{1,1}}, 0.05);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "nneighbor_t_2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз nneighbor_d_2
            //      Серіалізация KD-дерева
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {
                    double[,] a = new double[,]{{0,0},{0,1},{1,0},{1,1}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.NegativeInfinity);
                    int nx = 2;
                    int ny = 0;
                    int normtype = 2;
                    st_analysis.kdtree kdt0;
                    st_analysis.kdtree kdt1;
                    string s;
                    double[] x;
                    double[,] r0 = new double[0,0];
                    double[,] r1 = new double[0,0];

                    //
                    // Побудова дереве і його серіалізация
                    //
                    st_analysis.kdtreebuild(a, nx, ny, normtype, out kdt0);
                    st_analysis.kdtreeserialize(kdt0, out s);
                    st_analysis.kdtreeunserialize(s, out kdt1);

                    //
                    // Порівняння результатів запитів KNN
                    //
```

КОПЗ_2025

```
                    x = new double[]{-1,0};
                    st_analysis.kdtreequeryknn(kdt0, x, 1);
                    st_analysis.kdtreequeryresultsx(kdt0, ref r0);
                    st_analysis.kdtreequeryknn(kdt1, x, 1);
                    st_analysis.kdtreequeryresultsx(kdt1, ref r1);
                    _TestResult = _TestResult && doc_test_real_matrix(r0, new
double[,]{{0,0}}, 0.05);
                    _TestResult = _TestResult && doc_test_real_matrix(r1, new
double[,]{{0,0}}, 0.05);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "nneighbor_d_2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз basestat_d_base
            //      Basic functionality (moments, adev, median, percentile)
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {
                    double[] x = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double mean;
                    double variance;
                    double skewness;
                    double kurtosis;
                    double adev;
                    double p;
                    double v;

                    //
                 // розрахунки зразка моменту
                     // (Середнє відхилення, дисперсія, асиметрія, ексцес)
//
                    st_analysis.samplemoments(x, out mean, out variance, out
skewness, out kurtosis);
                    _TestResult = _TestResult && doc_test_real(mean, 28.5,
0.01);
                    _TestResult = _TestResult && doc_test_real(variance,
801.1667, 0.01);
                    _TestResult = _TestResult && doc_test_real(skewness, 0.5751,
0.01);
                    _TestResult = _TestResult && doc_test_real(kurtosis, -
1.2666, 0.01);

                    //
                    // середнє відхилення
                    //
                    st_analysis.sampleadev(x, out adev);
                    _TestResult = _TestResult && doc_test_real(adev, 23.2,
0.01);
```

КСПЗ-2025

```
                    //
                    // Медіана і процентиль
                    //
                    st_analysis.samplemedian(x, out v);
                    _TestResult = _TestResult && doc_test_real(v, 20.5, 0.01);
                    p = 0.5;
                    if( _spoil_scenario==3 )
                        p = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        p = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        p = (double)System.Double.NegativeInfinity;
                    st_analysis.samplepercentile(x, p, out v);
                    _TestResult = _TestResult && doc_test_real(v, 20.5, 0.01);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "basestat_d_base");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз basestat_d_c2
            //      Кореляція (коваріація) між двома випадковими величинами
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<10; _spoil_scenario++)
            {
                try
                {
                    //
                    // два зразки - x і y, вимірювання залежності між ними
//
                    double[] x = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_adding_element(ref x);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new double[]{0,1,2,3,4,5,6,7,8,9};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double v;

                / /
                     / / Розраховуються три види залежності:
                     / / * Коваріація
```

КОПЗ - 2025

```
                    / / * Кореляція Пірсона
                    / / * Кореляція Спірмена
                    / /                    v = st_analysis.cov2(x, y);
                    _TestResult = _TestResult && doc_test_real(v, 82.5, 0.001);
                    v = st_analysis.pearsoncorr2(x, y);
                    _TestResult = _TestResult && doc_test_real(v, 0.9627,
0.001);
                    v = st_analysis.spearmancorr2(x, y);
                    _TestResult = _TestResult && doc_test_real(v, 1.000, 0.001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "basestat_d_c2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз basestat_d_cm
            //     Кореляція (коваріація) між компонентами випадкового вектора
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {
                   / /
                     / / X являє собою зразок матриці:
                     / / * I-й рядок відповідає I-му спостереженню
                     / / * J-й стовпчик відповідає j-й зміній
                     / /                    double[,] x = new
double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-1,1},{-1,0,9}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double[,] c;

                    //
                    // Розраховуються три види залежності
                   // * Коваріація
                    // * Кореляція Пірсона
                     // * Кореляція Спірмена                    //
                    // Результат зберігається в C, C з [I, J] дорівнює кореляції
                     // (Ковариация) між I-й і j-й змінної X.
//                    st_analysis.covm(x, out c);
                    _TestResult = _TestResult && doc_test_real_matrix(c, new
double[,]{{1.80,0.60,-1.40},{0.60,0.70,-0.80},{-1.40,-0.80,14.70}}, 0.01);
                    st_analysis.pearsoncorrm(x, out c);
                    _TestResult = _TestResult && doc_test_real_matrix(c, new
double[,]{{1.000,0.535,-0.272},{0.535,1.000,-0.249},{-0.272,-0.249,1.000}},
0.01);
                    st_analysis.spearmancorrm(x, out c);
                    _TestResult = _TestResult && doc_test_real_matrix(c, new
double[,]{{1.000,0.556,-0.306},{0.556,1.000,-0.750},{-0.306,-0.750,1.000}},
0.01);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
```

КСІВ-2025

```
            { throw; }
        }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "basestat_d_cm");
        _TotalResult = _TotalResult && _TestResult;


        //
        // Статистичний аналіз basestat_d_cm2
        //     Кореляція (коваріація) між двома випадковими векторами
//
        _TestResult = true;
        for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
        {
            try
            {
             / /
                 / / X і Y наведені приклади матриць:
                 / / * I-й рядок відповідає I-го спостереження
                 / / * J-й стовпець відповідає j-й змінної
                 / /                    double[,] x = new
double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-1,1},{-1,0,9}};
                if( _spoil_scenario==0 )
                    spoil_matrix_by_value(ref x, (double)System.Double.NaN);
                if( _spoil_scenario==1 )
                    spoil_matrix_by_value(ref x,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==2 )
                    spoil_matrix_by_value(ref x,
(double)System.Double.NegativeInfinity);
                double[,] y = new double[,]{{2,3},{2,1},{-1,6},{-
9,9},{7,1}};
                if( _spoil_scenario==3 )
                    spoil_matrix_by_value(ref y, (double)System.Double.NaN);
                if( _spoil_scenario==4 )
                    spoil_matrix_by_value(ref y,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==5 )
                    spoil_matrix_by_value(ref y,
(double)System.Double.NegativeInfinity);
                double[,] c;

                //
                // Розраховуються три види залежності
                // * Коваріація
                   // * Кореляції Пірсона
                 // * Кореляції Спірмена
                //
// Результат зберігається в C, C з [I, J] дорівнює кореляції
                 // (Ковариация) між I-й змінної X і J-й змінної Y.
st_analysis.covm2(x, y, out c);
                _TestResult = _TestResult && doc_test_real_matrix(c, new
double[,]{{4.100,-3.250},{2.450,-1.500},{13.450,-5.750}}, 0.01);
                st_analysis.pearsoncorrm2(x, y, out c);
                _TestResult = _TestResult && doc_test_real_matrix(c, new
double[,]{{0.519,-0.699},{0.497,-0.518},{0.596,-0.433}}, 0.01);
                st_analysis.spearmancorrm2(x, y, out c);
                _TestResult = _TestResult && doc_test_real_matrix(c, new
double[,]{{0.541,-0.649},{0.216,-0.433},{0.433,-0.135}}, 0.01);
                _TestResult = _TestResult && (_spoil_scenario==-1);
            }
            catch(st_analysis.st_analysisexception)
            { _TestResult = _TestResult && (_spoil_scenario!=-1); }
            catch
            { throw; }
        }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "basestat_d_cm2");
        _TotalResult = _TotalResult && _TestResult;
```

КОПІЯ 2025

```
            //
            // Статистичний аналіз basestat_t_base

                     _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<34; _spoil_scenario++)
            {
                try
                {
                    double mean;
                    double variance;
                    double skewness;
                    double kurtosis;
                    double adev;
                    double p;
                    double v;


                    double[] x1 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x1,
(double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x1,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x1,
(double)System.Double.NegativeInfinity);
                    st_analysis.samplemoments(x1, out mean, out variance, out
skewness, out kurtosis);
                    double[] x2 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==3 )
                        spoil_vector_by_value(ref x2,
(double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref x2,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref x2,
(double)System.Double.NegativeInfinity);
                    st_analysis.sampleadev(x2, out adev);
                    double[] x3 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref x3,
(double)System.Double.NaN);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref x3,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_value(ref x3,
(double)System.Double.NegativeInfinity);
                    st_analysis.samplemedian(x3, out v);
                    double[] x4 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==9 )
                        spoil_vector_by_value(ref x4,
(double)System.Double.NaN);
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref x4,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref x4,
(double)System.Double.NegativeInfinity);
                    p = 0.5;
                    if( _spoil_scenario==12 )
                        p = (double)System.Double.NaN;
                    if( _spoil_scenario==13 )
                        p = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==14 )
```

КСПЗ-2025

```
                        p = (double)System.Double.NegativeInfinity;
                    st_analysis.samplepercentile(x4, p, out v);


                    double[] x5 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==15 )
                        spoil_vector_by_value(ref x5,
(double)System.Double.NaN);
                    if( _spoil_scenario==16 )
                        spoil_vector_by_value(ref x5,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==17 )
                        spoil_vector_by_value(ref x5,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==18 )
                        spoil_vector_by_deleting_element(ref x5);
                    st_analysis.samplemoments(x5, 10, out mean, out variance,
out skewness, out kurtosis);
                    double[] x6 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==19 )
                        spoil_vector_by_value(ref x6,
(double)System.Double.NaN);
                    if( _spoil_scenario==20 )
                        spoil_vector_by_value(ref x6,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==21 )
                        spoil_vector_by_value(ref x6,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==22 )
                        spoil_vector_by_deleting_element(ref x6);
                    st_analysis.sampleadev(x6, 10, out adev);
                    double[] x7 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==23 )
                        spoil_vector_by_value(ref x7,
(double)System.Double.NaN);
                    if( _spoil_scenario==24 )
                        spoil_vector_by_value(ref x7,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==25 )
                        spoil_vector_by_value(ref x7,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==26 )
                        spoil_vector_by_deleting_element(ref x7);
                    st_analysis.samplemedian(x7, 10, out v);
                    double[] x8 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==27 )
                        spoil_vector_by_value(ref x8,
(double)System.Double.NaN);
                    if( _spoil_scenario==28 )
                        spoil_vector_by_value(ref x8,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==29 )
                        spoil_vector_by_value(ref x8,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==30 )
                        spoil_vector_by_deleting_element(ref x8);
                    p = 0.5;
                    if( _spoil_scenario==31 )
                        p = (double)System.Double.NaN;
                    if( _spoil_scenario==32 )
                        p = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==33 )
                        p = (double)System.Double.NegativeInfinity;
                    st_analysis.samplepercentile(x8, 10, p, out v);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
```

KGB 2025

```
            { throw; }
        }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "basestat_t_base");
        _TotalResult = _TotalResult && _TestResult;


        //
        // Статистичний аналіз basestat_t_covcorr
        //
        _TestResult = true;
        for(_spoil_scenario=-1; _spoil_scenario<126; _spoil_scenario++)
        {
            try
            {
                double v;
                double[,] c;


                double[] x1 = new double[]{0,1,4,9,16,25,36,49,64,81};
                if( _spoil_scenario==0 )
                    spoil_vector_by_value(ref x1,
(double)System.Double.NaN);
                if( _spoil_scenario==1 )
                    spoil_vector_by_value(ref x1,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==2 )
                    spoil_vector_by_value(ref x1,
(double)System.Double.NegativeInfinity);
                if( _spoil_scenario==3 )
                    spoil_vector_by_adding_element(ref x1);
                if( _spoil_scenario==4 )
                    spoil_vector_by_deleting_element(ref x1);
                double[] y1 = new double[]{0,1,2,3,4,5,6,7,8,9};
                if( _spoil_scenario==5 )
                    spoil_vector_by_value(ref y1,
(double)System.Double.NaN);
                if( _spoil_scenario==6 )
                    spoil_vector_by_value(ref y1,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==7 )
                    spoil_vector_by_value(ref y1,
(double)System.Double.NegativeInfinity);
                if( _spoil_scenario==8 )
                    spoil_vector_by_adding_element(ref y1);
                if( _spoil_scenario==9 )
                    spoil_vector_by_deleting_element(ref y1);
                v = st_analysis.cov2(x1, y1);
                double[] x2 = new double[]{0,1,4,9,16,25,36,49,64,81};
                if( _spoil_scenario==10 )
                    spoil_vector_by_value(ref x2,
(double)System.Double.NaN);
                if( _spoil_scenario==11 )
                    spoil_vector_by_value(ref x2,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==12 )
                    spoil_vector_by_value(ref x2,
(double)System.Double.NegativeInfinity);
                if( _spoil_scenario==13 )
                    spoil_vector_by_adding_element(ref x2);
                if( _spoil_scenario==14 )
                    spoil_vector_by_deleting_element(ref x2);
                double[] y2 = new double[]{0,1,2,3,4,5,6,7,8,9};
                if( _spoil_scenario==15 )
                    spoil_vector_by_value(ref y2,
(double)System.Double.NaN);
                if( _spoil_scenario==16 )
                    spoil_vector_by_value(ref y2,
(double)System.Double.PositiveInfinity);
```

КБІЗ 2025

```
                    if( _spoil_scenario==17 )
                        spoil_vector_by_value(ref y2,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==18 )
                        spoil_vector_by_adding_element(ref y2);
                    if( _spoil_scenario==19 )
                        spoil_vector_by_deleting_element(ref y2);
                    v = st_analysis.pearsoncorr2(x2, y2);
                    double[] x3 = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==20 )
                        spoil_vector_by_value(ref x3,
(double)System.Double.NaN);
                    if( _spoil_scenario==21 )
                        spoil_vector_by_value(ref x3,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==22 )
                        spoil_vector_by_value(ref x3,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==23 )
                        spoil_vector_by_adding_element(ref x3);
                    if( _spoil_scenario==24 )
                        spoil_vector_by_deleting_element(ref x3);
                    double[] y3 = new double[]{0,1,2,3,4,5,6,7,8,9};
                    if( _spoil_scenario==25 )
                        spoil_vector_by_value(ref y3,
(double)System.Double.NaN);
                    if( _spoil_scenario==26 )
                        spoil_vector_by_value(ref y3,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==27 )
                        spoil_vector_by_value(ref y3,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==28 )
                        spoil_vector_by_adding_element(ref y3);
                    if( _spoil_scenario==29 )
                        spoil_vector_by_deleting_element(ref y3);
                    v = st_analysis.spearmancorr2(x3, y3);


                    double[] x1a = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==30 )
                        spoil_vector_by_value(ref x1a,
(double)System.Double.NaN);
                    if( _spoil_scenario==31 )
                        spoil_vector_by_value(ref x1a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==32 )
                        spoil_vector_by_value(ref x1a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==33 )
                        spoil_vector_by_deleting_element(ref x1a);
                    double[] y1a = new double[]{0,1,2,3,4,5,6,7,8,9};
                    if( _spoil_scenario==34 )
                        spoil_vector_by_value(ref y1a,
(double)System.Double.NaN);
                    if( _spoil_scenario==35 )
                        spoil_vector_by_value(ref y1a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==36 )
                        spoil_vector_by_value(ref y1a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==37 )
                        spoil_vector_by_deleting_element(ref y1a);
                    v = st_analysis.cov2(x1a, y1a, 10);
                    double[] x2a = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==38 )
                        spoil_vector_by_value(ref x2a,
(double)System.Double.NaN);
                    if( _spoil_scenario==39 )
```

КБПЗ-2025

```
                        spoil_vector_by_value(ref x2a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==40 )
                        spoil_vector_by_value(ref x2a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==41 )
                        spoil_vector_by_deleting_element(ref x2a);
                    double[] y2a = new double[]{0,1,2,3,4,5,6,7,8,9};
                    if( _spoil_scenario==42 )
                        spoil_vector_by_value(ref y2a,
(double)System.Double.NaN);
                    if( _spoil_scenario==43 )
                        spoil_vector_by_value(ref y2a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==44 )
                        spoil_vector_by_value(ref y2a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==45 )
                        spoil_vector_by_deleting_element(ref y2a);
                    v = st_analysis.pearsoncorr2(x2a, y2a, 10);
                    double[] x3a = new double[]{0,1,4,9,16,25,36,49,64,81};
                    if( _spoil_scenario==46 )
                        spoil_vector_by_value(ref x3a,
(double)System.Double.NaN);
                    if( _spoil_scenario==47 )
                        spoil_vector_by_value(ref x3a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==48 )
                        spoil_vector_by_value(ref x3a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==49 )
                        spoil_vector_by_deleting_element(ref x3a);
                    double[] y3a = new double[]{0,1,2,3,4,5,6,7,8,9};
                    if( _spoil_scenario==50 )
                        spoil_vector_by_value(ref y3a,
(double)System.Double.NaN);
                    if( _spoil_scenario==51 )
                        spoil_vector_by_value(ref y3a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==52 )
                        spoil_vector_by_value(ref y3a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==53 )
                        spoil_vector_by_deleting_element(ref y3a);
                    v = st_analysis.spearmancorr2(x3a, y3a, 10);


                    double[,] x4 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==54 )
                        spoil_matrix_by_value(ref x4,
(double)System.Double.NaN);
                    if( _spoil_scenario==55 )
                        spoil_matrix_by_value(ref x4,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==56 )
                        spoil_matrix_by_value(ref x4,
(double)System.Double.NegativeInfinity);
                    st_analysis.covm(x4, out c);
                    double[,] x5 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==57 )
                        spoil_matrix_by_value(ref x5,
(double)System.Double.NaN);
                    if( _spoil_scenario==58 )
                        spoil_matrix_by_value(ref x5,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==59 )
```

KGB-2025

```
                        spoil_matrix_by_value(ref x5,
(double)System.Double.NegativeInfinity);
                    st_analysis.pearsoncorrm(x5, out c);
                    double[,] x6 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==60 )
                        spoil_matrix_by_value(ref x6,
(double)System.Double.NaN);
                    if( _spoil_scenario==61 )
                        spoil_matrix_by_value(ref x6,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==62 )
                        spoil_matrix_by_value(ref x6,
(double)System.Double.NegativeInfinity);
                    st_analysis.spearmancorrm(x6, out c);


                    double[,] x7 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==63 )
                        spoil_matrix_by_value(ref x7,
(double)System.Double.NaN);
                    if( _spoil_scenario==64 )
                        spoil_matrix_by_value(ref x7,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==65 )
                        spoil_matrix_by_value(ref x7,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==66 )
                        spoil_matrix_by_deleting_row(ref x7);
                    if( _spoil_scenario==67 )
                        spoil_matrix_by_deleting_col(ref x7);
                    st_analysis.covm(x7, 5, 3, out c);
                    double[,] x8 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==68 )
                        spoil_matrix_by_value(ref x8,
(double)System.Double.NaN);
                    if( _spoil_scenario==69 )
                        spoil_matrix_by_value(ref x8,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==70 )
                        spoil_matrix_by_value(ref x8,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==71 )
                        spoil_matrix_by_deleting_row(ref x8);
                    if( _spoil_scenario==72 )
                        spoil_matrix_by_deleting_col(ref x8);
                    st_analysis.pearsoncorrm(x8, 5, 3, out c);
                    double[,] x9 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==73 )
                        spoil_matrix_by_value(ref x9,
(double)System.Double.NaN);
                    if( _spoil_scenario==74 )
                        spoil_matrix_by_value(ref x9,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==75 )
                        spoil_matrix_by_value(ref x9,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==76 )
                        spoil_matrix_by_deleting_row(ref x9);
                    if( _spoil_scenario==77 )
                        spoil_matrix_by_deleting_col(ref x9);
                    st_analysis.spearmancorrm(x9, 5, 3, out c);


                    double[,] x10 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
```

KGTS-2025

```
                    if( _spoil_scenario==78 )
                        spoil_matrix_by_value(ref x10,
(double)System.Double.NaN);
                    if( _spoil_scenario==79 )
                        spoil_matrix_by_value(ref x10,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==80 )
                        spoil_matrix_by_value(ref x10,
(double)System.Double.NegativeInfinity);
                    double[,] y10 = new double[,]{{2,3},{2,1},{-1,6},{-
9,9},{7,1}};
                    if( _spoil_scenario==81 )
                        spoil_matrix_by_value(ref y10,
(double)System.Double.NaN);
                    if( _spoil_scenario==82 )
                        spoil_matrix_by_value(ref y10,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==83 )
                        spoil_matrix_by_value(ref y10,
(double)System.Double.NegativeInfinity);
                    st_analysis.covm2(x10, y10, out c);
                    double[,] x11 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==84 )
                        spoil_matrix_by_value(ref x11,
(double)System.Double.NaN);
                    if( _spoil_scenario==85 )
                        spoil_matrix_by_value(ref x11,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==86 )
                        spoil_matrix_by_value(ref x11,
(double)System.Double.NegativeInfinity);
                    double[,] y11 = new double[,]{{2,3},{2,1},{-1,6},{-
9,9},{7,1}};
                    if( _spoil_scenario==87 )
                        spoil_matrix_by_value(ref y11,
(double)System.Double.NaN);
                    if( _spoil_scenario==88 )
                        spoil_matrix_by_value(ref y11,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==89 )
                        spoil_matrix_by_value(ref y11,
(double)System.Double.NegativeInfinity);
                    st_analysis.pearsoncorrm2(x11, y11, out c);
                    double[,] x12 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==90 )
                        spoil_matrix_by_value(ref x12,
(double)System.Double.NaN);
                    if( _spoil_scenario==91 )
                        spoil_matrix_by_value(ref x12,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==92 )
                        spoil_matrix_by_value(ref x12,
(double)System.Double.NegativeInfinity);
                    double[,] y12 = new double[,]{{2,3},{2,1},{-1,6},{-
9,9},{7,1}};
                    if( _spoil_scenario==93 )
                        spoil_matrix_by_value(ref y12,
(double)System.Double.NaN);
                    if( _spoil_scenario==94 )
                        spoil_matrix_by_value(ref y12,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==95 )
                        spoil_matrix_by_value(ref y12,
(double)System.Double.NegativeInfinity);
                    st_analysis.spearmancorrm2(x12, y12, out c);
```

KGU3_2025

```
                    double[,] x13 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==96 )
                        spoil_matrix_by_value(ref x13,
(double)System.Double.NaN);
                    if( _spoil_scenario==97 )
                        spoil_matrix_by_value(ref x13,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==98 )
                        spoil_matrix_by_value(ref x13,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==99 )
                        spoil_matrix_by_deleting_row(ref x13);
                    if( _spoil_scenario==100 )
                        spoil_matrix_by_deleting_col(ref x13);
                    double[,] y13 = new double[,]{{2,3},{2,1},{-1,6},{-
9,9},{7,1}};
                    if( _spoil_scenario==101 )
                        spoil_matrix_by_value(ref y13,
(double)System.Double.NaN);
                    if( _spoil_scenario==102 )
                        spoil_matrix_by_value(ref y13,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==103 )
                        spoil_matrix_by_value(ref y13,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==104 )
                        spoil_matrix_by_deleting_row(ref y13);
                    if( _spoil_scenario==105 )
                        spoil_matrix_by_deleting_col(ref y13);
                    st_analysis.covm2(x13, y13, 5, 3, 2, out c);
                    double[,] x14 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==106 )
                        spoil_matrix_by_value(ref x14,
(double)System.Double.NaN);
                    if( _spoil_scenario==107 )
                        spoil_matrix_by_value(ref x14,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==108 )
                        spoil_matrix_by_value(ref x14,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==109 )
                        spoil_matrix_by_deleting_row(ref x14);
                    if( _spoil_scenario==110 )
                        spoil_matrix_by_deleting_col(ref x14);
                    double[,] y14 = new double[,]{{2,3},{2,1},{-1,6},{-
9,9},{7,1}};
                    if( _spoil_scenario==111 )
                        spoil_matrix_by_value(ref y14,
(double)System.Double.NaN);
                    if( _spoil_scenario==112 )
                        spoil_matrix_by_value(ref y14,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==113 )
                        spoil_matrix_by_value(ref y14,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==114 )
                        spoil_matrix_by_deleting_row(ref y14);
                    if( _spoil_scenario==115 )
                        spoil_matrix_by_deleting_col(ref y14);
                    st_analysis.pearsoncorrm2(x14, y14, 5, 3, 2, out c);
                    double[,] x15 = new double[,]{{1,0,1},{1,1,0},{-1,1,0},{-2,-
1,1},{-1,0,9}};
                    if( _spoil_scenario==116 )
                        spoil_matrix_by_value(ref x15,
(double)System.Double.NaN);
                    if( _spoil_scenario==117 )
```

KSTR-2025

```
                        spoil_matrix_by_value(ref x15,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==118 )
                        spoil_matrix_by_value(ref x15,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==119 )
                        spoil_matrix_by_deleting_row(ref x15);
                    if( _spoil_scenario==120 )
                        spoil_matrix_by_deleting_col(ref x15);
                    double[,] y15 = new double[,]{{2,3},{2,1},{-1,6},{-
9,9},{7,1}};
                    if( _spoil_scenario==121 )
                        spoil_matrix_by_value(ref y15,
(double)System.Double.NaN);
                    if( _spoil_scenario==122 )
                        spoil_matrix_by_value(ref y15,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==123 )
                        spoil_matrix_by_value(ref y15,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==124 )
                        spoil_matrix_by_deleting_row(ref y15);
                    if( _spoil_scenario==125 )
                        spoil_matrix_by_deleting_col(ref y15);
                    st_analysis.spearmancorrm2(x15, y15, 5, 3, 2, out c);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка",
"basestat_t_covcorr");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matinv_d_r1
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<7; _spoil_scenario++)
            {
                try
                {
                    double[,] a = new double[,]{{1,-1},{1,1}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref a);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref a);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref a);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_col(ref a);
                    int info;
                    st_analysis.matinvreport rep;
                    st_analysis.rmatrixinverse(ref a, out info, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    _TestResult = _TestResult && doc_test_real_matrix(a, new
double[,]{{0.5,0.5},{-0.5,0.5}}, 0.00005);
```

КПЗ – 2025

```
                    _TestResult = _TestResult && doc_test_real(rep.r1, 0.5,
0.00005);
                    _TestResult = _TestResult && doc_test_real(rep.rinf, 0.5,
0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matinv_d_r1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matinv_d_c1
            //     Комплексна зворотня матриця
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<7; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex[,] a = new st_analysis.complex[,]{{new
st_analysis.complex(0,1),-1},{new st_analysis.complex(0,1),1}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(st_analysis.complex)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref a);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref a);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref a);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_col(ref a);
                    int info;
                    st_analysis.matinvreport rep;
                    st_analysis.cmatrixinverse(ref a, out info, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    _TestResult = _TestResult && doc_test_complex_matrix(a, new
st_analysis.complex[,]{{new st_analysis.complex(0,-0.5),new
st_analysis.complex(0,-0.5)},{-0.5,0.5}}, 0.00005);
                    _TestResult = _TestResult && doc_test_real(rep.r1, 0.5,
0.00005);
                    _TestResult = _TestResult && doc_test_real(rep.rinf, 0.5,
0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matinv_d_c1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matinv_d_spd1
```

КСЗ-2025

```
            //      SPD зворотня матриця
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<7; _spoil_scenario++)
            {
                try
                {
                    double[,] a = new double[,]{{2,1},{1,2}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref a);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref a);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref a);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_col(ref a);
                    int info;
                    st_analysis.matinvreport rep;
                    st_analysis.spdmatrixinverse(ref a, out info, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    _TestResult = _TestResult && doc_test_real_matrix(a, new
double[,]{{0.666666,-0.333333},{-0.333333,0.666666}}, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matinv_d_spd1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matinv_d_hpd1
            //      HPD зворотня матриця
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<7; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex[,] a = new
st_analysis.complex[,]{{2,1},{1,2}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(st_analysis.complex)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref a);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref a);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref a);
                    if( _spoil_scenario==6 )
```

КбПЗ_2015

```
                    spoil_matrix_by_deleting_col(ref a);
                int info;
                st_analysis.matinvreport rep;
                st_analysis.hpdmatrixinverse(ref a, out info, out rep);
                _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                _TestResult = _TestResult && doc_test_complex_matrix(a, new
st_analysis.complex[,]{{0.666666,-0.333333},{-0.333333,0.666666}}, 0.00005);
                _TestResult = _TestResult && (_spoil_scenario==-1);
            }
            catch(st_analysis.st_analysisexception)
            { _TestResult = _TestResult && (_spoil_scenario!=-1); }
            catch
            { throw; }
        }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "matinv_d_hpd1");
        _TotalResult = _TotalResult && _TestResult;


        //
        // Статистичний аналіз matinv_t_r1
            //
        _TestResult = true;
        try
        {
            double[,] a = new double[,]{{1,-1},{-2,2}};
            int info;
            st_analysis.matinvreport rep;
            st_analysis.rmatrixinverse(ref a, out info, out rep);
            _TestResult = _TestResult && doc_st_analysisnt(info, -3);
            _TestResult = _TestResult && doc_test_real(rep.r1, 0.0,
0.00005);
            _TestResult = _TestResult && doc_test_real(rep.rinf, 0.0,
0.00005);
        }
        catch(st_analysis.st_analysisexception)
        { _TestResult = false; }
        catch
        { throw; }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "matinv_t_r1");
        _TotalResult = _TotalResult && _TestResult;


        //
        // Статистичний аналіз matinv_t_c1
        //
        _TestResult = true;
        try
        {
            st_analysis.complex[,] a = new st_analysis.complex[,]{{new
st_analysis.complex(0,1),new st_analysis.complex(0,-1)},{-2,2}};
            int info;
            st_analysis.matinvreport rep;
            st_analysis.cmatrixinverse(ref a, out info, out rep);
            _TestResult = _TestResult && doc_st_analysisnt(info, -3);
            _TestResult = _TestResult && doc_test_real(rep.r1, 0.0,
0.00005);
            _TestResult = _TestResult && doc_test_real(rep.rinf, 0.0,
0.00005);
        }
        catch(st_analysis.st_analysisexception)
        { _TestResult = false; }
        catch
        { throw; }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "matinv_t_c1");
        _TotalResult = _TotalResult && _TestResult;
```

КОПІЯ 2025

```
            //
            // Статистичний аналіз matinv_e_spd1
            //
            _TestResult = true;
            try
            {
                double[,] a = new double[,]{{1,0},{1,1}};
                int info;
                st_analysis.matinvreport rep;
                st_analysis.spdmatrixinverse(ref a, out info, out rep);
                _TestResult = false;
            }
            catch(st_analysis.st_analysisexception)
            {}
            catch
            { throw; }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matinv_e_spd1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matinv_e_hpd1
            //
            //
            _TestResult = true;
            try
            {
                st_analysis.complex[,] a = new
st_analysis.complex[,]{{1,0},{1,1}};
                int info;
                st_analysis.matinvreport rep;
                st_analysis.hpdmatrixinverse(ref a, out info, out rep);
                _TestResult = false;
            }
            catch(st_analysis.st_analysisexception)
            {}
            catch
            { throw; }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matinv_e_hpd1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз mincg_d_1
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {
              //
                    // мінімізація F (x, y) = 100 * (x +3) ^ 4 + (Y-3) ^ 4
                    // З нелінійним методом сполучених градієнтів.
                    //                    double[] x = new double[]{0,0};
                  if( _spoil_scenario==0 )
                      spoil_vector_by_value(ref x, (double)System.Double.NaN);
                  if( _spoil_scenario==1 )
                      spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                  if( _spoil_scenario==2 )
                      spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                  double epsg = 0.0000000001;
                  if( _spoil_scenario==3 )
                      epsg = (double)System.Double.NaN;
                  if( _spoil_scenario==4 )
```

КСПЗ_2025

```
                    epsg = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==5 )
                    epsg = (double)System.Double.NegativeInfinity;
                double epsf = 0;
                if( _spoil_scenario==6 )
                    epsf = (double)System.Double.NaN;
                if( _spoil_scenario==7 )
                    epsf = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==8 )
                    epsf = (double)System.Double.NegativeInfinity;
                double epsx = 0;
                if( _spoil_scenario==9 )
                    epsx = (double)System.Double.NaN;
                if( _spoil_scenario==10 )
                    epsx = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==11 )
                    epsx = (double)System.Double.NegativeInfinity;
                int maxits = 0;
                st_analysis.mincgstate state;
                st_analysis.mincgreport rep;

                st_analysis.mincgcreate(x, out state);
                st_analysis.mincgsetcond(state, epsg, epsf, epsx, maxits);
                st_analysis.mincgoptimize(state, function1_grad, null,
null);
                st_analysis.mincgresults(state, out x, out rep);

                _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,3}, 0.005);
                _TestResult = _TestResult && (_spoil_scenario==-1);
            }
            catch(st_analysis.st_analysisexception)
            { _TestResult = _TestResult && (_spoil_scenario!=-1); }
            catch
            { throw; }
        }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "mincg_d_1");
        _TotalResult = _TotalResult && _TestResult;


        //
        // Статистичний аналіз mincg_d_2
        //
        //
        _TestResult = true;
        for(_spoil_scenario=-1; _spoil_scenario<18; _spoil_scenario++)
        {
            try
            {

                double[] x = new double[]{0,0};
                if( _spoil_scenario==0 )
                    spoil_vector_by_value(ref x, (double)System.Double.NaN);
                if( _spoil_scenario==1 )
                    spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==2 )
                    spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                double epsg = 0.0000000001;
                if( _spoil_scenario==3 )
                    epsg = (double)System.Double.NaN;
                if( _spoil_scenario==4 )
                    epsg = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==5 )
                    epsg = (double)System.Double.NegativeInfinity;
```

КОПЗ_2025

```
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    double stpmax = 0.1;
                    if( _spoil_scenario==12 )
                        stpmax = (double)System.Double.NaN;
                    if( _spoil_scenario==13 )
                        stpmax = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==14 )
                        stpmax = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.mincgstate state;
                    st_analysis.mincgreport rep;

                    // перший запуск
                    st_analysis.mincgcreate(x, out state);
                    st_analysis.mincgsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.mincgsetstpmax(state, stpmax);
                    st_analysis.mincgoptimize(state, function1_grad, null,
null);
                    st_analysis.mincgresults(state, out x, out rep);

                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,3}, 0.005);

                    // другий запуск - алгоритм поновлюється mincgrestartfrom ()
x = new double[]{10,10};
                    if( _spoil_scenario==15 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==16 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==17 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    st_analysis.mincgrestartfrom(state, x);
                    st_analysis.mincgoptimize(state, function1_grad, null,
null);
                    st_analysis.mincgresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "mincg_d_2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз mincg_numdiff
```

КБТЗ - 2025

```
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<15; _spoil_scenario++)
            {
                try
                {


                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    double diffstep = 1.0e-6;
                    if( _spoil_scenario==12 )
                        diffstep = (double)System.Double.NaN;
                    if( _spoil_scenario==13 )
                        diffstep = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==14 )
                        diffstep = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.mincgstate state;
                    st_analysis.mincgreport rep;

                    st_analysis.mincgcreatef(x, diffstep, out state);
                    st_analysis.mincgsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.mincgoptimize(state, function1_func, null,
null);
                    st_analysis.mincgresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "mincg_numdiff");
```

КОПІЯ 2025

```
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз mincg_ftrim
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 1.0e-6;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.mincgstate state;
                    st_analysis.mincgreport rep;

                    st_analysis.mincgcreate(x, out state);
                    st_analysis.mincgsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.mincgoptimize(state, s1_grad, null, null);
                    st_analysis.mincgresults(state, out x, out rep);

                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-0.99917305}, 0.000005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "mincg_ftrim");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minbleic_d_1
            //
```

КОПІЯ 2025

```
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<22; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double[] bndl = new double[]{-1,-1};
                    if( _spoil_scenario==3 )
                        spoil_vector_by_value(ref bndl,
(double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref bndl);
                    double[] bndu = new double[]{+1,+1};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref bndu,
(double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_deleting_element(ref bndu);
                    st_analysis.minbleicstate state;
                    st_analysis.minbleicreport rep;


                    double epsg = 0.000001;
                    if( _spoil_scenario==7 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==8 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==9 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==10 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==11 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==12 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==13 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==14 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==15 )
                        epsx = (double)System.Double.NegativeInfinity;


                    double epso = 0.00001;
                    if( _spoil_scenario==16 )
                        epso = (double)System.Double.NaN;
                    if( _spoil_scenario==17 )
                        epso = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==18 )
                        epso = (double)System.Double.NegativeInfinity;
                    double epsi = 0.00001;
                    if( _spoil_scenario==19 )
                        epsi = (double)System.Double.NaN;
                    if( _spoil_scenario==20 )
                        epsi = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==21 )
                        epsi = (double)System.Double.NegativeInfinity;
```

Kolb 2025

```
                    st_analysis.minbleiccreate(x, out state);
                    st_analysis.minbleicsetbc(state, bndl, bndu);
                    st_analysis.minbleicsetinnercond(state, epsg, epsf, epsx);
                    st_analysis.minbleicsetoutercond(state, epso, epsi);
                    st_analysis.minbleicoptimize(state, function1_grad, null,
null);
                    st_analysis.minbleicresults(state, out x, out rep);


                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-1,1}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minbleic_d_1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minbleic_d_2
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<24; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{5,5};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double[,] c = new double[,]{{1,0,2},{1,1,6}};
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_value(ref c, (double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_value(ref c,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_value(ref c,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_row(ref c);
                    if( _spoil_scenario==7 )
                        spoil_matrix_by_deleting_col(ref c);
                    int[] ct = new int[]{1,1};
                    if( _spoil_scenario==8 )
                        spoil_vector_by_deleting_element(ref ct);
                    st_analysis.minbleicstate state;
                    st_analysis.minbleicreport rep;


                    double epsg = 0.000001;
                    if( _spoil_scenario==9 )
                        epsg = (double)System.Double.NaN;
```

КБ ІЗ_2025

```
                    if( _spoil_scenario==10 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==12 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==13 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==14 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==15 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==16 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==17 )
                        epsx = (double)System.Double.NegativeInfinity;


                    double epso = 0.00001;
                    if( _spoil_scenario==18 )
                        epso = (double)System.Double.NaN;
                    if( _spoil_scenario==19 )
                        epso = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==20 )
                        epso = (double)System.Double.NegativeInfinity;
                    double epsi = 0.00001;
                    if( _spoil_scenario==21 )
                        epsi = (double)System.Double.NaN;
                    if( _spoil_scenario==22 )
                        epsi = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==23 )
                        epsi = (double)System.Double.NegativeInfinity;


                    st_analysis.minbleiccreate(x, out state);
                    st_analysis.minbleicsetlc(state, c, ct);
                    st_analysis.minbleicsetinnercond(state, epsg, epsf, epsx);
                    st_analysis.minbleicsetoutercond(state, epso, epsi);
                    st_analysis.minbleicoptimize(state, function1_grad, null,
null);
                    st_analysis.minbleicresults(state, out x, out rep);


                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{2,4}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minbleic_d_2");
            _TotalResult = _TotalResult && _TestResult;



            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<25; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
```

КоПЗ-2025

```
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double[] bndl = new double[]{-1,-1};
                    if( _spoil_scenario==3 )
                        spoil_vector_by_value(ref bndl,
(double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref bndl);
                    double[] bndu = new double[]{+1,+1};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref bndu,
(double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_deleting_element(ref bndu);
                    st_analysis.minbleicstate state;
                    st_analysis.minbleicreport rep;


                    double epsg = 0.000001;
                    if( _spoil_scenario==7 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==8 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==9 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==10 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==11 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==12 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==13 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==14 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==15 )
                        epsx = (double)System.Double.NegativeInfinity;


                    double epso = 0.00001;
                    if( _spoil_scenario==16 )
                        epso = (double)System.Double.NaN;
                    if( _spoil_scenario==17 )
                        epso = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==18 )
                        epso = (double)System.Double.NegativeInfinity;
                    double epsi = 0.00001;
                    if( _spoil_scenario==19 )
                        epsi = (double)System.Double.NaN;
                    if( _spoil_scenario==20 )
                        epsi = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==21 )
                        epsi = (double)System.Double.NegativeInfinity;


                    double diffstep = 1.0e-6;
                    if( _spoil_scenario==22 )
                        diffstep = (double)System.Double.NaN;
                    if( _spoil_scenario==23 )
                        diffstep = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==24 )
```

Kolb 2025

```
                        diffstep = (double)System.Double.NegativeInfinity;


                    st_analysis.minbleiccreatef(x, diffstep, out state);
                    st_analysis.minbleicsetbc(state, bndl, bndu);
                    st_analysis.minbleicsetinnercond(state, epsg, epsf, epsx);
                    st_analysis.minbleicsetoutercond(state, epso, epsi);
                    st_analysis.minbleicoptimize(state, function1_func, null,
null);
                    st_analysis.minbleicresults(state, out x, out rep);
                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-1,1}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minbleic_numdiff");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minbleic_ftrim
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<18; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 1.0e-6;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    double epso = 1.0e-6;
                    if( _spoil_scenario==12 )
                        epso = (double)System.Double.NaN;
```

КСПЗ-2025

```
                    if( _spoil_scenario==13 )
                        epso = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==14 )
                        epso = (double)System.Double.NegativeInfinity;
                    double epsi = 1.0e-6;
                    if( _spoil_scenario==15 )
                        epsi = (double)System.Double.NaN;
                    if( _spoil_scenario==16 )
                        epsi = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==17 )
                        epsi = (double)System.Double.NegativeInfinity;
                    st_analysis.minbleicstate state;
                    st_analysis.minbleicreport rep;

                    st_analysis.minbleiccreate(x, out state);
                    st_analysis.minbleicsetinnercond(state, epsg, epsf, epsx);
                    st_analysis.minbleicsetoutercond(state, epso, epsi);
                    st_analysis.minbleicoptimize(state, s1_grad, null, null);
                    st_analysis.minbleicresults(state, out x, out rep);

                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-0.99917305}, 0.000005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minbleic_ftrim");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз mcpd_simple1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {

                    st_analysis.mcpdstate s;
                    st_analysis.mcpdreport rep;
                    double[,] p;
                    double[,] track0 = new
double[,]{{1.00000,0.00000},{0.95000,0.05000},{0.92750,0.07250},{0.91738,0.08263
},{0.91282,0.08718}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref track0,
(double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref track0,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref track0,
(double)System.Double.NegativeInfinity);
                    double[,] track1 = new
double[,]{{0.80000,0.20000},{0.86000,0.14000},{0.88700,0.11300},{0.89915,0.10085
}};
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_value(ref track1,
(double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_value(ref track1,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==5 )
```

КПЗ – 2025

```
                        spoil_matrix_by_value(ref track1,
(double)System.Double.NegativeInfinity);

                    st_analysis.mcpdcreate(2, out s);
                    st_analysis.mcpdaddtrack(s, track0);
                    st_analysis.mcpdaddtrack(s, track1);
                    st_analysis.mcpdsolve(s);
                    st_analysis.mcpdresults(s, out p, out rep);


                    _TestResult = _TestResult && doc_test_real_matrix(p, new
double[,]{{0.95,0.50},{0.05,0.50}}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "mcpd_simple1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз mcpd_simple2
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {

                    st_analysis.mcpdstate s;
                    st_analysis.mcpdreport rep;
                    double[,] p;
                    double[,] track0 = new
double[,]{{1.000000,0.000000,0.000000},{0.950000,0.050000,0.000000},{0.927500,0.
060000,0.012500},{0.911125,0.061375,0.027500},{0.896256,0.060900,0.042844}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref track0,
(double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref track0,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref track0,
(double)System.Double.NegativeInfinity);
                    double[,] track1 = new
double[,]{{0.800000,0.200000,0.000000},{0.860000,0.090000,0.050000},{0.862000,0.
065500,0.072500},{0.851650,0.059475,0.088875},{0.838805,0.057451,0.103744}};
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_value(ref track1,
(double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_value(ref track1,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_value(ref track1,
(double)System.Double.NegativeInfinity);

                    st_analysis.mcpdcreate(3, out s);
                    st_analysis.mcpdaddtrack(s, track0);
                    st_analysis.mcpdaddtrack(s, track1);
                    st_analysis.mcpdaddec(s, 0, 2, 0.0);
                    st_analysis.mcpdaddec(s, 1, 2, 0.0);
                    st_analysis.mcpdaddec(s, 2, 2, 1.0);
                    st_analysis.mcpdaddec(s, 2, 0, 0.0);
                    st_analysis.mcpdsolve(s);
```

КОПІЯ 2025

```
                    st_analysis.mcpdresults(s, out p, out rep);

                    _TestResult = _TestResult && doc_test_real_matrix(p, new
double[,]{{0.95,0.50,0.00},{0.05,0.25,0.00},{0.00,0.25,1.00}}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "mcpd_simple2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlbfgs_d_1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlbfgsstate state;
                    st_analysis.minlbfgsreport rep;

                    st_analysis.minlbfgscreate(1, x, out state);
                    st_analysis.minlbfgssetcond(state, epsg, epsf, epsx,
maxits);
                    st_analysis.minlbfgsoptimize(state, function1_grad, null,
null);
                    st_analysis.minlbfgsresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
```

КБТВ-2025

```
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlbfgs_d_1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlbfgs_d_2
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<18; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    double stpmax = 0.1;
                    if( _spoil_scenario==12 )
                        stpmax = (double)System.Double.NaN;
                    if( _spoil_scenario==13 )
                        stpmax = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==14 )
                        stpmax = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlbfgsstate state;
                    st_analysis.minlbfgsreport rep;

                    // перший запуск
                    st_analysis.minlbfgscreate(1, x, out state);
                    st_analysis.minlbfgssetcond(state, epsg, epsf, epsx,
maxits);
```

КОПІЯ 2025

```
                    st_analysis.minlbfgssetstpmax(state, stpmax);
                    st_analysis.minlbfgsoptimize(state, function1_grad, null,
null);
                    st_analysis.minlbfgsresults(state, out x, out rep);

                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,3}, 0.005);

                    // другий запуск - алгоритм перезавантажується
                    x = new double[]{10,10};
                    if( _spoil_scenario==15 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==16 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==17 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    st_analysis.minlbfgsrestartfrom(state, x);
                    st_analysis.minlbfgsoptimize(state, function1_grad, null,
null);
                    st_analysis.minlbfgsresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlbfgs_d_2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlbfgs_numdiff
            //
_TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<15; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
```

КбПЗ - 2025

```
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    double diffstep = 1.0e-6;
                    if( _spoil_scenario==12 )
                        diffstep = (double)System.Double.NaN;
                    if( _spoil_scenario==13 )
                        diffstep = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==14 )
                        diffstep = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlbfgsstate state;
                    st_analysis.minlbfgsreport rep;

                    st_analysis.minlbfgscreatef(1, x, diffstep, out state);
                    st_analysis.minlbfgssetcond(state, epsg, epsf, epsx,
maxits);
                    st_analysis.minlbfgsoptimize(state, function1_func, null,
null);
                    st_analysis.minlbfgsresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlbfgs_numdiff");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlbfgs_ftrim
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 1.0e-6;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
```

КоПЗ_2025

```
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlbfgsstate state;
                    st_analysis.minlbfgsreport rep;

                    st_analysis.minlbfgscreate(1, x, out state);
                    st_analysis.minlbfgssetcond(state, epsg, epsf, epsx,
maxits);
                    st_analysis.minlbfgsoptimize(state, s1_grad, null, null);
                    st_analysis.minlbfgsresults(state, out x, out rep);

                     _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-0.99917305}, 0.000005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlbfgs_ftrim");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз odesolver_d1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<13; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{1};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] x = new double[]{0,1,2,3};
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double eps = 0.00001;
                    if( _spoil_scenario==7 )
```

КбПЗ 2025

```
                        eps = (double)System.Double.NaN;
                    if( _spoil_scenario==8 )
                        eps = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==9 )
                        eps = (double)System.Double.NegativeInfinity;
                    double h = 0;
                    if( _spoil_scenario==10 )
                        h = (double)System.Double.NaN;
                    if( _spoil_scenario==11 )
                        h = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==12 )
                        h = (double)System.Double.NegativeInfinity;
                    st_analysis.odesolverstate s;
                    int m;
                    double[] xtbl;
                    double[,] ytbl;
                    st_analysis.odesolverreport rep;
                    st_analysis.odesolverrkck(y, x, eps, h, out s);
                    st_analysis.odesolversolve(s, ode_function_1_diff, null);
                    st_analysis.odesolverresults(s, out m, out xtbl, out ytbl,
out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(m, 4);
                    _TestResult = _TestResult && doc_test_real_vector(xtbl, new
double[]{0,1,2,3}, 0.005);
                    _TestResult = _TestResult && doc_test_real_matrix(ytbl, new
double[,]{{1},{0.367},{0.135},{0.050}}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "odesolver_d1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз fft_complex_d1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {

                    st_analysis.complex[] z = new st_analysis.complex[]{new
st_analysis.complex(0,1),new st_analysis.complex(0,1),new
st_analysis.complex(0,1),new st_analysis.complex(0,1)};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.NegativeInfinity);
                    st_analysis.fftc1d(ref z);
                    _TestResult = _TestResult && doc_test_complex_vector(z, new
st_analysis.complex[]{new st_analysis.complex(0,4),0,0,0}, 0.0001);


                    st_analysis.fftc1dinv(ref z);
                    _TestResult = _TestResult && doc_test_complex_vector(z, new
st_analysis.complex[]{new st_analysis.complex(0,1),new
```

КбПЗ 2025

```
st_analysis.complex(0,1),new st_analysis.complex(0,1),new
st_analysis.complex(0,1)}, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "fft_complex_d1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз fft_complex_d2
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {

                    st_analysis.complex[] z = new
st_analysis.complex[]{0,1,0,new st_analysis.complex(0,1)};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.NegativeInfinity);
                    st_analysis.fftc1d(ref z);
                    _TestResult = _TestResult && doc_test_complex_vector(z, new
st_analysis.complex[]{new st_analysis.complex(1,+1),new st_analysis.complex(-1,-
1),new st_analysis.complex(-1,-1),new st_analysis.complex(1,+1)}, 0.0001);



                    st_analysis.fftc1dinv(ref z);
                    _TestResult = _TestResult && doc_test_complex_vector(z, new
st_analysis.complex[]{0,1,0,new st_analysis.complex(0,1)}, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "fft_complex_d2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз fft_real_d1
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{1,1,1,1};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
```

КПІ 2025

```
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    st_analysis.complex[] f;
                    double[] x2;
                    st_analysis.fftr1d(x, out f);
                    _TestResult = _TestResult && doc_test_complex_vector(f, new
st_analysis.complex[]{4,0,0,0}, 0.0001);


                    st_analysis.fftr1dinv(f, out x2);
                    _TestResult = _TestResult && doc_test_real_vector(x2, new
double[]{1,1,1,1}, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "fft_real_d1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз fft_real_d2
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{1,2,3,4};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    st_analysis.complex[] f;
                    double[] x2;
                    st_analysis.fftr1d(x, out f);
                    _TestResult = _TestResult && doc_test_complex_vector(f, new
st_analysis.complex[]{10,new st_analysis.complex(-2,+2),-2,new
st_analysis.complex(-2,-2)}, 0.0001);


                    st_analysis.fftr1dinv(f, out x2);
                    _TestResult = _TestResult && doc_test_real_vector(x2, new
double[]{1,2,3,4}, 0.0001);
                    f = new st_analysis.complex[]{10,new st_analysis.complex(-
2,+2),-2};
                    st_analysis.fftr1dinv(f, 4, out x2);
                    _TestResult = _TestResult && doc_test_real_vector(x2, new
double[]{1,2,3,4}, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
```

КОПЗ_2025

```
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "fft_real_d2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз fft_complex_e1
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<3; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex[] z = new st_analysis.complex[]{0,2,0,-
2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref z,
(st_analysis.complex)System.Double.NegativeInfinity);
                    st_analysis.fftc1dinv(ref z);
                    _TestResult = _TestResult && doc_test_complex_vector(z, new
st_analysis.complex[]{0,new st_analysis.complex(0,1),0,new
st_analysis.complex(0,-1)}, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "fft_complex_e1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз autogk_d1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {

                    double a = 0;
                    if( _spoil_scenario==0 )
                        a = (double)System.Double.NaN;
                    if( _spoil_scenario==1 )
                        a = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==2 )
                        a = (double)System.Double.NegativeInfinity;
                    double b = 1;
                    if( _spoil_scenario==3 )
                        b = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        b = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        b = (double)System.Double.NegativeInfinity;
                    st_analysis.autogkstate s;
                    double v;
                    st_analysis.autogkreport rep;
```

КбТВ_2025

```
                    st_analysis.autogksmooth(a, b, out s);
                    st_analysis.autogkintegrate(s, int_function_1_func, null);
                    st_analysis.autogkresults(s, out v, out rep);

                    _TestResult = _TestResult && doc_test_real(v, 1.7182,
0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "autogk_d1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_d_calcdiff
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,1,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_adding_element(ref x);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new double[]{0,0,2};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double t = -1;
                    if( _spoil_scenario==10 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        t = (double)System.Double.NegativeInfinity;
                    double v;
                    double dv;
                    double d2v;
                    st_analysis.barycentricinterpolant p;

                                        st_analysis.polynomialbuild(x, y, out p);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
```

КСТБ 2025

```
                                        st_analysis.barycentricdiff1(p, t, out
v, out dv);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && doc_test_real(dv, -3.0,
0.00005);

                                        st_analysis.barycentricdiff1(p, t, out
v, out dv);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && doc_test_real(dv, -3.0,
0.00005);
                    st_analysis.barycentricdiff2(p, t, out v, out dv, out d2v);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && doc_test_real(dv, -3.0,
0.00005);
                    _TestResult = _TestResult && doc_test_real(d2v, 2.0,
0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка",
"polint_d_calcdiff");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_d_conv
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<5; _spoil_scenario++)
            {
                try
                {

                    double[] a = new double[]{0,-1,+1};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref a, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref a,
(double)System.Double.NegativeInfinity);
                    double t = 2;
                    if( _spoil_scenario==3 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.NegativeInfinity;
                    double[] a2;
                    double v;
                    st_analysis.barycentricinterpolant p;


                    st_analysis.polynomialpow2bar(a, out p);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.005);

                                        st_analysis.polynomialbar2pow(p, out
a2);
                    _TestResult = _TestResult && doc_test_real_vector(a2, new
double[]{0,-1,+1}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
```

КОПЗ – 2025

```
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_d_conv");
            _TotalResult = _TotalResult && _TestResult;



            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<11; _spoil_scenario++)
            {
                try
                {

                    double[] y_eqdist = new double[]{0,0,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y_eqdist,
(double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y_eqdist,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y_eqdist,
(double)System.Double.NegativeInfinity);
                    double[] y_cheb1 = new double[]{-
0.116025,0.000000,1.616025};
                    if( _spoil_scenario==3 )
                        spoil_vector_by_value(ref y_cheb1,
(double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref y_cheb1,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y_cheb1,
(double)System.Double.NegativeInfinity);
                    double[] y_cheb2 = new double[]{0,0,2};
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y_cheb2,
(double)System.Double.NaN);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y_cheb2,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_value(ref y_cheb2,
(double)System.Double.NegativeInfinity);
                    st_analysis.barycentricinterpolant p_eqdist;
                    st_analysis.barycentricinterpolant p_cheb1;
                    st_analysis.barycentricinterpolant p_cheb2;
                    double[] a_eqdist;
                    double[] a_cheb1;
                    double[] a_cheb2;


                    st_analysis.polynomialbar2pow(p_eqdist, out a_eqdist);
                    _TestResult = _TestResult && doc_test_real_vector(a_eqdist,
new double[]{0,-1,+1}, 0.00005);

                    st_analysis.polynomialbuildcheb1(-1, +1, y_cheb1, out
p_cheb1);
                    st_analysis.polynomialbar2pow(p_cheb1, out a_cheb1);
                    _TestResult = _TestResult && doc_test_real_vector(a_cheb1,
new double[]{0,-1,+1}, 0.00005);

                    st_analysis.polynomialbuildcheb2(-1, +1, y_cheb2, out
p_cheb2);
                    st_analysis.polynomialbar2pow(p_cheb2, out a_cheb2);
```

КОПІЯ 2025

```
                    _TestResult = _TestResult && doc_test_real_vector(a_cheb2,
new double[]{0,-1,+1}, 0.00005);


                    double t = -2;
                    if( _spoil_scenario==9 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==10 )
                        t = (double)System.Double.NegativeInfinity;
                    double v;
                    v = st_analysis.polynomialcalceqdist(0.0, 2.0, y_eqdist, t);
                    _TestResult = _TestResult && doc_test_real(v, 6.0, 0.00005);

                    v = st_analysis.polynomialcalccheb1(-1, +1, y_cheb1, t);
                    _TestResult = _TestResult && doc_test_real(v, 6.0, 0.00005);

                    v = st_analysis.polynomialcalccheb2(-1, +1, y_cheb2, t);
                    _TestResult = _TestResult && doc_test_real(v, 6.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_d_spec");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<10; _spoil_scenario++)
            {
                try
                {
                    double[] x = new double[]{0,1,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new double[]{0,0,2};
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_deleting_element(ref y);
                    double t = -1;
                    if( _spoil_scenario==8 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==9 )
                        t = (double)System.Double.NegativeInfinity;
                    st_analysis.barycentricinterpolant p;
                    double v;
                    st_analysis.polynomialbuild(x, y, 3, out p);
```

КОПЗ – 2025

```
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_2
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{0,0,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref y);
                    double t = -1;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        t = (double)System.Double.NegativeInfinity;
                    st_analysis.barycentricinterpolant p;
                    double v;
                    st_analysis.polynomialbuildeqdist(0.0, 2.0, y, 3, out p);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_3
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{-0.116025,0.000000,1.616025};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
```

КОПІЯ 2025

```
                    spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==2 )
                    spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                if( _spoil_scenario==3 )
                    spoil_vector_by_deleting_element(ref y);
                double t = -1;
                if( _spoil_scenario==4 )
                    t = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==5 )
                    t = (double)System.Double.NegativeInfinity;
                st_analysis.barycentricinterpolant p;
                double v;
                st_analysis.polynomialbuildcheb1(-1.0, +1.0, y, 3, out p);
                v = st_analysis.barycentriccalc(p, t);
                _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                _TestResult = _TestResult && (_spoil_scenario==-1);
            }
            catch(st_analysis.st_analysisexception)
            { _TestResult = _TestResult && (_spoil_scenario!=-1); }
            catch
            { throw; }
        }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "polint_t_3");
        _TotalResult = _TotalResult && _TestResult;


        //
        // Статистичний аналіз polint_t_4
        //
        //
        _TestResult = true;
        for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
        {
            try
            {
                double[] y = new double[]{0,0,2};
                if( _spoil_scenario==0 )
                    spoil_vector_by_value(ref y, (double)System.Double.NaN);
                if( _spoil_scenario==1 )
                    spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==2 )
                    spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                if( _spoil_scenario==3 )
                    spoil_vector_by_deleting_element(ref y);
                double t = -2;
                if( _spoil_scenario==4 )
                    t = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==5 )
                    t = (double)System.Double.NegativeInfinity;
                double a = -1;
                if( _spoil_scenario==6 )
                    a = (double)System.Double.NaN;
                if( _spoil_scenario==7 )
                    a = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==8 )
                    a = (double)System.Double.NegativeInfinity;
                double b = +1;
                if( _spoil_scenario==9 )
                    b = (double)System.Double.NaN;
                if( _spoil_scenario==10 )
                    b = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==11 )
                    b = (double)System.Double.NegativeInfinity;
                st_analysis.barycentricinterpolant p;
```

КОПЗ_2025

```
                    double v;
                    st_analysis.polynomialbuildcheb2(a, b, y, 3, out p);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 6.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_4");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_5
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{0,0,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref y);
                    double t = -1;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        t = (double)System.Double.NegativeInfinity;
                    double v;
                    v = st_analysis.polynomialcalceqdist(0.0, 2.0, y, 3, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_5");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_6
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{-0.116025,0.000000,1.616025};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
```

КПІ 2025

```
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref y);
                    double t = -1;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        t = (double)System.Double.NegativeInfinity;
                    double a = -1;
                    if( _spoil_scenario==6 )
                        a = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        a = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        a = (double)System.Double.NegativeInfinity;
                    double b = +1;
                    if( _spoil_scenario==9 )
                        b = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        b = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        b = (double)System.Double.NegativeInfinity;
                    double v;
                    v = st_analysis.polynomialcalccheb1(a, b, y, 3, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_6");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_7
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{0,0,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref y);
                    double t = -2;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        t = (double)System.Double.NegativeInfinity;
                    double a = -1;
                    if( _spoil_scenario==6 )
                        a = (double)System.Double.NaN;
```

КбТВ_2025

```
                    if( _spoil_scenario==7 )
                        a = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        a = (double)System.Double.NegativeInfinity;
                    double b = +1;
                    if( _spoil_scenario==9 )
                        b = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        b = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        b = (double)System.Double.NegativeInfinity;
                    double v;
                    v = st_analysis.polynomialcalccheb2(a, b, y, 3, t);
                    _TestResult = _TestResult && doc_test_real(v, 6.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_7");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_8
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<5; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{0,0,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    double t = -1;
                    if( _spoil_scenario==3 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.NegativeInfinity;
                    st_analysis.barycentricinterpolant p;
                    double v;
                    st_analysis.polynomialbuildeqdist(0.0, 2.0, y, out p);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_8");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_9
            //
```

КСВ-2025

```
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<11; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{-0.116025,0.000000,1.616025};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    double t = -1;
                    if( _spoil_scenario==3 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.NegativeInfinity;
                    double a = -1;
                    if( _spoil_scenario==5 )
                        a = (double)System.Double.NaN;
                    if( _spoil_scenario==6 )
                        a = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==7 )
                        a = (double)System.Double.NegativeInfinity;
                    double b = +1;
                    if( _spoil_scenario==8 )
                        b = (double)System.Double.NaN;
                    if( _spoil_scenario==9 )
                        b = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==10 )
                        b = (double)System.Double.NegativeInfinity;
                    st_analysis.barycentricinterpolant p;
                    double v;
                    st_analysis.polynomialbuildcheb1(a, b, y, out p);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_9");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_10
            //
            //
            System.Console.WriteLine("50/91");
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<11; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{0,0,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
```

КОПІЯ 2025

```
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    double t = -2;
                    if( _spoil_scenario==3 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.NegativeInfinity;
                    double a = -1;
                    if( _spoil_scenario==5 )
                        a = (double)System.Double.NaN;
                    if( _spoil_scenario==6 )
                        a = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==7 )
                        a = (double)System.Double.NegativeInfinity;
                    double b = +1;
                    if( _spoil_scenario==8 )
                        b = (double)System.Double.NaN;
                    if( _spoil_scenario==9 )
                        b = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==10 )
                        b = (double)System.Double.NegativeInfinity;
                    st_analysis.barycentricinterpolant p;
                    double v;
                    st_analysis.polynomialbuildcheb2(a, b, y, out p);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 6.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_10");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз polint_t_11
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<5; _spoil_scenario++)
            {
                try
                {
                    double[] y = new double[]{0,0,2};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    double t = -1;
                    if( _spoil_scenario==3 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.NegativeInfinity;
                    double v;
                    v = st_analysis.polynomialcalceqdist(0.0, 2.0, y, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
```

КПЗ 2025

```
            { throw; }
        }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "polint_t_11");
        _TotalResult = _TotalResult && _TestResult;


        //
        // Статистичний аналіз polint_t_12
        //
        //
        _TestResult = true;
        for(_spoil_scenario=-1; _spoil_scenario<11; _spoil_scenario++)
        {
            try
            {
                double[] y = new double[]{-0.116025,0.000000,1.616025};
                if( _spoil_scenario==0 )
                    spoil_vector_by_value(ref y, (double)System.Double.NaN);
                if( _spoil_scenario==1 )
                    spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==2 )
                    spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                double t = -1;
                if( _spoil_scenario==3 )
                    t = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==4 )
                    t = (double)System.Double.NegativeInfinity;
                double a = -1;
                if( _spoil_scenario==5 )
                    a = (double)System.Double.NaN;
                if( _spoil_scenario==6 )
                    a = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==7 )
                    a = (double)System.Double.NegativeInfinity;
                double b = +1;
                if( _spoil_scenario==8 )
                    b = (double)System.Double.NaN;
                if( _spoil_scenario==9 )
                    b = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==10 )
                    b = (double)System.Double.NegativeInfinity;
                double v;
                v = st_analysis.polynomialcalccheb1(a, b, y, t);
                _TestResult = _TestResult && doc_test_real(v, 2.0, 0.00005);
                _TestResult = _TestResult && (_spoil_scenario==-1);
            }
            catch(st_analysis.st_analysisexception)
            { _TestResult = _TestResult && (_spoil_scenario!=-1); }
            catch
            { throw; }
        }
        if( !_TestResult)
            System.Console.WriteLine("{0,-32} Помилка", "polint_t_12");
        _TotalResult = _TotalResult && _TestResult;


        //
        // Статистичний аналіз polint_t_13
        //
        //
        _TestResult = true;
        for(_spoil_scenario=-1; _spoil_scenario<11; _spoil_scenario++)
        {
            try
            {
                double[] y = new double[]{0,0,2};
```

КОПІЯ 2025

```
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    double t = -2;
                    if( _spoil_scenario==3 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==4 )
                        t = (double)System.Double.NegativeInfinity;
                    double a = -1;
                    if( _spoil_scenario==5 )
                        a = (double)System.Double.NaN;
                    if( _spoil_scenario==6 )
                        a = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==7 )
                        a = (double)System.Double.NegativeInfinity;
                    double b = +1;
                    if( _spoil_scenario==8 )
                        b = (double)System.Double.NaN;
                    if( _spoil_scenario==9 )
                        b = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==10 )
                        b = (double)System.Double.NegativeInfinity;
                    double v;
                    v = st_analysis.polynomialcalccheb2(a, b, y, t);
                    _TestResult = _TestResult && doc_test_real(v, 6.0, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "polint_t_13");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз spline1d_d_linear
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{-1.0,-0.5,0.0,+0.5,+1.0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_adding_element(ref x);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new double[]{+1.0,0.25,0.0,0.25,+1.0};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
```

КбПЗ_2025

```
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double t = 0.25;
                    if( _spoil_scenario==10 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        t = (double)System.Double.NegativeInfinity;
                    double v;
                    st_analysis.spline1dinterpolant s;

                                        st_analysis.spline1dbuildlinear(x, y, out
s);

                                        v = st_analysis.spline1dcalc(s, t);
                    _TestResult = _TestResult && doc_test_real(v, 0.125,
0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка",
"spline1d_d_linear");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз spline1d_d_cubic
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<10; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{-1.0,-0.5,0.0,+0.5,+1.0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new double[]{+1.0,0.25,0.0,0.25,+1.0};
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_deleting_element(ref y);
                    double t = 0.25;
```

КПЗ_2025

```
                    if( _spoil_scenario==8 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==9 )
                        t = (double)System.Double.NegativeInfinity;
                    double v;
                    st_analysis.spline1dinterpolant s;
                    int natural_bound_type = 2;

                    st_analysis.spline1dbuildcubic(x, y, out s);
                    v = st_analysis.spline1dcalc(s, t);
                    _TestResult = _TestResult && doc_test_real(v, 0.0625,
0.00001);


                    st_analysis.spline1dbuildcubic(x, y, 5, natural_bound_type,
0.0, natural_bound_type, 0.0, out s);
                    v = st_analysis.spline1dcalc(s, t);
                    _TestResult = _TestResult && doc_test_real(v, 0.0580,
0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "spline1d_d_cubic");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз spline1d_d_griddiff
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<10; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{-1.0,-0.5,0.0,+0.5,+1.0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_adding_element(ref x);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new double[]{+1.0,0.25,0.0,0.25,+1.0};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] d1;
                    double[] d2;
```

КБПЗ – 2025

```
                    st_analysis.spline1dgriddiffcubic(x, y, out d1);
                    _TestResult = _TestResult && doc_test_real_vector(d1, new
double[]{-2.0,-1.0,0.0,+1.0,+2.0}, 0.0001);


                    st_analysis.spline1dgriddiff2cubic(x, y, out d1, out d2);
                    _TestResult = _TestResult && doc_test_real_vector(d1, new
double[]{-2.0,-1.0,0.0,+1.0,+2.0}, 0.0001);
                    _TestResult = _TestResult && doc_test_real_vector(d2, new
double[]{2.0,2.0,2.0,2.0,2.0}, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка",
"spline1d_d_griddiff");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз spline1d_d_convdiff
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<11; _spoil_scenario++)
            {
                try
                {

                    double[] x_old = new double[]{-1.0,-0.5,0.0,+0.5,+1.0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x_old,
(double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x_old,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x_old,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref x_old);
                    double[] y_old = new double[]{+1.0,0.25,0.0,0.25,+1.0};
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref y_old,
(double)System.Double.NaN);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y_old,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y_old,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_deleting_element(ref y_old);
                    double[] x_new = new double[]{-1.00,-0.75,-0.50,-
0.25,0.00,+0.25,+0.50,+0.75,+1.00};
                    if( _spoil_scenario==8 )
                        spoil_vector_by_value(ref x_new,
(double)System.Double.NaN);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_value(ref x_new,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==10 )
```

КОПІЯ 2025

```
                        spoil_vector_by_value(ref x_new,
(double)System.Double.NegativeInfinity);
                    double[] y_new;
                    double[] d1_new;
                    double[] d2_new;


                    st_analysis.spline1dconvdiffcubic(x_old, y_old, x_new, out
y_new, out d1_new);
                    _TestResult = _TestResult && doc_test_real_vector(y_new, new
double[]{1.0000,0.5625,0.2500,0.0625,0.0000,0.0625,0.2500,0.5625,1.0000},
0.0001);
                    _TestResult = _TestResult && doc_test_real_vector(d1_new,
new double[]{-2.0,-1.5,-1.0,-0.5,0.0,0.5,1.0,1.5,2.0}, 0.0001);


                    st_analysis.spline1dconvdiff2cubic(x_old, y_old, x_new, out
y_new, out d1_new, out d2_new);
                    _TestResult = _TestResult && doc_test_real_vector(y_new, new
double[]{1.0000,0.5625,0.2500,0.0625,0.0000,0.0625,0.2500,0.5625,1.0000},
0.0001);
                    _TestResult = _TestResult && doc_test_real_vector(d1_new,
new double[]{-2.0,-1.5,-1.0,-0.5,0.0,0.5,1.0,1.5,2.0}, 0.0001);
                    _TestResult = _TestResult && doc_test_real_vector(d2_new,
new double[]{2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0,2.0}, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка",
"spline1d_d_convdiff");
            _TotalResult = _TotalResult && _TestResult;



            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<13; _spoil_scenario++)
            {
                try
                {

                    double[,] a = new double[,]{{2,0},{0,2}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref a);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref a);
                    double[] b = new double[]{-6,-4};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref b, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref b,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref b,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_deleting_element(ref b);
```

КоПЗ_2025

```
                    double[] x0 = new double[]{0,1};
                    if( _spoil_scenario==9 )
                        spoil_vector_by_value(ref x0,
(double)System.Double.NaN);
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref x0,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref x0,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==12 )
                        spoil_vector_by_deleting_element(ref x0);
                    double[] x;
                    st_analysis.minqpstate state;
                    st_analysis.minqpreport rep;

                    st_analysis.minqpcreate(2, out state);
                    st_analysis.minqpsetquadraticterm(state, a);
                    st_analysis.minqpsetlinearterm(state, b);
                    st_analysis.minqpsetstartingpoint(state, x0);
                    st_analysis.minqpoptimize(state);
                    st_analysis.minqpresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{3,2}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minqp_d_u1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minqp_d_bc1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<17; _spoil_scenario++)
            {
                try
                {

                    double[,] a = new double[,]{{2,0},{0,2}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref a, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref a,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref a);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref a);
                    double[] b = new double[]{-6,-4};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref b, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref b,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
```

КбПЗ – 2025

```
                        spoil_vector_by_value(ref b,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_deleting_element(ref b);
                    double[] x0 = new double[]{0,1};
                    if( _spoil_scenario==9 )
                        spoil_vector_by_value(ref x0,
(double)System.Double.NaN);
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref x0,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref x0,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==12 )
                        spoil_vector_by_deleting_element(ref x0);
                    double[] bndl = new double[]{0.0,0.0};
                    if( _spoil_scenario==13 )
                        spoil_vector_by_value(ref bndl,
(double)System.Double.NaN);
                    if( _spoil_scenario==14 )
                        spoil_vector_by_deleting_element(ref bndl);
                    double[] bndu = new double[]{2.5,2.5};
                    if( _spoil_scenario==15 )
                        spoil_vector_by_value(ref bndu,
(double)System.Double.NaN);
                    if( _spoil_scenario==16 )
                        spoil_vector_by_deleting_element(ref bndu);
                    double[] x;
                    st_analysis.minqpstate state;
                    st_analysis.minqpreport rep;

                    st_analysis.minqpcreate(2, out state);
                    st_analysis.minqpsetquadraticterm(state, a);
                    st_analysis.minqpsetlinearterm(state, b);
                    st_analysis.minqpsetstartingpoint(state, x0);
                    st_analysis.minqpsetbc(state, bndl, bndu);
                    st_analysis.minqpoptimize(state);
                    st_analysis.minqpresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{2.5,2}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minqp_d_bc1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlm_d_v
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
```

КСПЗ-2025

```
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlmstate state;
                    st_analysis.minlmreport rep;

                    st_analysis.minlmcreatev(2, x, 0.0001, out state);
                    st_analysis.minlmsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.minlmoptimize(state, function1_fvec, null,
null);
                    st_analysis.minlmresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,+3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlm_d_v");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlm_d_vj
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
```

КОПІЯ_2025

```
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlmstate state;
                    st_analysis.minlmreport rep;

                    st_analysis.minlmcreatevj(2, x, out state);
                    st_analysis.minlmsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.minlmoptimize(state, function1_fvec,
function1_jac, null, null);
                    st_analysis.minlmresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,+3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlm_d_vj");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlm_d_fgh
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
```

КПЗ-2025

```
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlmstate state;
                    st_analysis.minlmreport rep;

                    st_analysis.minlmcreatefgh(x, out state);
                    st_analysis.minlmsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.minlmoptimize(state, function1_func,
function1_grad, function1_hess, null, null);
                    st_analysis.minlmresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,+3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlm_d_fgh");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlm_d_vb
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<16; _spoil_scenario++)
            {
                try
                {

                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double[] bndl = new double[]{-1,-1};
                    if( _spoil_scenario==3 )
```

КПІ_2025

```
                        spoil_vector_by_value(ref bndl,
(double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref bndl);
                    double[] bndu = new double[]{+1,+1};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref bndu,
(double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_deleting_element(ref bndu);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==7 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==8 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==9 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==10 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==11 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==12 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==13 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==14 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==15 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlmstate state;
                    st_analysis.minlmreport rep;

                    st_analysis.minlmcreatev(2, x, 0.0001, out state);
                    st_analysis.minlmsetbc(state, bndl, bndu);
                    st_analysis.minlmsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.minlmoptimize(state, function1_fvec, null,
null);
                    st_analysis.minlmresults(state, out x, out rep);

                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-1,+1}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlm_d_vb");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlm_d_restarts
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<15; _spoil_scenario++)
            {
                try
                {

                    double[] x;
```

КОПІЯ 2025

```
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==0 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==1 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==2 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==3 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==6 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlmstate state;
                    st_analysis.minlmreport rep;


                    x = new double[]{10,10};
                    if( _spoil_scenario==9 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    st_analysis.minlmcreatev(2, x, 0.0001, out state);
                    st_analysis.minlmsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.minlmoptimize(state, function1_fvec, null,
null);
                    st_analysis.minlmresults(state, out x, out rep);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,+3}, 0.005);


                    x = new double[]{4,4};
                    if( _spoil_scenario==12 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==13 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==14 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    st_analysis.minlmrestartfrom(state, x);
                    st_analysis.minlmoptimize(state, function2_fvec, null,
null);
                    st_analysis.minlmresults(state, out x, out rep);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{0,1}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlm_d_restarts");
            _TotalResult = _TotalResult && _TestResult;
```

КОПІЯ 2025

```
            //
            // Статистичний аналіз minlm_t_1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {
                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlmstate state;
                    st_analysis.minlmreport rep;
                    st_analysis.minlmcreatefj(2, x, out state);
                    st_analysis.minlmsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.minlmoptimize(state, function1_func,
function1_jac, null, null);
                    st_analysis.minlmresults(state, out x, out rep);
                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,+3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlm_t_1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз minlm_t_2
            //
            //
```

КНТБ 2025

```
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<12; _spoil_scenario++)
            {
                try
                {
                    double[] x = new double[]{0,0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    double epsg = 0.0000000001;
                    if( _spoil_scenario==3 )
                        epsg = (double)System.Double.NaN;
                    if( _spoil_scenario==4 )
                        epsg = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==5 )
                        epsg = (double)System.Double.NegativeInfinity;
                    double epsf = 0;
                    if( _spoil_scenario==6 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==7 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==8 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0;
                    if( _spoil_scenario==9 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==10 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    st_analysis.minlmstate state;
                    st_analysis.minlmreport rep;
                    st_analysis.minlmcreatefgj(2, x, out state);
                    st_analysis.minlmsetcond(state, epsg, epsf, epsx, maxits);
                    st_analysis.minlmoptimize(state, function1_func,
function1_grad, function1_jac, null, null);
                    st_analysis.minlmresults(state, out x, out rep);
                    _TestResult = _TestResult &&
doc_st_analysisnt(rep.terminationtype, 4);
                    _TestResult = _TestResult && doc_test_real_vector(x, new
double[]{-3,+3}, 0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "minlm_t_2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_nlf
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<27; _spoil_scenario++)
            {
                try
                {
```

КОПІЯ 2025

```
                    double[,] x = new double[,]{{-1},{-0.8},{-0.6},{-0.4},{-
0.2},{0},{0.2},{0.4},{0.6},{0.8},{1.0}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref x);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref x);
                    double[] y = new
double[]{0.223130,0.382893,0.582748,0.786628,0.941765,1.000000,0.941765,0.786628
,0.582748,0.382893,0.223130};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] c = new double[]{0.3};
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref c, (double)System.Double.NaN);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref c,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==12 )
                        spoil_vector_by_value(ref c,
(double)System.Double.NegativeInfinity);
                    double epsf = 0;
                    if( _spoil_scenario==13 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==14 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==15 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0.000001;
                    if( _spoil_scenario==16 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==17 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==18 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    int info;
                    st_analysis.lsfitstate state;
                    st_analysis.lsfitreport rep;
                    double diffstep = 0.0001;
                    if( _spoil_scenario==19 )
                        diffstep = (double)System.Double.NaN;
                    if( _spoil_scenario==20 )
                        diffstep = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==21 )
                        diffstep = (double)System.Double.NegativeInfinity;
                    st_analysis.lsfitcreatef(x, y, c, diffstep, out state);
                    st_analysis.lsfitsetcond(state, epsf, epsx, maxits);
                    st_analysis.lsfitfit(state, function_cx_1_func, null, null);
                    st_analysis.lsfitresults(state, out info, out c, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 2);
```

Kotlin 2025

```
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.5}, 0.05);


                    double[] w = new double[]{1,1,1,1,1,1,1,1,1,1,1};
                    if( _spoil_scenario==22 )
                        spoil_vector_by_value(ref w, (double)System.Double.NaN);
                    if( _spoil_scenario==23 )
                        spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==24 )
                        spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==25 )
                        spoil_vector_by_adding_element(ref w);
                    if( _spoil_scenario==26 )
                        spoil_vector_by_deleting_element(ref w);
                    st_analysis.lsfitcreatewf(x, y, w, c, diffstep, out state);
                    st_analysis.lsfitsetcond(state, epsf, epsx, maxits);
                    st_analysis.lsfitfit(state, function_cx_1_func, null, null);
                    st_analysis.lsfitresults(state, out info, out c, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 2);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.5}, 0.05);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_nlf");
            _TotalResult = _TotalResult && _TestResult;



            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<24; _spoil_scenario++)
            {
                try
                {

                    double[,] x = new double[,]{{-1},{-0.8},{-0.6},{-0.4},{-
0.2},{0},{0.2},{0.4},{0.6},{0.8},{1.0}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref x);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref x);
                    double[] y = new
double[]{0.223130,0.382893,0.582748,0.786628,0.941765,1.000000,0.941765,0.786628
,0.582748,0.382893,0.223130};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
```

КБТЗ_2025

```
                    spoil_vector_by_adding_element(ref y);
                if( _spoil_scenario==9 )
                    spoil_vector_by_deleting_element(ref y);
                double[] c = new double[]{0.3};
                if( _spoil_scenario==10 )
                    spoil_vector_by_value(ref c, (double)System.Double.NaN);
                if( _spoil_scenario==11 )
                    spoil_vector_by_value(ref c,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==12 )
                    spoil_vector_by_value(ref c,
(double)System.Double.NegativeInfinity);
                double epsf = 0;
                if( _spoil_scenario==13 )
                    epsf = (double)System.Double.NaN;
                if( _spoil_scenario==14 )
                    epsf = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==15 )
                    epsf = (double)System.Double.NegativeInfinity;
                double epsx = 0.000001;
                if( _spoil_scenario==16 )
                    epsx = (double)System.Double.NaN;
                if( _spoil_scenario==17 )
                    epsx = (double)System.Double.PositiveInfinity;
                if( _spoil_scenario==18 )
                    epsx = (double)System.Double.NegativeInfinity;
                int maxits = 0;
                int info;
                st_analysis.lsfitstate state;
                st_analysis.lsfitreport rep;


                st_analysis.lsfitcreatefg(x, y, c, true, out state);
                st_analysis.lsfitsetcond(state, epsf, epsx, maxits);
                st_analysis.lsfitfit(state, function_cx_1_func,
function_cx_1_grad, null, null);
                st_analysis.lsfitresults(state, out info, out c, out rep);
                _TestResult = _TestResult && doc_st_analysisnt(info, 2);
                _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.5}, 0.05);


                double[] w = new double[]{1,1,1,1,1,1,1,1,1,1,1};
                if( _spoil_scenario==19 )
                    spoil_vector_by_value(ref w, (double)System.Double.NaN);
                if( _spoil_scenario==20 )
                    spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                if( _spoil_scenario==21 )
                    spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                if( _spoil_scenario==22 )
                    spoil_vector_by_adding_element(ref w);
                if( _spoil_scenario==23 )
                    spoil_vector_by_deleting_element(ref w);
                st_analysis.lsfitcreatewfg(x, y, w, c, true, out state);
                st_analysis.lsfitsetcond(state, epsf, epsx, maxits);
                st_analysis.lsfitfit(state, function_cx_1_func,
function_cx_1_grad, null, null);
                st_analysis.lsfitresults(state, out info, out c, out rep);
                _TestResult = _TestResult && doc_st_analysisnt(info, 2);
                _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.5}, 0.05);
                _TestResult = _TestResult && (_spoil_scenario==-1);
            }
            catch(st_analysis.st_analysisexception)
            { _TestResult = _TestResult && (_spoil_scenario!=-1); }
            catch
            { throw; }
```

КОПИЯ 2025

```
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_nlfg");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_nlfgh
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<24; _spoil_scenario++)
            {
                try
                {

                    double[,] x = new double[,]{{-1},{-0.8},{-0.6},{-0.4},{-
0.2},{0},{0.2},{0.4},{0.6},{0.8},{1.0}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref x);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref x);
                    double[] y = new
double[]{0.223130,0.382893,0.582748,0.786628,0.941765,1.000000,0.941765,0.786628
,0.582748,0.382893,0.223130};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] c = new double[]{0.3};
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref c, (double)System.Double.NaN);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref c,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==12 )
                        spoil_vector_by_value(ref c,
(double)System.Double.NegativeInfinity);
                    double epsf = 0;
                    if( _spoil_scenario==13 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==14 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==15 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0.000001;
                    if( _spoil_scenario==16 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==17 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==18 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
```

КОПІЯ 2025

```
                    int info;
                    st_analysis.lsfitstate state;
                    st_analysis.lsfitreport rep;


                    st_analysis.lsfitcreatefgh(x, y, c, out state);
                    st_analysis.lsfitsetcond(state, epsf, epsx, maxits);
                    st_analysis.lsfitfit(state, function_cx_1_func,
function_cx_1_grad, function_cx_1_hess, null, null);
                    st_analysis.lsfitresults(state, out info, out c, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 2);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.5}, 0.05);


                    double[] w = new double[]{1,1,1,1,1,1,1,1,1,1,1};
                    if( _spoil_scenario==19 )
                        spoil_vector_by_value(ref w, (double)System.Double.NaN);
                    if( _spoil_scenario==20 )
                        spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==21 )
                        spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==22 )
                        spoil_vector_by_adding_element(ref w);
                    if( _spoil_scenario==23 )
                        spoil_vector_by_deleting_element(ref w);
                    st_analysis.lsfitcreatewfgh(x, y, w, c, out state);
                    st_analysis.lsfitsetcond(state, epsf, epsx, maxits);
                    st_analysis.lsfitfit(state, function_cx_1_func,
function_cx_1_grad, function_cx_1_hess, null, null);
                    st_analysis.lsfitresults(state, out info, out c, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 2);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.5}, 0.05);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_nlfgh");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_nlfb
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<26; _spoil_scenario++)
            {
                try
                {

                    double[,] x = new double[,]{{-1},{-0.8},{-0.6},{-0.4},{-
0.2},{0},{0.2},{0.4},{0.6},{0.8},{1.0}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
```

КПЗ-2025

```
                        spoil_matrix_by_deleting_row(ref x);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref x);
                    double[] y = new
double[]{0.223130,0.382893,0.582748,0.786628,0.941765,1.000000,0.941765,0.786628
,0.582748,0.382893,0.223130};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] c = new double[]{0.3};
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref c, (double)System.Double.NaN);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref c,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==12 )
                        spoil_vector_by_value(ref c,
(double)System.Double.NegativeInfinity);
                    double[] bndl = new double[]{0.0};
                    if( _spoil_scenario==13 )
                        spoil_vector_by_value(ref bndl,
(double)System.Double.NaN);
                    if( _spoil_scenario==14 )
                        spoil_vector_by_deleting_element(ref bndl);
                    double[] bndu = new double[]{1.0};
                    if( _spoil_scenario==15 )
                        spoil_vector_by_value(ref bndu,
(double)System.Double.NaN);
                    if( _spoil_scenario==16 )
                        spoil_vector_by_deleting_element(ref bndu);
                    double epsf = 0;
                    if( _spoil_scenario==17 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==18 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==19 )
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 0.000001;
                    if( _spoil_scenario==20 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==21 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==22 )
                        epsx = (double)System.Double.NegativeInfinity;
                    int maxits = 0;
                    int info;
                    st_analysis.lsfitstate state;
                    st_analysis.lsfitreport rep;
                    double diffstep = 0.0001;
                    if( _spoil_scenario==23 )
                        diffstep = (double)System.Double.NaN;
                    if( _spoil_scenario==24 )
                        diffstep = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==25 )
                        diffstep = (double)System.Double.NegativeInfinity;

                    st_analysis.lsfitcreatef(x, y, c, diffstep, out state);
                    st_analysis.lsfitsetbc(state, bndl, bndu);
                    st_analysis.lsfitsetcond(state, epsf, epsx, maxits);
                    st_analysis.lsfitfit(state, function_cx_1_func, null, null);
```

KGU 2025

```
                    st_analysis.lsfitresults(state, out info, out c, out rep);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.0}, 0.05);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_nlfb");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_nlscale
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<30; _spoil_scenario++)
            {
                try
                {

                    double[,] x = new
double[,]{{2000},{2001},{2002},{2003},{2004},{2005},{2006},{2007},{2008}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref x);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref x);
                    double[] y = new
double[]{4323239600000.0,4560913100000.0,5564091500000.0,6743189300000.0,7284064
600000.0,7050129600000.0,7092221500000.0,8483907600000.0,8625804400000.0};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] c = new double[]{1.0e+13,1,1};
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref c, (double)System.Double.NaN);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref c,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==12 )
                        spoil_vector_by_value(ref c,
(double)System.Double.NegativeInfinity);
                    double epsf = 0;
                    if( _spoil_scenario==13 )
                        epsf = (double)System.Double.NaN;
                    if( _spoil_scenario==14 )
                        epsf = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==15 )
```

Київ 2025

```
                        epsf = (double)System.Double.NegativeInfinity;
                    double epsx = 1.0e-5;
                    if( _spoil_scenario==16 )
                        epsx = (double)System.Double.NaN;
                    if( _spoil_scenario==17 )
                        epsx = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==18 )
                        epsx = (double)System.Double.NegativeInfinity;
                    double[] bndl = new double[]{-
System.Double.PositiveInfinity,-10,0.1};
                    if( _spoil_scenario==19 )
                        spoil_vector_by_value(ref bndl,
(double)System.Double.NaN);
                    if( _spoil_scenario==20 )
                        spoil_vector_by_deleting_element(ref bndl);
                    double[] bndu = new
double[]{System.Double.PositiveInfinity,+10,2.0};
                    if( _spoil_scenario==21 )
                        spoil_vector_by_value(ref bndu,
(double)System.Double.NaN);
                    if( _spoil_scenario==22 )
                        spoil_vector_by_deleting_element(ref bndu);
                    double[] s = new double[]{1.0e+12,1,1};
                    if( _spoil_scenario==23 )
                        spoil_vector_by_value(ref s, (double)System.Double.NaN);
                    if( _spoil_scenario==24 )
                        spoil_vector_by_value(ref s,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==25 )
                        spoil_vector_by_value(ref s,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==26 )
                        spoil_vector_by_deleting_element(ref s);
                    int maxits = 0;
                    int info;
                    st_analysis.lsfitstate state;
                    st_analysis.lsfitreport rep;
                    double diffstep = 1.0e-5;
                    if( _spoil_scenario==27 )
                        diffstep = (double)System.Double.NaN;
                    if( _spoil_scenario==28 )
                        diffstep = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==29 )
                        diffstep = (double)System.Double.NegativeInfinity;

                    st_analysis.lsfitcreatef(x, y, c, diffstep, out state);
                    st_analysis.lsfitsetcond(state, epsf, epsx, maxits);
                    st_analysis.lsfitsetbc(state, bndl, bndu);
                    st_analysis.lsfitsetscale(state, s);
                    st_analysis.lsfitfit(state, function_debt_func, null, null);
                    st_analysis.lsfitresults(state, out info, out c, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 2);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{4.142560e+12,0.434240,0.565376}, -0.005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_nlscale");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_lin
            //
```

КОПІЯ 2025

```
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<13; _spoil_scenario++)
            {
                try
                {

                    double[,] fmatrix = new
double[,]{{0.606531},{0.670320},{0.740818},{0.818731},{0.904837},{1.000000},{1.1
05171},{1.221403},{1.349859},{1.491825},{1.648721}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref fmatrix,
(double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref fmatrix,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref fmatrix,
(double)System.Double.NegativeInfinity);
                    double[] y = new
double[]{1.133719,1.306522,1.504604,1.554663,1.884638,2.072436,2.257285,2.534068
,2.622017,2.897713,3.219371};
                    if( _spoil_scenario==3 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_deleting_element(ref y);
                    int info;
                    double[] c;
                    st_analysis.lsfitreport rep;


                    st_analysis.lsfitlinear(y, fmatrix, out info, out c, out
rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.98650}, 0.00005);



                    double[] w = new double[]{1.414213,1,1,1,1,1,1,1,1,1,1};
                    if( _spoil_scenario==8 )
                        spoil_vector_by_value(ref w, (double)System.Double.NaN);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_adding_element(ref w);
                    if( _spoil_scenario==12 )
                        spoil_vector_by_deleting_element(ref w);
                    st_analysis.lsfitlinearw(y, w, fmatrix, out info, out c, out
rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{1.983354}, 0.00005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
```

КЭП - 2025

```
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_lin");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_linc
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<20; _spoil_scenario++)
            {
                try
                {

                    double[] y = new
double[]{0.072436,0.246944,0.491263,0.522300,0.714064,0.921929};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref y);
                    double[,] fmatrix = new
double[,]{{1,0.0},{1,0.2},{1,0.4},{1,0.6},{1,0.8},{1,1.0}};
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_value(ref fmatrix,
(double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_value(ref fmatrix,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_matrix_by_value(ref fmatrix,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_matrix_by_adding_row(ref fmatrix);
                    if( _spoil_scenario==9 )
                        spoil_matrix_by_adding_col(ref fmatrix);
                    if( _spoil_scenario==10 )
                        spoil_matrix_by_deleting_row(ref fmatrix);
                    if( _spoil_scenario==11 )
                        spoil_matrix_by_deleting_col(ref fmatrix);
                    double[,] cmatrix = new double[,]{{1,0,0}};
                    if( _spoil_scenario==12 )
                        spoil_matrix_by_value(ref cmatrix,
(double)System.Double.NaN);
                    if( _spoil_scenario==13 )
                        spoil_matrix_by_value(ref cmatrix,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==14 )
                        spoil_matrix_by_value(ref cmatrix,
(double)System.Double.NegativeInfinity);
                    int info;
                    double[] c;
                    st_analysis.lsfitreport rep;



                    st_analysis.lsfitlinearc(y, fmatrix, cmatrix, out info, out
c, out rep);
```

КБІЗ-2025

```
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{0,0.932933}, 0.0005);


                    double[] w = new double[]{1,1.414213,1,1,1,1};
                    if( _spoil_scenario==15 )
                        spoil_vector_by_value(ref w, (double)System.Double.NaN);
                    if( _spoil_scenario==16 )
                        spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==17 )
                        spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==18 )
                        spoil_vector_by_adding_element(ref w);
                    if( _spoil_scenario==19 )
                        spoil_vector_by_deleting_element(ref w);
                    st_analysis.lsfitlinearwc(y, w, fmatrix, cmatrix, out info,
out c, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    _TestResult = _TestResult && doc_test_real_vector(c, new
double[]{0,0.938322}, 0.0005);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_linc");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_pol
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<20; _spoil_scenario++)
            {
                try
                {

                    double[] x = new
double[]{0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_adding_element(ref x);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new
double[]{0.00,0.05,0.26,0.32,0.33,0.43,0.60,0.60,0.77,0.98,1.02};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
```

КоПЗ_2025

```
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    int m = 2;
                    double t = 2;
                    if( _spoil_scenario==10 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==11 )
                        t = (double)System.Double.NegativeInfinity;
                    int info;
                    st_analysis.barycentricinterpolant p;
                    st_analysis.polynomialfitreport rep;
                    double v;


                    st_analysis.polynomialfit(x, y, m, out info, out p, out
rep);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.011, 0.002);


                    double[] w = new double[]{1,1.414213562,1,1,1,1,1,1,1,1,1};
                    if( _spoil_scenario==12 )
                        spoil_vector_by_value(ref w, (double)System.Double.NaN);
                    if( _spoil_scenario==13 )
                        spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==14 )
                        spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==15 )
                        spoil_vector_by_adding_element(ref w);
                    if( _spoil_scenario==16 )
                        spoil_vector_by_deleting_element(ref w);
                    double[] xc = new double[0];
                    if( _spoil_scenario==17 )
                        spoil_vector_by_adding_element(ref xc);
                    double[] yc = new double[0];
                    if( _spoil_scenario==18 )
                        spoil_vector_by_adding_element(ref yc);
                    int[] dc = new int[0];
                    if( _spoil_scenario==19 )
                        spoil_vector_by_adding_element(ref dc);
                    st_analysis.polynomialfitwc(x, y, w, xc, yc, dc, m, out
info, out p, out rep);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.023, 0.002);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_pol");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_polc
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<29; _spoil_scenario++)
            {
                try
                {
```

КОПІЯ 2025

```
                    double[] x = new double[]{1.0,1.0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_adding_element(ref x);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new double[]{0.9,1.1};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] w = new double[]{1,1};
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref w, (double)System.Double.NaN);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==12 )
                        spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==13 )
                        spoil_vector_by_adding_element(ref w);
                    if( _spoil_scenario==14 )
                        spoil_vector_by_deleting_element(ref w);
                    double[] xc = new double[]{0};
                    if( _spoil_scenario==15 )
                        spoil_vector_by_value(ref xc,
(double)System.Double.NaN);
                    if( _spoil_scenario==16 )
                        spoil_vector_by_value(ref xc,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==17 )
                        spoil_vector_by_value(ref xc,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==18 )
                        spoil_vector_by_adding_element(ref xc);
                    if( _spoil_scenario==19 )
                        spoil_vector_by_deleting_element(ref xc);
                    double[] yc = new double[]{0};
                    if( _spoil_scenario==20 )
                        spoil_vector_by_value(ref yc,
(double)System.Double.NaN);
                    if( _spoil_scenario==21 )
                        spoil_vector_by_value(ref yc,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==22 )
                        spoil_vector_by_value(ref yc,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==23 )
                        spoil_vector_by_adding_element(ref yc);
                    if( _spoil_scenario==24 )
                        spoil_vector_by_deleting_element(ref yc);
                    int[] dc = new int[]{0};
```

KGS 2025

```
                    if( _spoil_scenario==25 )
                        spoil_vector_by_adding_element(ref dc);
                    if( _spoil_scenario==26 )
                        spoil_vector_by_deleting_element(ref dc);
                    double t = 2;
                    if( _spoil_scenario==27 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==28 )
                        t = (double)System.Double.NegativeInfinity;
                    int m = 2;
                    int info;
                    st_analysis.barycentricinterpolant p;
                    st_analysis.polynomialfitreport rep;
                    double v;

                    st_analysis.polynomialfitwc(x, y, w, xc, yc, dc, m, out
info, out p, out rep);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.000, 0.001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_polc");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_d_spline
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<19; _spoil_scenario++)
            {
                try
                {
                    //
                    // порушення сплайнів зашумлених даних
                    //
                    // Ми маємо:
                    // * x - абсцис
                    // * y - вектор експериментальних даних, прямої з невеликим
шумом
                    //
                    double[] x = new
double[]{0.00,0.10,0.20,0.30,0.40,0.50,0.60,0.70,0.80,0.90};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_adding_element(ref x);
                    if( _spoil_scenario==4 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new
double[]{0.10,0.00,0.30,0.40,0.30,0.40,0.62,0.68,0.75,0.95};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==7 )
```

КОПЗ – 2025

```
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_adding_element(ref y);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_deleting_element(ref y);
                    int info;
                    double v;
                    st_analysis.spline1dinterpolant s;
                    st_analysis.spline1dfitreport rep;
                    double rho;


                    rho = -5.0;
                    if( _spoil_scenario==10 )
                        rho = (double)System.Double.NaN;
                    if( _spoil_scenario==11 )
                        rho = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==12 )
                        rho = (double)System.Double.NegativeInfinity;
                    st_analysis.spline1dfitpenalized(x, y, 50, rho, out info,
out s, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    v = st_analysis.spline1dcalc(s, 0.0);
                    _TestResult = _TestResult && doc_test_real(v, 0.10, 0.01);



                    rho = +10.0;
                    if( _spoil_scenario==13 )
                        rho = (double)System.Double.NaN;
                    if( _spoil_scenario==14 )
                        rho = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==15 )
                        rho = (double)System.Double.NegativeInfinity;
                    st_analysis.spline1dfitpenalized(x, y, 50, rho, out info,
out s, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    v = st_analysis.spline1dcalc(s, 1.0);
                    _TestResult = _TestResult && doc_test_real(v, 0.969, 0.001);


                    rho = +3.0;
                    if( _spoil_scenario==16 )
                        rho = (double)System.Double.NaN;
                    if( _spoil_scenario==17 )
                        rho = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==18 )
                        rho = (double)System.Double.NegativeInfinity;
                    st_analysis.spline1dfitpenalized(x, y, 50, rho, out info,
out s, out rep);
                    _TestResult = _TestResult && doc_st_analysisnt(info, 1);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_d_spline");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_t_polfit_1
            //
            //
            _TestResult = true;
```

КПЗ_2025

```
            for(_spoil_scenario=-1; _spoil_scenario<10; _spoil_scenario++)
            {
                try
                {
                    double[] x = new
double[]{0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new
double[]{0.00,0.05,0.26,0.32,0.33,0.43,0.60,0.60,0.77,0.98,1.02};
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_deleting_element(ref y);
                    int m = 2;
                    double t = 2;
                    if( _spoil_scenario==8 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==9 )
                        t = (double)System.Double.NegativeInfinity;
                    int info;
                    st_analysis.barycentricinterpolant p;
                    st_analysis.polynomialfitreport rep;
                    double v;
                    st_analysis.polynomialfit(x, y, 11, m, out info, out p, out
rep);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.011, 0.002);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_t_polfit_1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_t_polfit_2
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<14; _spoil_scenario++)
            {
                try
                {
                    double[] x = new
double[]{0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0};
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
```

КСТЗ-2025

```
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new
double[]{0.00,0.05,0.26,0.32,0.33,0.43,0.60,0.60,0.77,0.98,1.02};
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] w = new double[]{1,1.414213562,1,1,1,1,1,1,1,1,1};
                    if( _spoil_scenario==8 )
                        spoil_vector_by_value(ref w, (double)System.Double.NaN);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_deleting_element(ref w);
                    double[] xc = new double[0];
                    double[] yc = new double[0];
                    int[] dc = new int[0];
                    int m = 2;
                    double t = 2;
                    if( _spoil_scenario==12 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==13 )
                        t = (double)System.Double.NegativeInfinity;
                    int info;
                    st_analysis.barycentricinterpolant p;
                    st_analysis.polynomialfitreport rep;
                    double v;
                    st_analysis.polynomialfitwc(x, y, w, 11, xc, yc, dc, 0, m,
out info, out p, out rep);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.023, 0.002);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_t_polfit_2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз lsfit_t_polfit_3
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<23; _spoil_scenario++)
            {
                try
                {
                    double[] x = new double[]{1.0,1.0};
```

КОПІЯ 2025

```
                    if( _spoil_scenario==0 )
                        spoil_vector_by_value(ref x, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_vector_by_value(ref x,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_vector_by_value(ref x,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_vector_by_deleting_element(ref x);
                    double[] y = new double[]{0.9,1.1};
                    if( _spoil_scenario==4 )
                        spoil_vector_by_value(ref y, (double)System.Double.NaN);
                    if( _spoil_scenario==5 )
                        spoil_vector_by_value(ref y,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==6 )
                        spoil_vector_by_value(ref y,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==7 )
                        spoil_vector_by_deleting_element(ref y);
                    double[] w = new double[]{1,1};
                    if( _spoil_scenario==8 )
                        spoil_vector_by_value(ref w, (double)System.Double.NaN);
                    if( _spoil_scenario==9 )
                        spoil_vector_by_value(ref w,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==10 )
                        spoil_vector_by_value(ref w,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==11 )
                        spoil_vector_by_deleting_element(ref w);
                    double[] xc = new double[]{0};
                    if( _spoil_scenario==12 )
                        spoil_vector_by_value(ref xc,
(double)System.Double.NaN);
                    if( _spoil_scenario==13 )
                        spoil_vector_by_value(ref xc,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==14 )
                        spoil_vector_by_value(ref xc,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==15 )
                        spoil_vector_by_deleting_element(ref xc);
                    double[] yc = new double[]{0};
                    if( _spoil_scenario==16 )
                        spoil_vector_by_value(ref yc,
(double)System.Double.NaN);
                    if( _spoil_scenario==17 )
                        spoil_vector_by_value(ref yc,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==18 )
                        spoil_vector_by_value(ref yc,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==19 )
                        spoil_vector_by_deleting_element(ref yc);
                    int[] dc = new int[]{0};
                    if( _spoil_scenario==20 )
                        spoil_vector_by_deleting_element(ref dc);
                    int m = 2;
                    double t = 2;
                    if( _spoil_scenario==21 )
                        t = (double)System.Double.PositiveInfinity;
                    if( _spoil_scenario==22 )
                        t = (double)System.Double.NegativeInfinity;
                    int info;
                    st_analysis.barycentricinterpolant p;
                    st_analysis.polynomialfitreport rep;
                    double v;
```

KGB-2025

```
                    st_analysis.polynomialfitwc(x, y, w, 2, xc, yc, dc, 1, m,
out info, out p, out rep);
                    v = st_analysis.barycentriccalc(p, t);
                    _TestResult = _TestResult && doc_test_real(v, 2.000, 0.001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "lsfit_t_polfit_3");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matdet_d_1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<7; _spoil_scenario++)
            {
                try
                {
                    double[,] b = new double[,]{{1,2},{2,1}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref b);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_col(ref b);
                    double a;
                    a = st_analysis.rmatrixdet(b);
                    _TestResult = _TestResult && doc_test_real(a, -3, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_d_1");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matdet_d_2
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<5; _spoil_scenario++)
            {
                try
                {
                    double[,] b = new double[,]{{5,4},{4,5}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b, (double)System.Double.NaN);
```

КБ-3 2025

```
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref b);
                    double a;
                    a = st_analysis.rmatrixdet(b, 2);
                    _TestResult = _TestResult && doc_test_real(a, 9, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_d_2");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matdet_d_3
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<7; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex[,] b = new st_analysis.complex[,]{{new
st_analysis.complex(1,+1),2},{2,new st_analysis.complex(1,-1)}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref b);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_col(ref b);
                    st_analysis.complex a;
                    a = st_analysis.cmatrixdet(b);
                    _TestResult = _TestResult && doc_test_complex(a, -2,
0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_d_3");
            _TotalResult = _TotalResult && _TestResult;
```

КСПБ-2025

```
            //
            // Статистичний аналіз matdet_d_4
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<5; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex a;
                    st_analysis.complex[,] b = new st_analysis.complex[,]{{new
st_analysis.complex(0,5),4},{new st_analysis.complex(0,4),5}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref b);
                    a = st_analysis.cmatrixdet(b, 2);
                    _TestResult = _TestResult && doc_test_complex(a, new
st_analysis.complex(0,9), 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_d_4");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matdet_d_5
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<7; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex a;
                    st_analysis.complex[,] b = new
st_analysis.complex[,]{{9,1},{2,1}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref b);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_col(ref b);
```

КОПІЯ 2025

```
                    a = st_analysis.cmatrixdet(b);
                    _TestResult = _TestResult && doc_test_complex(a, 7, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_d_5");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matdet_t_0
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<5; _spoil_scenario++)
            {
                try
                {
                    double a;
                    double[,] b = new double[,]{{3,4},{-4,3}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref b);
                    a = st_analysis.rmatrixdet(b, 2);
                    _TestResult = _TestResult && doc_test_real(a, 25, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_t_0");
            _TotalResult = _TotalResult && _TestResult;


            //
            // Статистичний аналіз matdet_t_1
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<9; _spoil_scenario++)
            {
                try
                {
                    double a;
                    double[,] b = new double[,]{{1,2},{2,5}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
```

КОПІЯ 2025

```
                        spoil_matrix_by_value(ref b,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref b);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_col(ref b);
                    int[] p = new int[]{1,1};
                    if( _spoil_scenario==7 )
                        spoil_vector_by_adding_element(ref p);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_deleting_element(ref p);
                    a = st_analysis.rmatrixludet(b, p);
                    _TestResult = _TestResult && doc_test_real(a, -5, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_t_1");
            _TotalResult = _TotalResult && _TestResult;
            //
            // Статистичний аналіз matdet_t_2
            //
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {
                    double a;
                    double[,] b = new double[,]{{5,4},{4,5}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b, (double)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(double)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref b);
                    int[] p = new int[]{0,1};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_deleting_element(ref p);
                    a = st_analysis.rmatrixludet(b, p, 2);
                    _TestResult = _TestResult && doc_test_real(a, 25, 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_t_2");
            _TotalResult = _TotalResult && _TestResult;
            //
            // Статистичний аналіз matdet_t_3
            //      Визначник розрахунку, складні матриці, повна форма
            //
```

КБТЗ_2025

```
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<5; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex a;
                    st_analysis.complex[,] b = new st_analysis.complex[,]{{new
st_analysis.complex(0,5),4},{-4,new st_analysis.complex(0,5)}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref b);
                    a = st_analysis.cmatrixdet(b, 2);
                    _TestResult = _TestResult && doc_test_complex(a, -9,
0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_t_3");
            _TotalResult = _TotalResult && _TestResult;
            //
            // Статистичний аналіз matdet_t_4
            //      Визначник розрахунку, складні матриці, LU, коротка форма
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<9; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex a;
                    st_analysis.complex[,] b = new
st_analysis.complex[,]{{1,2},{2,new st_analysis.complex(0,5)}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_adding_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_adding_col(ref b);
                    if( _spoil_scenario==5 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==6 )
                        spoil_matrix_by_deleting_col(ref b);
                    int[] p = new int[]{1,1};
                    if( _spoil_scenario==7 )
                        spoil_vector_by_adding_element(ref p);
                    if( _spoil_scenario==8 )
                        spoil_vector_by_deleting_element(ref p);
                    a = st_analysis.cmatrixludet(b, p);
```

КОПЗ-2025

```
                    _TestResult = _TestResult && doc_test_complex(a, new
st_analysis.complex(0,-5), 0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_t_4");
            _TotalResult = _TotalResult && _TestResult;
            //
            // Статистичний аналіз matdet_t_5
            //      Визначник розрахунку, складні матриці, LU, повна форма
            //
            _TestResult = true;
            for(_spoil_scenario=-1; _spoil_scenario<6; _spoil_scenario++)
            {
                try
                {
                    st_analysis.complex a;
                    st_analysis.complex[,] b = new st_analysis.complex[,]{{5,new
st_analysis.complex(0,4)},{4,5}};
                    if( _spoil_scenario==0 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NaN);
                    if( _spoil_scenario==1 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.PositiveInfinity);
                    if( _spoil_scenario==2 )
                        spoil_matrix_by_value(ref b,
(st_analysis.complex)System.Double.NegativeInfinity);
                    if( _spoil_scenario==3 )
                        spoil_matrix_by_deleting_row(ref b);
                    if( _spoil_scenario==4 )
                        spoil_matrix_by_deleting_col(ref b);
                    int[] p = new int[]{0,1};
                    if( _spoil_scenario==5 )
                        spoil_vector_by_deleting_element(ref p);
                    a = st_analysis.cmatrixludet(b, p, 2);
                    _TestResult = _TestResult && doc_test_complex(a, 25,
0.0001);
                    _TestResult = _TestResult && (_spoil_scenario==-1);
                }
                catch(st_analysis.st_analysisexception)
                { _TestResult = _TestResult && (_spoil_scenario!=-1); }
                catch
                { throw; }
            }
            if( !_TestResult)
                System.Console.WriteLine("{0,-32} Помилка", "matdet_t_5");
            _TotalResult = _TotalResult && _TestResult;
            System.Console.WriteLine("91/91");
        }
        catch
        {
            System.Console.WriteLine("Необроблена виняткова ситуація!");
            return 1;
        }
        return _TotalResult ? 0 : 1;
    }
}
```

КОПІЯ 2025