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
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22М-1
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Одинцов Є.Д.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Одинцову Єгору Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 32-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Одинцов Є.Д. Дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи статистичного аналізу та фільтрації даних зі змінних носіїв.

Метою розробки є дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв.

Об'єктом дослідження є процес статистичного аналізу та фільтрації даних зі змінних носіїв.

Предметом дослідження є методи статистичного аналізу та фільтрації даних зі змінних носіїв.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв.

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 архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Visual C#.

Ключові слова: комп'ютерні науки, статистичний аналіз, фільтрація даних, змінні носії

ABSTRACT

Odyntsov E.D. Research and software implementation of a system of statistical analysis and filtering of data from variable media. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the system of statistical analysis and filtering of data from removable media.

The goal of development is research and software implementation of a system of statistical analysis and filtering of data from removable media.

The object of the study is the process of statistical analysis and filtering of data from variable media.

The subject of the study is methods of statistical analysis and filtering of data from variable media.

Research methods are based on information protection methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the system of statistical analysis and filtering of data from removable media.

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 convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Visual C# environment.

Keywords: computer science, statistical analysis, data filtering, variable media

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. Електронні стеганографічні методи можуть бути використані для кодування сигналу керування керуванням правами в інформаційний сигнал, що передається по незахищеному каналу зв'язку за допомогою змінних носіїв. Стеганографічні методи гарантують, що цифрова керуюча інформація практично непомітно і практично незмивно передається інформаційним сигналом. Ці методи можуть забезпечити наскрізний захист прав керування інформаційним сигналом незалежно від перетворень між аналоговим і цифровим. Електронний пристрій може відновлювати керуючу інформацію та використовувати її для електронного керування правами для забезпечення сумісності з віртуальним середовищем розповсюдження. В одному прикладі система кодує покажчики низької швидкості передачі даних у періоди часу сигналу вмісту з високою пропускною здатністю, щоб покращити загальний час читання/пошуку керуючої інформації.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв.

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

– Огляд існуючих систем статистичного аналізу та фільтрації даних зі змінних носіїв.

– Дослідження системи статистичного аналізу та фільтрації даних зі змінних носіїв.

– Програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв.

Об'єктом дослідження є процес статистичного аналізу та фільтрації даних зі змінних носіїв.

Предметом дослідження є методи статистичного аналізу та фільтрації даних зі змінних носіїв.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод статистичного аналізу та фільтрації даних зі змінних носіїв.

– Розроблено вітчизняний продукт статистичного аналізу та фільтрації даних зі змінних носіїв, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі статистичного аналізу та фільтрації даних зі змінних носіїв.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для виявлення несанкціонованого витоку конфіденційної інформації методом статистичного аналізу та фільтрації даних зі змінних носіїв з використанням стегоаналізу.

Стеганографічні методи дозволяють передавати секретні повідомлення, убудовані в контейнери, таким чином, що неможливо виявити навіть сам факт передачі.

Вбудовування повідомлень приводить до змін деяких властивостей контейнерів. При порушенні природності контейнера зміни можуть бути виявлені навіть неозброєним оком (якщо контейнер – зображення). Назве це суб'єктивним стегоаналізом.

З іншого боку, якщо зміни, що супроводжують вбудовування повідомлення в контейнер, не можуть бути виявлені самим атакуючої, те можливе застосування спеціальних програмних засобів (програмний стегоаналіз).

Фахівців в області стегоаналізу (за аналогією із криптоаналізом) будемо називати стегоаналітиками або, інакше, що атакують (зловмисниками), а спроби стегоаналітиків виявити, витягти або видалити убудоване повідомлення – атаками. Розглянуті в даній роботі погрози й атаки рівною мірою можуть бути застосовані як до стегосистемам, так і до цифрових водяних знаків. Справа в тому, що ці два напрямки мають загальні коріння й іноді невидимі цифрові водяні знаки трактуються як галузь стеганографії (або один з додатків). У ряді випадків, дійсно, для вбудовування цифрового водяного знака використовуються ті ж методи, що й для вбудовування секретного повідомлення. Правда, у силу особливостей реалізації, методи вбудовування водяних знаків менш піддані таким впливам, як геометричні перетворення або деякі операції обробки зображень, при яких змінюються молодші біти.

1.2 Область застосування

На жаль, надсилання цифрової інформації за допомогою таких відомих методів цифрового кодування є проблематичним, оскільки цифрова інформація не є постійною. Відносно легко видалити або видалити цифрову інформацію, закодовану з використанням попередніх методів, які зазвичай використовуються для накладення цифрових сигналів на аналоговий інформаційний сигнал. Аналогові канали зв'язку зазвичай можуть піддаватися різноманітній обробці сигналу, яка може (навмисно чи ненавмисно) видаляти цифрову інформацію, додану до аналогового сигналу, руйнуючи будь-яку систему, процес або техніку, яка залежить від наявності та читабельності цифрової інформації. Наприклад, телевізійний сигнал вертикального гасіння разом із будь-якими компонентами сигналу, розташованими в межах інтервалу вертикального гасіння, зазвичай усувається щоразу, коли відеосигнал обробляється комп'ютером.

Спроба використати небезпечні методи для забезпечення керування правами в кращому випадку неефективна, і може бути гіршою, ніж відсутність керування правами взагалі. Недобросовісні люди можуть повністю видалити незахищену керуючу інформацію, щоб відповідний інформаційний сигнал взагалі не підлягав контролю, наприклад, порушуючи механізми захисту від копіювання та дозволяючи користувачам уникати оплати за використання прав. Ще більш підлим є те, що недобросовісна особа може змінити незахищену систему, замінивши неправдиву інформацію керування замість належної інформації. Такі заміни можуть, наприклад, перенаправляти платежі комусь, крім законних власників прав, сприяючи електронному шахрайству та крадіжці.

Попередні незахищені методи не змогли вирішити загальну проблему щодо того, як забезпечити та безпечно керувати розширеним автоматичним електронним керуванням правами для аналогових та інших інформаційних сигналів, що передаються по незахищеному каналу зв'язку. Відсутність надійного керування правами для аналогових сигналів створює величезну

прогалину в будь-якій комплексній стратегії керування електронними правами та дає змогу споживачам та іншим особам обійти – принаймні певною мірою – навіть найпотужніші технології керування цифровими правами. Отже, існує реальна потреба в бездоганній інтеграції аналогових моделей доставки з сучасними електронними цифровими методами керування правами.

Дана робота вирішує ці та інші проблеми, забезпечуючи «наскрізний» надійний захист керування правами, що дозволяє постачальникам вмісту та власникам прав бути впевненими, що їхній вміст буде належним чином захищений – незалежно від типів пристроїв, форматів сигналізації та характеру обробки сигналів у ланцюжку розподілу контенту. Цей наскрізний захист також дозволяє авторизованим аналоговим пристроям легко, плавно та економічно ефективно інтегрувати в сучасну архітектуру керування цифровими правами.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Зараз уже існує досить велика кількість програм, що використовують стеганографію й комп'ютерні зображення як контейнери. Зупинимося на деяких з них, найпоширеніших. Всі ці програми в основному використовують алгоритми, засновані на впровадженні повідомлення в НЗБ контейнера.

S-Tools

За допомогою програми S-Tools (Steganography Tools) (рисунок 2.1), що має статус freeware, можна сховати інформацію в графічному або звуковому файлі. Причому графічний файл після цього можна спокійно переглянути, а звуковий – прослухати. Утиліта не вимагає інсталяції, досить розпакувати архів і запустити файл s-tools.exe. Архів програми займає всього лише порядку 280 Kb.

Технологія роботи програми така, що шифруємі дані спочатку стискуються, а вже потім безпосередньо шифруються. Програма може використовувати кілька різних алгоритмів шифрування даних залежно від бажання користувача, включаючи одні із кращих алгоритмів – DES, що сьогодні вже не задовольняє сучасним вимогам безпеки, Triple DES і IDEA. Останні два алгоритми забезпечують високий рівень захисту даних від дешифрування (дотепер не було зареєстровано жодного випадку дешифрування інформації, зашифрованої з використанням даних методів).

Сам процес шифрування інформації дуже простий: Для цього досить із провідника Windows перетягнути графічний або звуковий файл у вікно програми. У правому нижньому куті програми з'явиться інформація про розмір файлу, якому можна сховати. На наступному етапі потрібно перетягнути файл із

інформацією на зображення, увести пароль, вибрати варіант шифрування й визначити метод приховання. Через якийсь час програма видасть другу картинку з умовним ім'ям hidden data, що вже містить сховану інформацію. Потім варто зберегти нову картинку з конкретним ім'ям і розширенням gif або bmp, вибравши команду «Save as».

Рисунок 2.1 – Основне вікно програми S-Tools

Для розшифровки інформації потрібно перетягнути у вікно програми картинку зі схованою інформацією, вибрати з контекстного меню, викликуваного натисканням правої кнопки миші, команду «Reveal», потім увести пароль – і на екрані з'явиться додаткове вікно з ім'ям схованого файлу.

Steganos Security Suite

Програма Steganos Security Suite (рисунок 2.2) також є досить популярною програмою, по якості переважаючою S-Tools, однак не є безкоштовною. Даний програмний продукт являє собою універсальний набір засобів, необхідних для методу збереження конфіденційності інформації.

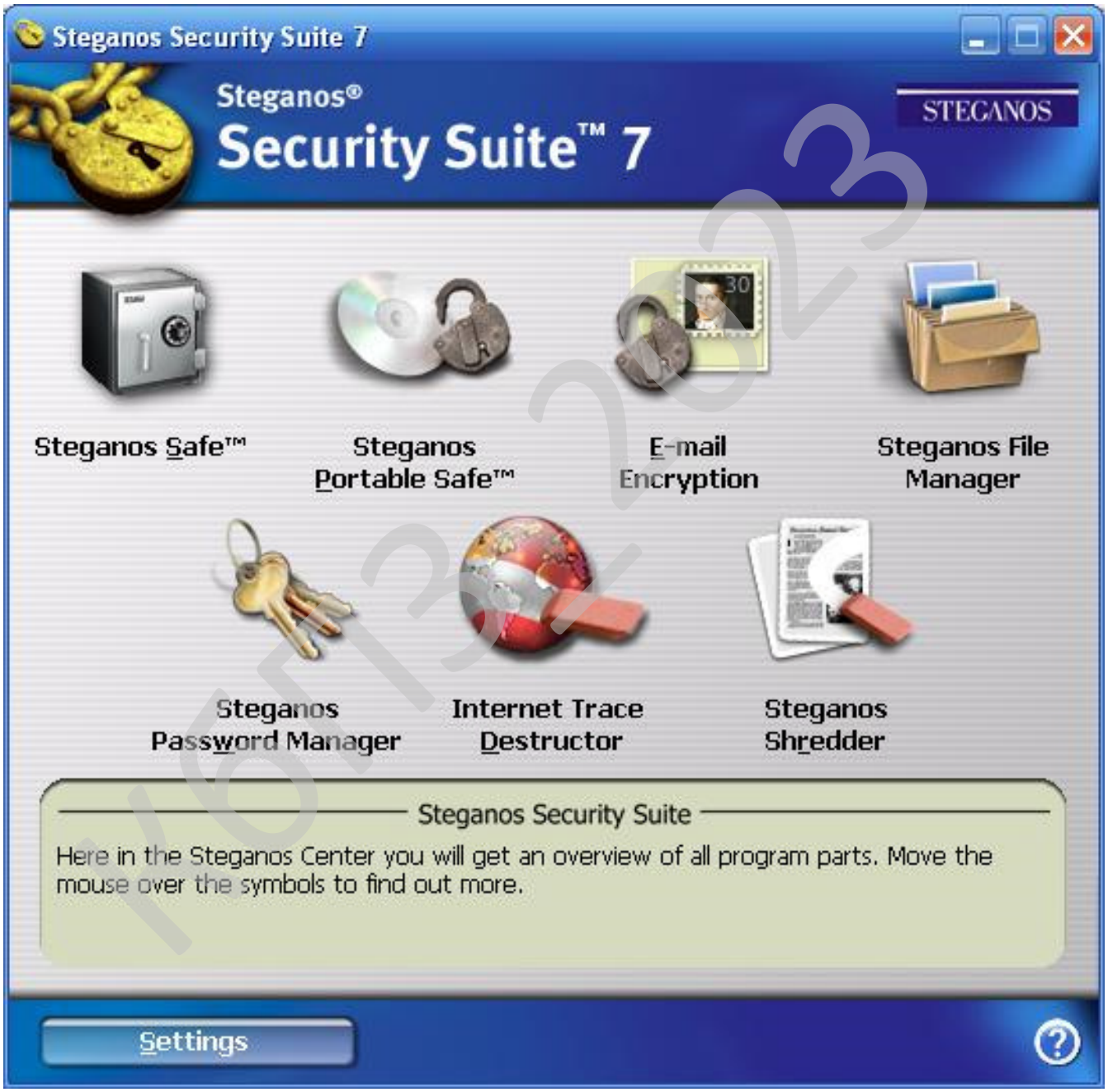

Рисунок 2.2 – Основне вікно програми Steganos

Програма дозволяє організовувати віртуальні зашифровані диски, шифрувати повідомлення електронної пошти, надійно видаляти файли з жорсткого диска й багато чого іншого. У більшість із можливостей, надаваних Steganos, убудовані стеганографічні методи. При шифруванні якого-небудь файлу можна додатково до цього вибрати контейнер (зображення формату BMP, JPEG або аудіофайл WAV), у який буде убудований попередньо стислий і зашифрований файл. Відносно формату BMP програма дозволяє використовувати зображення тільки в режимі True Color.

Secur Engine

Програма Secur Engine (рисунок 2.3) дозволяє як просто шифрувати файли з використанням криптографічних методів, так і вбудовувати їх у контейнери форматів BMP,JPEG, WAV. Є можливість вибрати один з 6 алгоритмів шифрування, одним із яких є вітчизняний алгоритм ДСТ.

Рисунок 2.3 – Основне вікно програми Secur Engine

Весь процес приховання й шифрування виконаний у формі майстра. Користувачеві пропонується послідовно вибрати файли, які йому необхідно сховати, алгоритм шифрування, файл-контейнер, у який будуть впроваджені дані, і ім'я контейнера, що виходить, із впровадженим повідомленням.

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 застосунків, виконуваних у середовищі .NET Framework. Мовою Visual C# можна розробляти звичайні клієнтські застосунки Windows, веб-служби XML, розподілені компоненти, застосунки типу " сервер-клієнт", застосунки баз даних і багато яких інших. В Visual C# є розширений редактор коду, конструктори зі зручним користувальницьким інтерфейсом, вбудований відладник і багато інших засобів, покликані спростити розробку застосунків 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, 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` – крапку входу застосунка – інкапсулюється 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, 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 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.

Крім служб часу виконання, в.NET Framework також є велика бібліотека, що складається з більш ніж 4000 класів, організованих по просторах імен, які забезпечують різноманітні корисні функції для будь-яких дій, починаючи від введення й виведення файлів для керуванням рядками для розбивки XML, і закінчуючи елементами керування Windows Forms. У звичайному застосунку мовою Visual C# бібліотека класів .NET Framework інтенсивно використовується для "устрою" коду.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи статистичного аналізу та фільтрації даних зі змінних носіїв.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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 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 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 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.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Дана робота може забезпечувати віртуальне середовище розповсюдження ("VDE"), в якому електронна керуюча інформація управління правами може бути доставлена через незахищені (наприклад, аналогові) канали зв'язку. Це віртуальне середовище розповсюдження є дуже гнучким і зручним, враховуючи існуючі та нові бізнес-моделі, а також забезпечуючи безпрецедентний ступінь гнучкості в сприянні спеціальному створенню нових домовленостей і відносин між електронною комерцією та учасниками ланцюга створення вартості – незалежно від того, чи вміст розповсюджується в цифрові та/або аналогові формати.

Дана робота додатково забезпечує наступні важливі та переваги:

– Незмивна та невидима безпечна техніка для надання інформації про керування правами.

– Незмивний метод зв’язування елементів електронної комерції та/або керування правами з аналоговим вмістом, таким як фільми, відео та звукозаписи.

– Постійний зв’язок елементів керування торгівлею та/або керуванням правами з вмістом від одного кінця системи розповсюдження до іншого – незалежно від кількості та типів перетворень між форматами сигналізації (наприклад, з аналогового на цифровий і з цифрового на аналоговий).

– Можливість вказати правила управління правами «без копіювання/одна копія/багато копій», а також більш складні моделі прав і ціноутворення транзакцій (такі як, наприклад, «оплата за перегляд» та інші).

– Можливість повної та бездоганної інтеграції з комплексними загальними електронними рішеннями для керування правами (такими як ті, що розкриті в патентній специфікації Гінтера та інших, згаданій вище).

– Безпечна доставка керуючої інформації в поєднанні з авторизованими аналоговими та іншими нецифровими та/або незахищеними механізмами доставки інформаційних сигналів.

– Можливість забезпечувати більш складні та/або більш гнучкі правила комерції та/або керування правами під час переходу вмісту з аналогової до цифрової сфери й назад.

– Гнучка здатність передавати правила торгівлі та/або керування правами, які впроваджують нові, оновлені чи додаткові бізнес-моделі, авторизованим аналоговим та/або цифровим пристроям.

Коротко кажучи, у цих роботах використовується «стеганографія» для фактично незмивного та практично непомітного кодування керування правами та/або правил електронної комерції та контролю в інформаційному сигналі, такому як, наприклад, аналоговий сигнал або оцифрована (наприклад, дискретизована) версія аналоговий сигнал.

Грецький термін «стеганографія» стосується різних методів секретного спілкування «прихованого письма», які дозволяють надійно передавати важливі повідомлення незахищеними каналами зв’язку. Ось кілька прикладів стеганографії:

У стародавній Персії важливе повідомлення колись було витатуйовано на поголеній шкірі голови довіреного посланця. Потім посланець дозволив своєму волоссю відростити, повністю приховавши повідомлення. Коли посланець дістався місця призначення, він знову збрив волосся, відкривши секретне повідомлення, щоб одержувач міг прочитати його на поголеній шкірі посланця. Див. Kahn, David, The Codebreakers, сторінка 81 і далі. і сторінка 513 і далі. (Макміллан 1967). Ця незвичайна техніка приховування повідомлення є однією з ілюстрацій «стеганографії».

Інша «стеганографічна» техніка кодує секретне повідомлення в іншому, звичайному повідомленні. Наприклад, повідомлення «Hey Elmer, Lisa Parked My Edsel» кодує секретне повідомлення «HELP ME» – перша літера кожного слова

повідомлення, що утворює літери секретного повідомлення («Hey Elmer, Lisa Parked My Edsel»). Варіанти цієї методики можуть забезпечити додатковий захист, але основна концепція та сама – знайти спосіб приховати секретне повідомлення в інформації, яка може або буде надіслана незахищеним каналом.

Невидимі чорнила – це ще одна широко використовувана техніка «стеганографії». Секретне повідомлення пишеться за допомогою спеціального зникаючого або невидимого чорнила. Повідомлення можна написати на чистому аркуші паперу або, як правило, на зворотному чи лицьовому боці аркуша паперу, на якому міститься звичайний або законний лист або інше письмове повідомлення. Одержувач виконує спеціальний процес над отриманим документом (наприклад, піддає його хімічному або іншому процесу, який робить невидимі чорнила видимими), щоб він чи вона могли прочитати повідомлення. Будь-хто, хто перехопить папір, не зможе виявити таємне повідомлення – або навіть знати, що воно там – якщо перехоплювач не знає, що потрібно шукати невидиме повідомлення, а також не знає, як поводитися з папером, щоб зробити невидиме чорнило видимим

У цих роботах використовується стеганографія, щоб гарантувати, що закодована керуюча інформація є як практично невидимою, так і практично незмивною, коли вона проходить по незахищеному каналу зв'язку. На приймальній стороні захищений надійний компонент (наприклад, захищене середовище обробки, описане в Ginter та ін.) відновлює стеганографічно закодовану керуючу інформацію та використовує відновлену інформацію для керування електронними правами (наприклад, на аналоговому або іншому інформаційні сигнали, що передаються по одному каналу).

Один конкретний аспект, наданий цією роботою, включає стеганографічне кодування інформації керування цифровими правами в інформаційний сигнал, такий як, наприклад, аналоговий або оцифрований телевізійний, відео- або радіосигнал. Процес стеганографічного кодування по суті нерозривно переплітає цифрову керуючу інформацію з зображеннями, звуками та/або іншим вмістом,

який несе інформаційний сигнал, але переважно без помітного погіршення або іншого впливу на ці зображення, звуки та/або інший вміст. Може бути важко виявити (навіть за допомогою освічених методів обробки сигналів), що аналоговий сигнал був стеганографічно закодований за допомогою контрольного сигналу керування правами, і може бути важко усунути стеганографічно закодований керуючий сигнал без руйнування або погіршення іншої інформації чи вмісту сигнал несе.

Дана робота також забезпечують безпечне, надійне захищене середовище обробки для відновлення стеганографічно закодованого керуючого сигналу з інформаційного сигналу та для забезпечення процесів керування правами на основі відновленого стеганографічно закодованого керуючого сигналу. Це дозволяє повністю інтегрувати (і зробити сумісним) механізм доставки інформаційного сигналу з цифровим віртуальним середовищем розповсюдження та/або іншою електронною системою керування правами.

Відповідно до ще одного аспекту, наданого цією роботою, стеганографічно закодована керуюча інформація управління цифровими правами може використовуватися разом із скремблованим і/або зашифрованим інформаційним сигналом. Скремблювання та/або шифрування можна використовувати для забезпечення управління правами, наданого відповідно до стеганографічно закодованої інформації керування керуванням правами. Наприклад, керуючий сигнал може бути стеганографічно декодований і використаний для контролю, принаймні частково, за яких обставин і/або як інформаційний сигнал повинен бути дешифрований і/або дешифрований.

Відповідно до ще однієї ознаки, наданої винаходом, цифрові сертифікати можуть використовуватися для безпечного забезпечення дотримання стеганографічно закодованої інформації керування правами.

Відповідно до ще однієї ознаки, наданої винаходом, стеганографія використовується для кодування інформаційного сигналу з інформацією управління правами у формі однієї або більше захищених організаційних

структур, пов'язаних з електронними засобами керування. Електронні засоби керування можуть, наприклад, визначати дозволені та/або необхідні операції з вмістом, а також наслідки виконання та/або невиконання таких операцій. Організаційна(-і) структура(-и) може(-ють) ідентифікувати, неявно чи явно, вміст, до якого застосовуються електронні елементи керування. Організаційна(і) структура(и) також може визначати обсяг контенту та семантику контенту.

Тип, обсяг і характеристики стеганографічно закодованої інформації керування правами є гнучкими та програмованими, забезпечуючи багатий, різноманітний механізм для розміщення широкого спектру схем керування правами. Інформацію про керування можна використовувати для безпечного застосування простих наслідків безпечного керування правами, наприклад елементів керування типу «копіювання/без копіювання/одна копія», але жодним чином не обмежується такими моделями. Навпаки, даний винахід може бути використаний для того, щоб увімкнути та застосувати набагато багатші, складніші моделі керування правами, включаючи, наприклад, такі, що включають аудит використання, автоматичні електронні платежі та використання додаткових електронних мережевих з’єднань. Крім того, механізми контролю керування правами, надані цією роботою, можна нескінченно розширювати та масштабувати - повністю адаптувати майбутні моделі, коли вони комерційно розгортаються, зберігаючи при цьому повну сумісність з різними (і, можливо, більш обмеженими) моделями керування правами, розгорнутими на попередніх етапах.

Організаційна(і) структура(и) може бути стеганографічно закодована таким чином, щоб вона була захищена з метою забезпечення секретності та/або цілісності. Застосовувані стеганографічні методи можуть забезпечувати певний ступінь захисту секретності або інші методи безпеки (наприклад, цифрове шифрування, цифрові печатки тощо) можуть бути використані для забезпечення бажаного або необхідного рівня безпеки та/або захисту цілісності стеганографічно закодованого інформації.

В одному прикладі організаційна(і) структура(и) може включати цифрові електронні контейнери, які безпечно містять відповідну цифрову електронну керуючу інформацію. Такі контейнери можуть, наприклад, використовувати криптографічні методи. В інших прикладах організаційна(і) структура(и) може визначати асоціації з іншою електронною керуючою інформацією. Інша електронна керуюча інформація може надаватися незалежно через той самий або інший шлях зв’язку, який використовується для доставки організаційної структури(-й).

В одному прикладі використовувані стеганографічні методи можуть передбачати застосування інформації про організаційну структуру у формі високочастотного "шуму" до аналогового інформаційного сигналу. Спектральні перетворення можуть бути використані для застосування та відновлення такого стеганографічно закодованого високочастотного "шуму". Оскільки високочастотні шумові компоненти інформаційного сигналу можуть бути по суті випадковими, додавання псевдовипадкового стеганографічно закодованого компонента керуючого сигналу може практично не призвести до помітного погіршення інформаційного сигналу, і його може бути важко видалити після введення (принаймні без додаткових знань про як сигнал був включений, який може містити спільний секрет).

Відповідно до іншого аспекту, передбаченого винаходом, процес стеганографічного кодування аналізує інформаційний сигнал, щоб визначити, яка надлишкова смуга пропускання доступна для стеганографічного кодування. Процес стеганографічного кодування може використовувати кодування зі змінною швидкістю передачі даних, щоб застосувати більше керуючої інформації до частин інформаційного сигналу, які використовують набагато менше, ніж уся доступна смуга пропускання каналу зв’язку, і застосувати менше керуючої інформації до частин інформаційного сигналу, які використовують майже всі доступної пропускної здатності каналу зв’язку.

Відповідно до ще одного аспекту, наданого винаходом, численні організаційні структури можуть бути стеганографічно закодовані в межах даного інформаційного сигналу. Кілька організаційних структур можуть застосовуватися до різних відповідних частин інформаційного сигналу, і/або численні організаційні структури можуть бути повторами або копіями одна одної, щоб гарантувати, що електронний пристрій має «пізній вхід» і/або здатність виправляти помилки та/або може швидко знайти відповідну організаційну структуру (структури), починаючи з будь-якої довільної частини потоку інформаційного сигналу.

Відповідно до ще одного аспекту, передбаченого цією роботою, організаційна структура може бути стеганографічно закодована в конкретній частині інформаційного сигналу, що несе вміст, до якого застосовується організаційна структура, таким чином встановлюючи неявну відповідність між організаційною структурою та ідентифікацією та /або обсяг і/або семантика інформаційного вмісту, до якого застосовується організаційна структура. Кореспонденція може, наприклад, включати явні компоненти (наприклад, внутрішньо визначені початкові/кінцеві точки), зі сховищем або іншим фізичним зв’язком, визначеним для зручності (тобто може мати сенс розмістити організаційну структуру поблизу місця її використання, щоб не шукати носій інформації, щоб знайти його).

Відповідно до ще одного аспекту, забезпеченого цією роботою, покажчики можуть бути стеганографічно закодовані в частини потоку інформаційного сигналу, який має невелику надлишкову доступну смугу пропускання. Такі вказівники можуть використовуватися, наприклад, для спрямування електронного пристрою на частини потоку інформаційного сигналу, які мають більш доступну смугу пропускання для стеганографічного кодування. Такі покажчики можуть забезпечувати покращений час доступу до стеганографічного декодування - особливо, наприклад, у програмах, у яких потік

інформаційних сигналів зберігається або іншим чином доступний на основі довільного доступу.

Звичайно, методи статистичного аналізу та фільтрації даних зі змінних носіїв з використанням стеганографічних методів й методів вбудовування водяних знаків мають певні розходження, і кожне із цих напрямків має свій шлях розвитку. Однак розглянуті нами атаки дозволяють перевірити стійкість як стегосистем, так і систем, заснованих на використанні цифрових водяних знаків.

Типи атакуючих

У ряді випадків модель стегосистеми формулюють у вигляді так званої «проблеми ув'язнених»: ув’язнені Аліса й Боб обмінюються секретними повідомленнями, однак це відбувається на очах (і в деяких випадках при активній участі) охоронця Віллі. У цьому випадку охоронець Віллі символізує реального зловмисника, що загрожує нормальному функціонуванню стегосистеми.

Можливі різні моделі його поводження: по-перше, Віллі може тільки стежити за повідомленнями, по-друге, він має можливість впливати на канал передачі повідомлень між ув'язненими, що ускладнює рішення проблеми. По-третє, в охоронця Віллі є можливість змінювати самі повідомлення Аліси й Боба.

Таким чином, ми маємо три типи атакуючих:

– Пасивний зловмисник, що тільки шпигує за повідомленнями, але не може в них нічого змінити.

– Активний зловмисник може модифікувати дані, які відправляються ув'язненими. Наприклад, він може модифікувати текст, не видозмінюючи його семантичний зміст. Реальний приклад активного охоронця – це цензура телеграм урядом Сполучених Штатів під час Другої світової війни: семантичний зміст телеграм не міг бути змінений, але цензори злегка видозмінювали точну редакцію тексту, заміняли слова на синоніми, що руйнувало можливі таємні послання.

– Злочинний зловмисник може не тільки видозмінювати повідомлення ув'язнених, але й становити нове з декількох або переписати послання цілком. У цьому випадку ув'язненим можна сподіватися на що завгодно, тільки не на

знаходження волі! На щастя, реальні ситуації, коли починає діяти злісний охоронець, досить рідкі, тому випадок злісного охоронця найчастіше не розглядається.

Погрози

Як уже було відзначено, до основних погроз безпеки стегосистем ставляться: виявлення стеганографічного каналу, добування, руйнування й підміна схованого повідомлення. Помітимо, що в якості виродженого случаю існує ще одна погроза, а саме – заборона на яку б те не було передачу послань.

Виявлення стеганографічного каналу є погрозою «нижнього рівня». Вона може бути здійснена будь-яким типом зловмисника. Якщо Віллі здатний виявити стеганографічний канал, то говорять, що стегосистема є нестійкою. Захист від цієї погрози вважається основним завданням стеганографії.

Добування схованого повідомлення полягає в тому, що зловмисник не тільки визначає існування стеганографічного каналу, але й перехвачує сховане повідомлення. Ця погроза може бути здійснена активним або злочинним зловмисником.

Руйнування схованого повідомлення має на увазі внесення в стего таких припустимих (не вимоги, що порушує, природності контейнера) змін, які не дозволять одержувачеві витягти убудоване повідомлення. Така погроза може бути здійснена як активним, так і злочинним зловмисником.

Підміна схованого повідомлення, будучи найбільш сильною погрозою стеганографічній системі, може бути здійснена тільки злочинним зловмисником. Суть її складається в добуванні наявного в стего схованого повідомлення й розміщенні замість нього іншого (помилкового) повідомлення.

Найбільш складної в реалізації є погроза виявлення стеганографічного каналу. Далі, у порядку зменшення складності випливають:

– підміна схованого повідомлення (для здійснення якого в загальному випадку необхідно знати як алгоритм приховання, так і секретний ключ);

– добування схованого повідомлення (для чого потрібно знати алгоритм приховання);

– руйнування схованого повідомлення (яке можливо зробити не знаючи ні алгоритм приховання, ні тим більше стегоключ).

На перший погляд здається дивним те, що найбільш складну з погляду реалізації погрозу (виявлення стеганографічного каналу) може здійснити навіть найбільш слабкий (пасивний) зловмисник. Насправді тут немає ніякого протиріччя, тому що це можливо лише в тому випадку, коли відправник порушує вимога природності контейнера, що й викликає підозру зловмисника. У противному випадку, завдання виявлення стеганографічного каналу є нетривіальною.

Атаки

Головною метою будь-якої атаки на стегосистему є, насамперед, – виявлення стеганографічного каналу. Максимально досяжним результатом при реалізації атаки – одержання повної інформації про стеганографічну систему.

Помітимо, що в наявній літературі, як правило, лише перераховуються можливі типи атак, без яких би те не було коментарів [4, 5, 6, 7].

Атака з відомим контейнером

Це сама слабка із всіх можливих атак. Маючи в розпорядженні вихідний контейнер, Віллі після його порівняння з контейнером, посланим Алісою, може зробити вивід про існування стегоканала. Подібну атаку може зробити пасивний зловмисник.

Трохи складніший наступний різновид цієї атаки – Аліса й Боб використовують канал з повторенням, при цьому в кожний контейнер, крім схованого повідомлення, можуть бути внесені деякі випадкові зміни. Тоді завдання Віллі складається у визначенні того, що втримується в контейнері – випадкова шумова послідовність або сховане повідомлення.

Атака з вибором контейнера

У цьому випадку Віллі сам вибирає тип контейнера (найбільш зручний для нього з погляду наступного аналізу) і створює умови, при яких Аліса й Боб можуть скористатися для передачі схованого повідомлення тільки цим контейнером.

Як і в попередньому типі атаки, порівняння вихідного й отриманого від Аліси контейнерів може допомогти Віллі зробити вивід про наявність або відсутність стеганографічного каналу. Атаку може здійснити пасивний зловмисник.

Атака з відомим стего

Це більше сильна атака, оскільки, розташовуючи стего, але не знаючи вихідного контейнера, Віллі не може однозначно встановити факт існування стегоканала. Його завдання в цьому випадку зводяться до аналізу переданого Алісою контейнера й визначенню, чи є він стего або порожнім контейнером. Подібну атаку може здійснити активний або зловмисний зловмисник.

Атака з вибором стего

Така атака припускає, що Віллі відомо деяка безліч стего й, можливо, реакція Боба на деякі них їх. Тоді завдання Віллі складається в «нав'язуванні» йому за певних умов якогось конкретного стего й аналізі отриманої відповіді.

Подібна атака може бути здійснена тільки при наявності каналу з повторенням. Її може реалізувати активний або злочинний зловмисник.

Атака з відомим схованим повідомленням

Можливі принаймні два варіанти розглянутої атаки:

– відомий стего, що відповідає схованому повідомленню. У цьому випадку завдання Віллі складається у визначенні секретного ключа (атака може бути здійснена активним або зловмисним зловмисником);

– відповідному схованому повідомленню стего невідомий. У цьому випадку завдання є нетривіальною й, загалом кажучи, може не мати рішення.

Атака з вибором схованого повідомлення

Це найдужчий тип атаки, що може зробити тільки зловмисний зловмисник. Сценарій атаки зводиться до наступного: Віллі «підкидає» відоме йому сховане повідомлення Алісі й, одержавши стего із цим повідомленням, призначене Бобові, аналізує його з метою встановити секретний ключ.

Крім того, існує ряд атак, спрямованих на руйнування убудованого повідомлення. В основному даний тип атак застосовується для руйнування або видалення цифрових водяних знаків, рідше для руйнування убудованого повідомлення (хоча далі по тексту використовується термін «убудоване повідомлення»).

Стиск із втратою даних

У цей час для зменшення розміру файлів повсюдно використовують стиск із втратою даних. Для цифрових зображень найбільш популярне використання формату JPEG. Однак для багатьох методів вбудовування повідомлення перетворення заповненого контейнера може бути фатальним: убудоване повідомлення або буде ушкоджено, або просто загублене.

Геометричні перетворення

Стійкість до геометричних перекручувань є неодмінною вимогою, що пред'являється до стегосистем. Однак не всі існуючі на даний момент системи витримують такого роду атаки.

Можливе використання наступних геометричних перетворень:

1. Зменшення розміру контейнера шляхом відрізання граничних областей. Таке перетворення може частково зруйнувати убудоване повідомлення й привести до втрати деякої частини інформації.

2. Поворот . Іноді поворот контейнера-зображення навіть на незначний кут може привести до часткового або повного руйнування убудованого повідомлення.

3. Масштабування . У цьому випадку використовуються різні коефіцієнти масштабування по горизонталі й вертикалі, що також може привести до часткового руйнування убудованого повідомлення.

Атаки, спрямовані на видалення убудованого повідомлення

Зашумлення контейнера. Внесення в заповнений контейнер додаткового шуму неминуче спричиняє втрату повідомлення.

З іншого боку, операція фільтрації також приводить до знищення убудованого повідомлення. У цьому випадку використовують або низькочастотний і високочастотний фільтри окремо, або обидва фільтри спільно.

Зміна гістограмми – розтягання або вирівнювання гістограмми, які іноді використовуються для компенсації недостатнього висвітлення.

Комбіновані атаки

Всі перераховані вище атаки, спрямовані на руйнування або видалення убудованого повідомлення, можуть комбінуватися один з одним. Наприклад, можна вирівняти гістограмму, а потім перетворити контейнер у формат JPEG або небагато обрізати контейнер, а потім провести операцію фільтрації. Однак не слід забувати, що багаторазовий вплив може привести до виникнення помітних перекручувань контейнера. Поява таких перекручувань може насторожити одержувача стего, і учасники можуть домовитися про зміну стеганографічного каналу.

У кожному разі, перш ніж використовувати той або інший тип атаки, спрямований на руйнування убудованого повідомлення, необхідно виявити стегоканал. Однак застосовувати геометричні атаки до будь-якого файлу, що може використовуватися як контейнер, безглуздо. По-перше, багато які стегосистеми стійкі до перетворень і трансформацій контейнера, а по-друге, може не вистачити обчислювальних і людських ресурсів. Отут як приклад можна привести ситуацію, у якій виявилися агенти ФБР, коли спробували без відповідних ресурсів контролювати не тільки телефонні переговори, але й Інтернет-трафік.

На закінчення необхідно відзначити, що наведені в роботі типи атак використовуються не тільки для виявлення, руйнування або добування убудованих повідомлень, але й для аналізу стійкості стеганографічних методів.

Запропонована класифікація атак не є остаточною й незмінною. Удосконалювання стеганографічних методів приведе до появи нових методів стегоаналізу.

3.2 Розробка структурної схеми

Структурна схема розробленої системи зображена на рисунку 3.1. З метою вдосконалювання теоретичної бази проведений загальний аналіз стеганографічних методів і алгоритмів, використовуваних на справжній момент у програмних засобах схованої передачі електронних документів, з використанням змінних носіїв. Досліджено специфічні уразливості програмних засобів і систем схованої передачі електронних документів. Уведено загальну класифікацію атак на стеганографічні системи зв'язку залежно від використовуваних для проведення атак уразливостей. Крім того велика увага приділена також і сучасним моделям стеганографічних систем зв'язку, представленим у різних публікаціях і узагальненим за результатами аналізу існуючого програмного забезпечення.

У результаті проведеного аналізу відомих методів, моделей систем і програмних засобів виявлені наступні загальні недоліки існуючих рішень в області стеганографічного методу збереження конфіденційності інформації:

– відсутність загальних єдиних підходів і проробленних базових рішень до побудови стеганографічних систем;

– низька стійкість до різних методів стеганоаналізу;

– низька стійкість до руйнуючих впливів;

– низька перешкодозахищеність;

– сильна залежність ступеня скритності від особливостей контейнера;

– відсутність методів оцінки рівня надійності й докази стійкості до атак пасивного зловмисника;

– слабість застосовуваних алгоритмів перетворення повідомлень;

– складність перебудови стеганографічних алгоритмів залежно від використовуваного ключа приховання;

– складність побудови надійних систем із симетричними й відкритими ключами;

– загальна надійність систем сильно залежить від обсягів приховуваної інформації.

Комерційне використання програмних засобів ЗІ крім іншого накладає додаткові обмеження, що стосуються в першу чергу забезпечення можливості широкого поширення програмних продуктів.

Таким чином, для стеганографічних методів методу збереження конфіденційності інформації в системах електронного документообігу з використанням змінних носіїв 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. За результатами проведених досліджень пропонується сформулювати вимоги до криптографічного й стеганографічного алгоритмів, розробити методи й алгоритми їхнього узгодження, проробити відповідну теоретичну базу. Крім того, з огляду на малу пропрацьованість питань, що стосуються стеганографічних методів методу збереження конфіденційності інформації й ефективної протидії методам стеганоаналізу, пропонується приділити їм першорядне значення.

Проведений аналіз показав, що всі відомі методи стеганоаналізу мають певні границі застосовності, чутливості й вірогідності результатів. На справжній момент існує реальна можливість побудови стеганографічних методів, які мали би абсолютну стійкість до відомих методів аналізу, тобто були б не виявляємі.

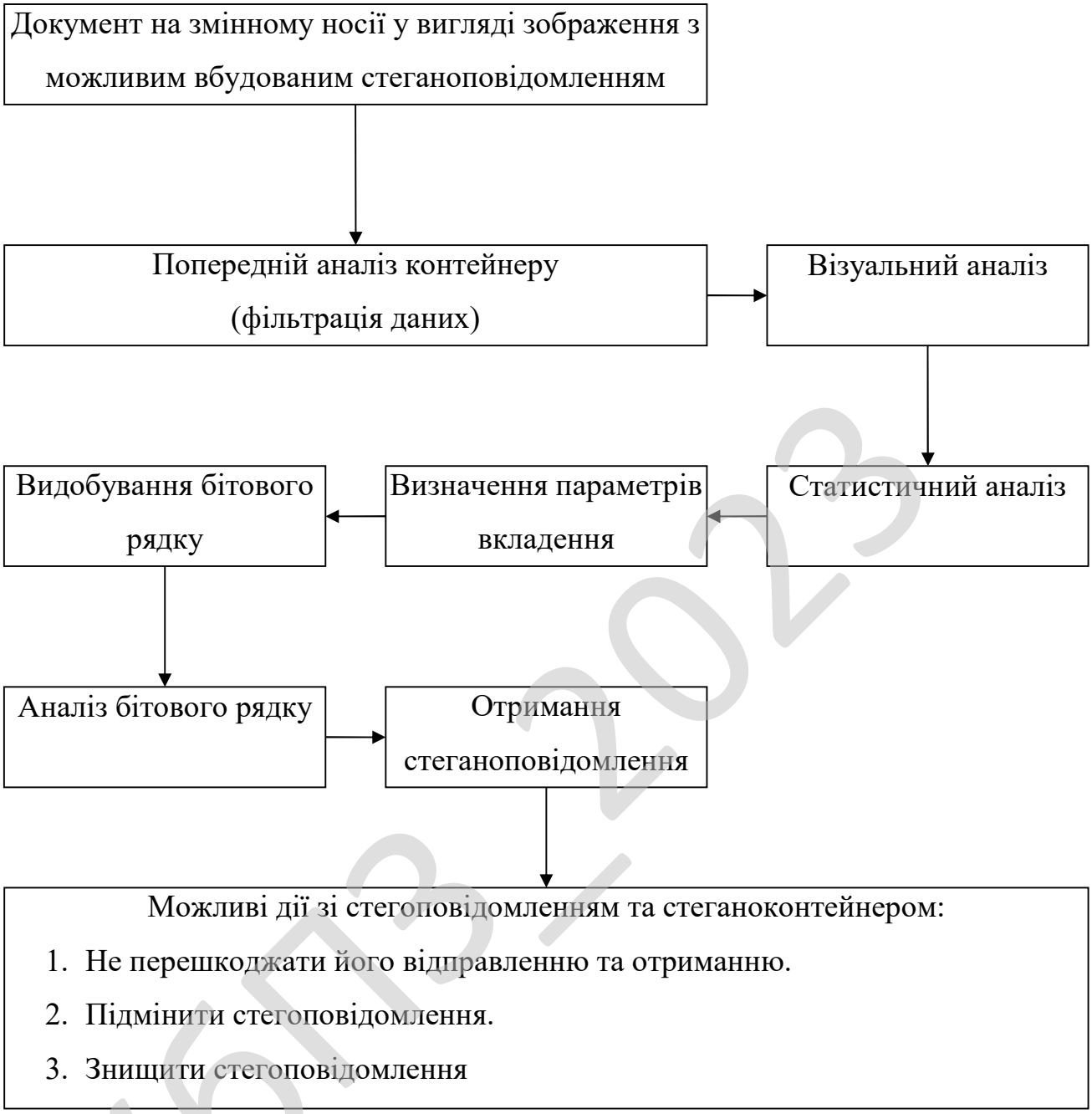

Рисунок 3.1 – Структурна схема системи

Цікавим фактом є те, що з урахуванням деяких обмежень ряд існуючих стеганографічних методів є, що виявляються не при вбудовуванні зверхмалих обсягів інформації. Даний факт обумовлений тим, що навіть найсучасніші методи аналізу мають певні пороги чутливості. Так, якщо після вбудовування повідомлення аналізовані показники контейнера перебувають у межах

припустимих погрішностей, він буде прийнятий за порожній. Крім зазначених висновків за результатами аналізу в першій частині були також сформульовані пропозиції по протидії сучасним методам стеганоаналізу.

Дані пропозиції рекомендується враховувати при побудові нових стеганографічних методів.

Криптостеганографічною системою будемо називати систему схованої передачі інформації на відкритих каналах зв'язку, засновану на спільному застосуванні криптографічних алгоритмів, стеганографічних методів, а також алгоритмів узгодження вхідних і вихідних даних зазначених алгоритмів і методів.

Криптографічна частина (E, D) забезпечує криптографічне закриття (попереднє шифрування) переданих повідомлень. Відповідає за перетворення переданих повідомлень до псевдовипадкового виду з рівномірним розподілом.

Стеганографічна частина (S) здійснює безпосереднє приховання й добування переданих даних, що пройшли процедуру попереднього шифрування, у контейнерах з C.

Алгоритми узгодження (MT) забезпечують узгодження криптографічної й стеганографічної частин системи по вхідним і вихідним даним. Відповідають за пряме приведення й зворотне перетворення отриманих з виходу криптографічної частини даних до двійкових послідовностей, аналогічним по своїх статистичних властивостях двійковим послідовностям, що витягається з порожніх контейнерів.

Модель криптостеганографічної системи представлена на рисунку 3.1. Для даної моделі криптостеганографічної системи зв'язку розглянуті можливості зловмисника по виявленню схованого каналу й добуванню схованих повідомлень. Показано, що при виконанні ряду вимог до компонентів системи успішна атака на системи даного виду можлива лише у випадку успішної атаки на криптографічні алгоритми. Тобто стійкість системи до атак пасивного зловмисника визначається стійкістю до злому криптографічної частини.

3.3 Розробка функціональної схеми

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

1. Інтерфейс користувача системи статистичного аналізу та фільтрації даних зі змінних носіїв, з метою виявлення несанкціонованого витоку конфіденційної інформації методом стегоаналізу.

2. Візуальні методи стегоаналізу.

3. 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.

4. Методи стегоаналізу заснованих на стисненні даних.

5. Методи стегоаналізу заснованих на використанні нейронних мереж.

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.

Рисунок 3.2 – Функціональна схема системи

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.

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.

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 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

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 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.

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

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):

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

$$F_2: 255 \leftrightarrow 0, 1 \leftrightarrow 2, 3 \leftrightarrow 4, \dots, 253 \leftrightarrow 254, 255 \leftrightarrow 0.$$

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.

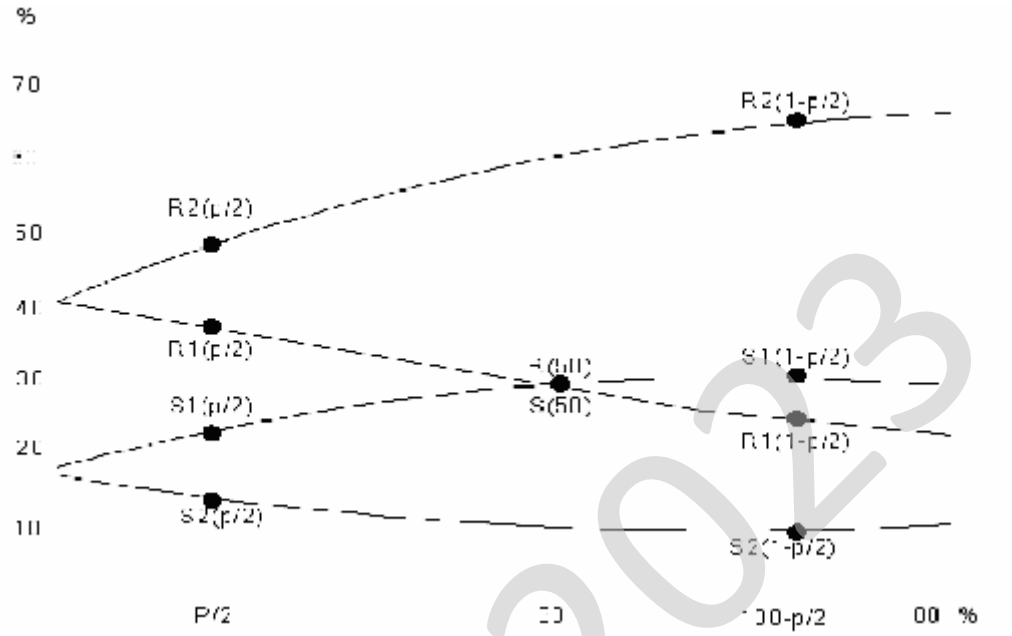

Рисунок 3.3 – RS-діаграма типового зображення

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. На рис. 3 наведена діаграма для типового зображення. На осі абсцис розташована кількість інвертованих біт x, шукана довжина повідомлення p, на осі ординат – відносні значення регулярних і сингулярних груп по відношенню до спільного числа груп зображення.

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 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.

Таблиця 3.1 – Типові відсотки значень розмірів хибно виявлених повідомлень RS-аналізом

Тип зображення	Хибно визначена довжина повідомлення у відсотках від об'єму контейнера
Фотографії, скановані зображення	= 1 %
Зображення з Інтернету, підготовлені до друку	= 15 ... 20 %
Дрібнотекстуровані зображення, високодеталізовані зображення	= 30 ... 100 %

У табл. 3.1 наведено типові значення розмірів виявлених повідомлень у пустих контейнерах для зображень різних типів [5].

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:

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

Нехай $\varphi(X)$ – псевдовипадкова зміна молодших біт всіх байтів послідовності X, що подається на вхід програми, а $Y = \varphi(X)$ – 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:

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

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 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:

$$M_n = \sum_{k=-N/2}^{N/2} |f_k|^n p(f_k), \qquad (5)$$

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).

$$p(f_k) = \frac{|H(f_k)|}{\sum_{k=-N/2}^{N/2} |H(f_k)|}, \quad (6)$$

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

$$H(f) = \int_{-\infty}^{\infty} h(x) e^{-j2\pi fx} dx, \quad (7)$$

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:

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

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:

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

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:

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

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].

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Процеси взаємодіють наступним чином. Першим процесом, який завантажується у системі, є процес виведення головного вікна програми. Він взаємодіє з наступними процесами:

– Процес вибору параметрів системи.

– Процес вибору диску для аналізу.

Процес вибору параметрів системи взаємодіє з наступними процесами:

– Процес вибору параметрів статистичного аналізу.

– Процес вибору параметрів фільтрації даних.

Рисунок 3.3 – Діаграма взаємодії процесів

Процес вибору диску для аналізу взаємодіє з процесом статистичного аналізу файлів, який, у свою чергу, взаємодіє з наступними процесами:

– Процес RS-аналізу.

– Процес аналізу на основі критерію χ^2.

– Процес аналізу на основі стиснення даних.

– Процес аналізу на основі використання нейронних мереж.

– Процес виведення результатів статистичного аналізу.

Процес виведення результатів статистичного аналізу взаємодіє з процесом виведення списку файлів за вкладеннями.

Процес виведення списку файлів за вкладеннями взаємодіє з процесом фільтрації даних.

Процес фільтрації даних взаємодіє з наступними процесами:

– Процес виведення результатів фільтрації даних.

– Процес видалення файлів, які не вдалося очистити.

– Процес очищення файлів від вкладень.

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.

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

4.1 Блок–схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи основної програми. Його блок-схема зображена на рисунку 4.1.

Робота основної програми складається з виконання наступної послідовності кроків:

Крок 1. Виведення основного вікна програми на екран.

Крок 2. Вибір диску для статистичного аналізу та фільтрації даних.

Крок 3. Виведення списку файлів та папок на диску.

Крок 4. Якщо обрана дія «провести статистичний аналіз», то перехід на наступний крок, інакше перехід на крок 6.

Крок 5. Виклик підпрограми «статистичний аналіз файлів на диску».

Крок 6. Якщо обрана дія «провести фільтрацію даних», то перехід на наступний крок, інакше перехід на крок 9.

Крок 7. Виклик підпрограми «Фільтрація даних на диску».

Крок 8. Виведення результатів фільтрації даних.

Крок 9. Якщо обрана дія «зміна параметрів аналізу», то перехід на наступний крок, інакше перехід на крок 12.

Крок 10. Вибір типів файлів зі списку, які слід перевіряти.

Крок 11. Вибір алгоритмів статистичного аналізу зі списку, які слід застосовувати.

Крок 12. Якщо обрана дія «зміна параметрів фільтрації», то перехід на наступний крок, інакше перехід на крок 14.

Крок 13. Вибір дій зі списку, які слід виконувати з файлами, що містять

вкладення.

Крок 14. Якщо обрана дія «вихід», то здійснюється вихід з програми, інакше перехід на крок 2.

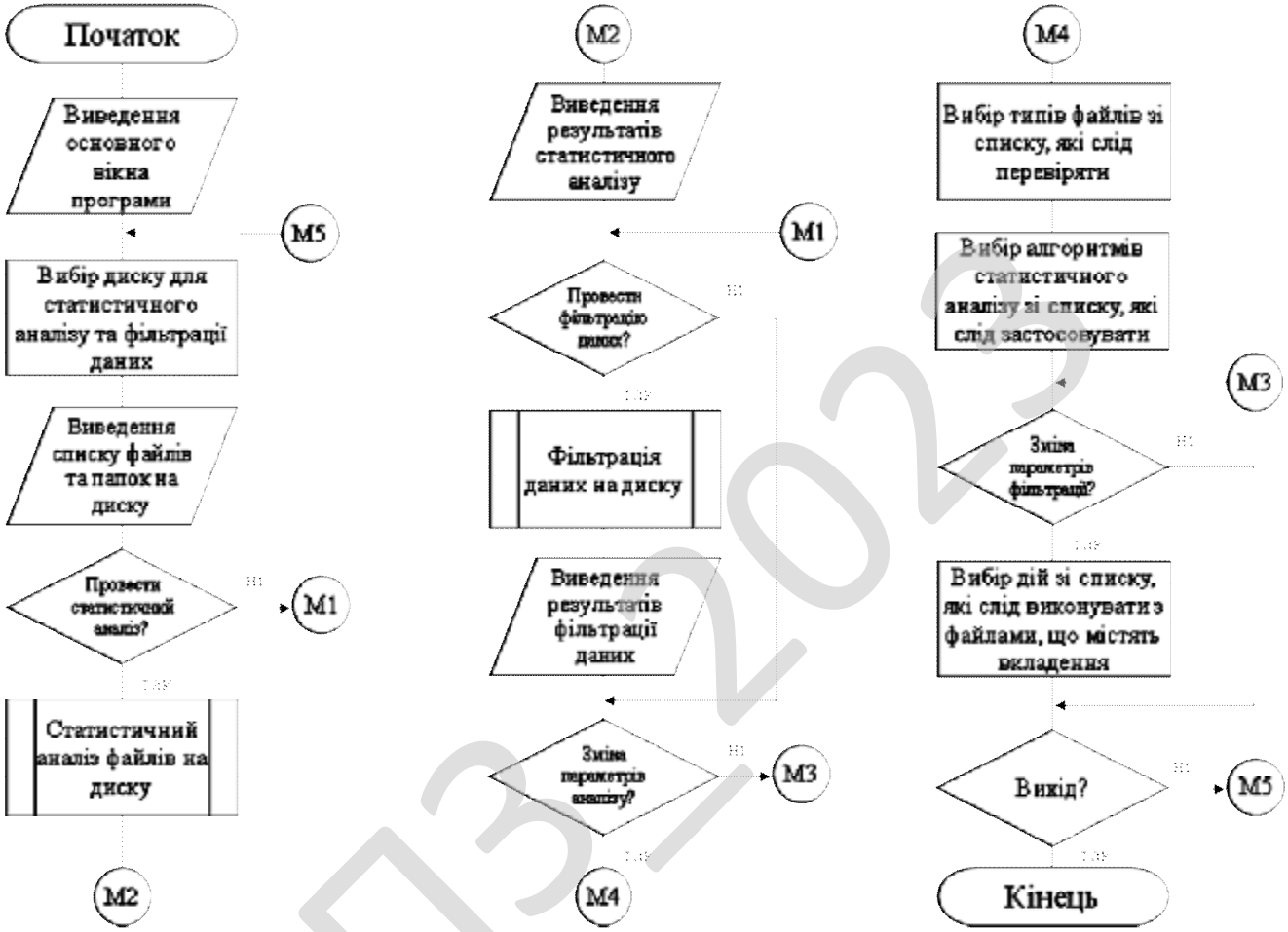

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображена блок-схема роботи підпрограми статистичного аналізу файлів на змінних носіях.

Дана підпрограма складається з наступної послідовності кроків:

Крок 1. Читання списку файлів на диску.

Крок 2. Читання параметрів статистичного аналізу

Крок 3. Від 1 до К, де К – кількість файлів на диску, повторюємо послідовність кроків 4-11.

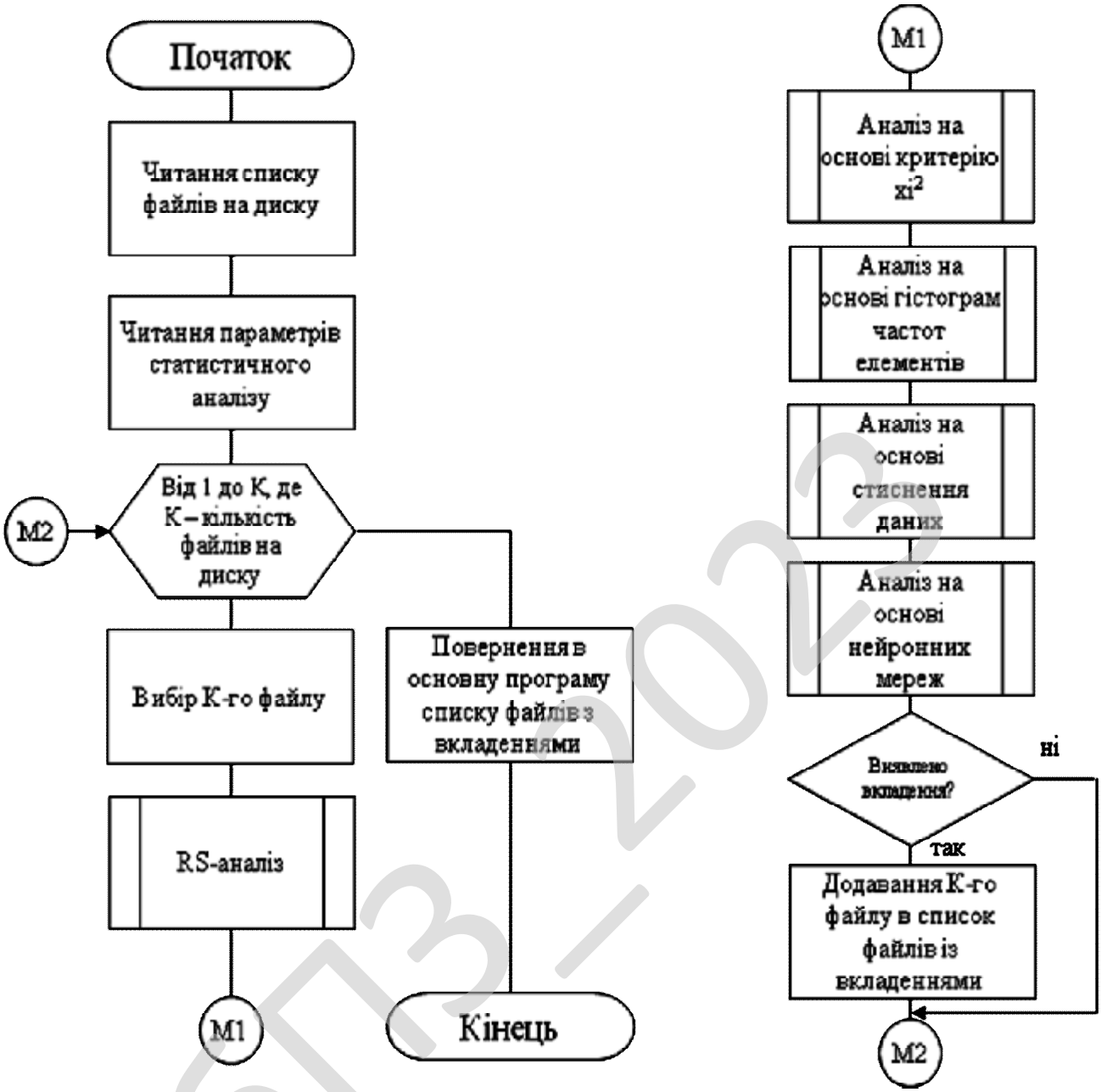

Рисунок 4.2 – Блок-схема підпрограми статистичного аналізу файлів на змінних носіях

Крок 4. Вибір К-го файлу.

Крок 5. Виклик підпрограми «RS-аналіз».

Крок 6. Виклик підпрограми «Аналіз на основі критерію xi^2».

Крок 7. Виклик підпрограми «Аналіз на основі гістограм частот елементів».

Крок 8. Виклик підпрограми «Аналіз на основі стиснення даних».

Крок 9. Виклик підпрограми «Аналіз на основі нейронних мереж».

Крок 10. Якщо у К-тому файлі виявлено вкладення, то перехід на наступний крок, інакше збільшення лічільнику файлів К на одиницю та перехід до наступної ітерації циклу (крок 4).

Крок 11. Додавання К-го файлу в список файлів із вкладеннями. Збільшення лічільнику файлів К на одиницю та перехід до наступної ітерації циклу (крок 4).

Крок 12. Вихід з циклу та повернення в основну програму списку файлів з вкладеннями.

На рисунку 4.3 зображена блок-схема роботи підпрограми фільтрації даних на змінних носіях.

Дана підпрограма складається з наступної послідовності кроків:

Крок 1. Читання списку файлів із вкладеннями.

Крок 2. Читання параметрів фільтрації даних.

Крок 3. Від 1 до К, де К – кількість файлів із вкладеннями, повторюємо полідовність кроків 4-10.

Крок 4. Обчислення, чи можливо видалити вкладення без шкоди для К-го файлу.

Крок 5. Якщо вкладення можна видалити, то перехід на наступний крок, інакше перехід на крок 7.

Крок 6. Видалення вкладення з К-го файлу.

Крок 7. Виведення повідомлення з запитом видалення К-го файлу.

Крок 8. Якщо користувач погодився видалити файл, то перехід на наступний крок, інакше перехід на крок 7.

Крок 9. Видалення К-го файлу.

Крок 10. Збільшення лічільнику файлів К на одиницю та перехід до наступної ітерації циклу (крок 4).

Крок 11. Вихід з циклу та повернення в основну програму звіту по результатам фільтрації даних.

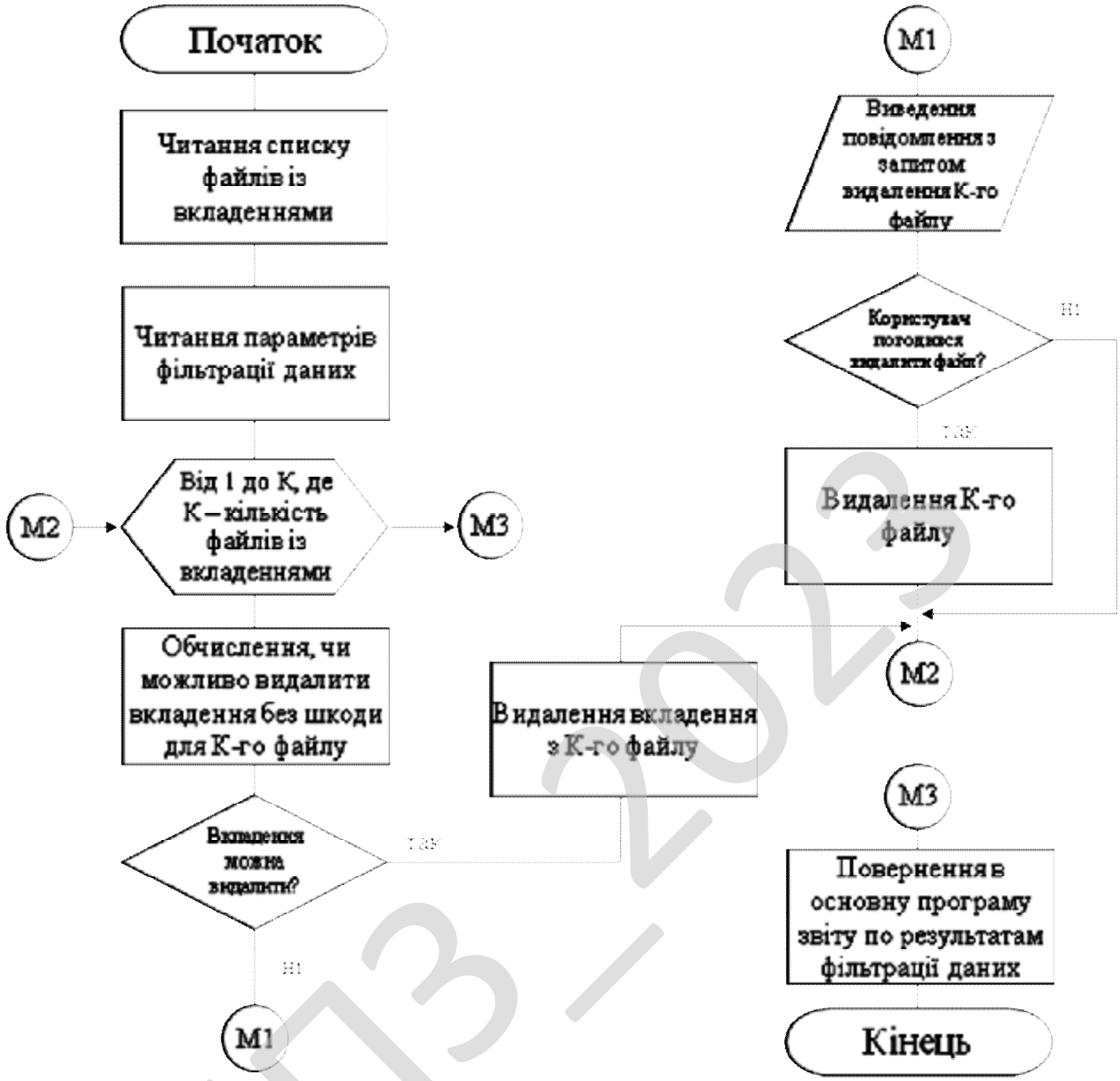

Рисунок 4.3 – Блок-схема підпрограми фільтрації даних на змінних носіях

Наведемо процедури статистичного RS-аналізу:

```
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++)
            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);
    }
    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];
        jac[0,1] = 0;
        jac[1,0] = 0;
        jac[1,1] = 1;
    }
```

Наведемо процедури статистичного аналізу з використанням критерію χ^2:

```
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;
    }
    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)
    {
        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];
    }
```

Наведемо процедури статистичного аналізу з використанням нейронної мережі:

```
private static void hladdoutputlayer(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, "HLAddOutputLayer: internal
error");
            neurooffs = hlnfieldwidth*neuroidx;
            connoffs = hlconnfieldwidth*connidx;
            if( !iscls )
            {

                //
                // регрес нейромережі
                //
```

```
                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;
```

```
                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;
                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;
                }
                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);
                hladdoutputlayer(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);
```

```
                hladdoutputlayer(network,   ref   idxconn,   ref   idxneuro,   ref
idxstruct, ref idxweights, 2, nhid1, nout, iscls, islinearout);
                return;
            }

            //
            // 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);
            hladdoutputlayer(network,  ref  idxconn,  ref  idxneuro,  ref  idxstruct,
ref idxweights, 3, nhid2, nout, iscls, islinearout);

}
```

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою наступного алгоритму захисту інформації DSA.

Відправник і одержувач електронного документа використовують при обчисленні великі цілі числа: G і P – прості числа, L біт кожне ($512 < L < 1024$); q – простої число довжиною 160 біт (дільник числа (P-1). Числа G, P, q є відкритими й можуть бути загальними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число X, $1 < X < q$. Число X є секретним ключем відправника для формування електронного цифрового підпису. Потім відправник обчислює значення $Y = G^X \bmod P$. Число Y є відкритим ключем для перевірки підпису відправника. Число Y передається всім одержувачам документів.

Цей алгоритм також передбачає використання однобічної функції гешування h(-). У стандарті DSS визначений алгоритм безпечного гешування SHA. Для того щоб підписати документ M, відправник гешує його в ціле геш-значення m: $m = h(M)$, $1 < m < q$, потім генерує випадкове ціле число K, $1 < K < q$, і обчислює число r: $r = (G^K \bmod P) \bmod q$. Потім відправник обчислює за допомогою секретного ключа X ціле число s:

$$s = \frac{m + r * X}{K} \bmod q \ .$$

Пара чисел r і s утворить цифровий підпис $S = (r, s)$ під документом M. Таким чином, підписане повідомлення являє собою трійку чисел $[M,r,s]$. Одержувач підписаного повідомлення $[M, r, s]$ перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна із цих умов не виконана. Потім одержувач обчислює значення $w = 1/s \bmod q$, геш-значення $m = h(M)$ і числа $u_1 = (m*w) \bmod q$, $u_2 = (r*w) \bmod q$. Далі одержувач за допомогою відкритого ключа Y обчислює значення $v = ((G^{U1} * Y^{U2}) \bmod P) \bmod q$ і перевіряє виконання умови $v = r$. Якщо умова $v = r$ виконується, тоді підпис $S = (r,s)$ під документом M визнається одержувачем справжнім.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1.

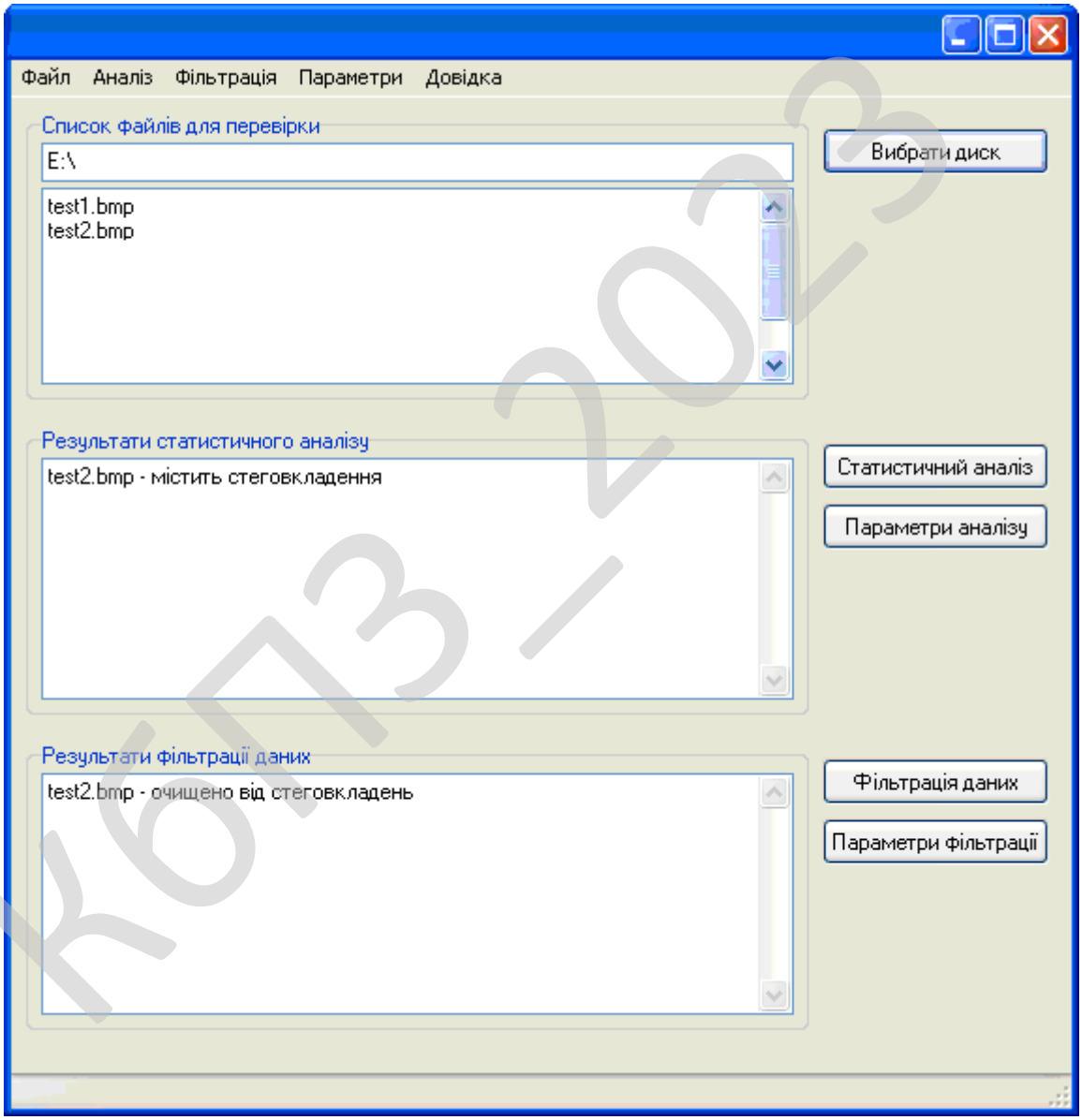

Рисунок 5.1 – Головне вікно програми

Для початку роботи з програмою користувач має під'єднати до комп'ютера змінний носій (флеш-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 зображене на рисунку 5.2.

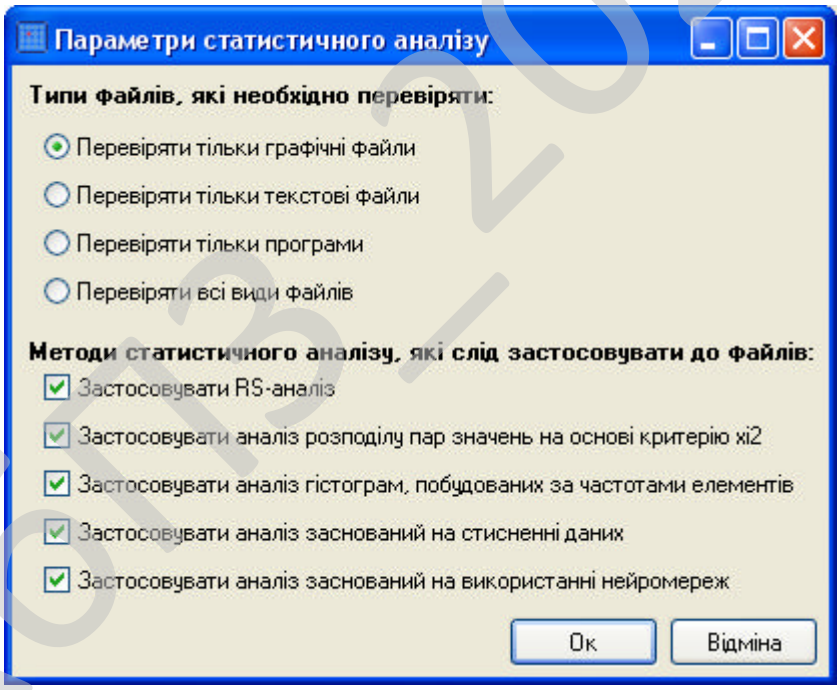

Рисунок 5.2 – Вікно вибору параметрів статистичного аналізу

В даному вікні можна вибрати наступні параметри аналізу файлів:

1. Типи файлів, які необхідно перевіряти:
   1) тільки графічні файли;
   2) тільки текстові файли;

3) тільки програми;

4) всі види файлів.

2. Методи перевірки файлів на вкладення, які слід використовувати:

1) RS-аналіз;

2) аналіз розподілу пар значень на основі критерію xi2;

3) аналіз гістограм, побудованих за частотами елементів;

4) аналіз заснований на стисненні даних;

5) аналіз заснований на використанні нейромереж.

По натисненні на кнопку **Фільтрація даних** здійснюється видалення стеговкладень зі змінного носія. Якщо є можливість видалити вкладення не пошкодивши сам файл, то видаляється лише вкладення, якщо ж такої можливості немає, то видаляється повністю весь файл. При видаленні файлу з'являється діалогове вікно, що містить запит до користувача на дозвіл виделення файлу. Результати процедури фільтрації даних виводяться у текстовому полі **Результати фільтрації даних** у вигляді списку, що містить назви файлів та через дифіс дії з ними виконані (очищення від вкладення чи видалення).

По натисненні на кнопку **Параметри фільтрації** з'являється діалогове вікно зображене на рисунку 5.3.

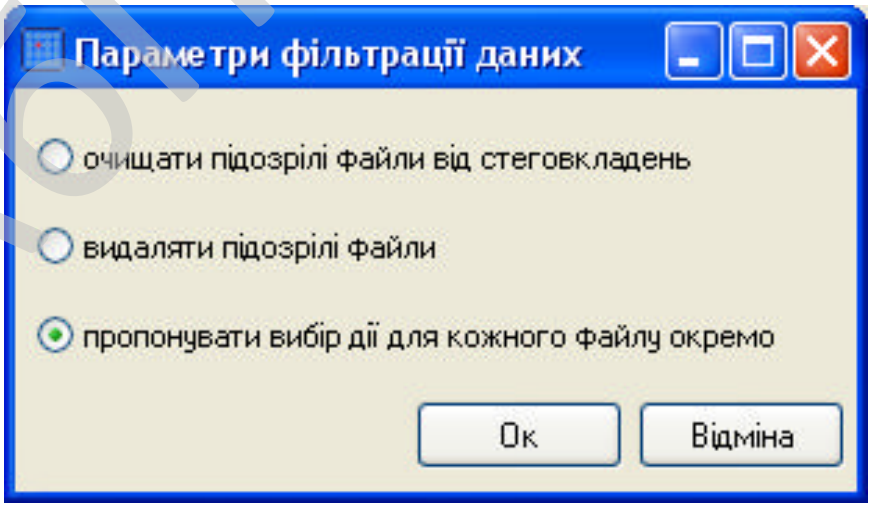

Рисунок 5.3 – Вікно вибору параметрів фільтрації даних

В даному вікні можна вибрати наступні параметри фільтрації даних, що визначають дії, які слід виконувати з файлами, що містять вкладення:

1. Очищати підозрілі файли від стеговкладень.

2. Видаляти підозрілі файли.

3. Пропонувати вибір дії для кожного файлу окремо.

На рисунку 5.4 зображене вікно «Про програму…», що з'являється при натисненні на пункт меню користувача **Довідка->Про програму…**.

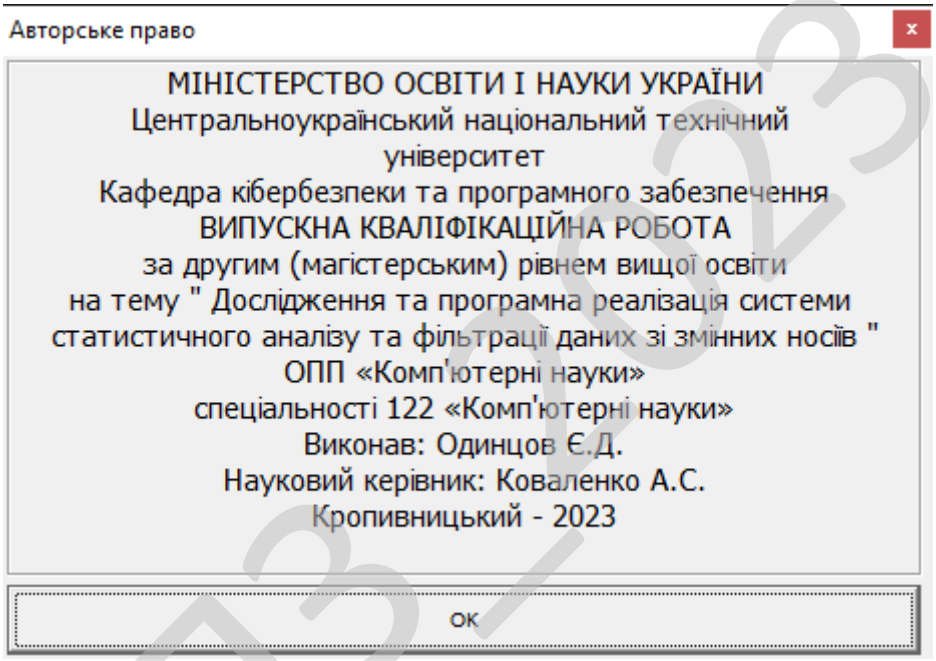

Рисунок 5.4 – Вікно «Про програму…»

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи статистичного аналізу та фільтрації даних зі змінних носіїв.

Метою розробки є дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв.

Об'єктом дослідження є процес статистичного аналізу та фільтрації даних зі змінних носіїв.

Предметом дослідження є методи статистичного аналізу та фільтрації даних зі змінних носіїв.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод статистичного аналізу та фільтрації даних зі змінних носіїв.

– Розроблено вітчизняний продукт статистичного аналізу та фільтрації даних зі змінних носіїв, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	17
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	7
8. Кількість форм вихідної інформації.	–	8
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	2
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	1
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	3
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	1
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	1
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	1
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	3
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	3
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	1
29. Досвід роботи з програмними інструментами розробки (1-6)	–	2
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	3
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	17000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(3{,}24+3{,}64+3{,}38+4{,}94+2{,}73) = 1{,}028.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}028} = 6{,}8 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}8 \cdot (0{,}88 \cdot 1 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}89 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1 \cdot 1{,}1 \cdot 1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}22 \cdot 1{,}12 \cdot 1 \cdot 1{,}1) = 8{,}38 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4).

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо *S* = 90 %

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 8{,}38^{0{,}33+0{,}2(1{,}028-1{,}01)} \cdot 100 = 177 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	15	Д7
Робочий проект	177	Ф 7.1-7.4
Впровадження	17	Д13
Всього	228	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{228 \cdot 1}{48\text{-}5} = 5,3 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	8	720	12
Монітор	60	8	480	8
Клавіатура	30	8	240	4
Маніпулятор «мишка»	30	8	240	4
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	8	240	4
Кабельні господарства ЛОМ на 1 м.п	2,5	180	450	7,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	45,16

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{45 \cdot 2}{1,2} = 75 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 75/(48 \cdot 8) = 0{,}2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12135	24270
Продакт-менеджер	0,25	12000	6000
Інженер-програміст	5,3	13000	137800
Інженер-електронщик	0,2	7500	3000
Інженер-системотехнік	0,25	7500	3750
Адміністратор мережі	0,5	7500	7500
Системний програміст	0,25	7500	3750
Дизайнер WEB	0,25	7500	3750
Інженер-верстальник	0,25	7700	3850
Бухгалтер-економіст	0,25	12500	6250
Всього за період розробки	$R_{сп}$ = 8,5	-	$Ф_{роб}$ = 199920

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{199920}{8,5 \cdot 48} = 490 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 9 \cdot 8 \cdot 20000 = 1440000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 144000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 9 \cdot 3500 = 31500 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми supercomp.kiev.ua за 29.10.23 – джерело http: //supercomp.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		12721
Системний блок		7721
Процесор	AMD A8-Series A8-9600 A8 9600 3,1 ГГц 6 Вт, він працює з відеокартою Radeon R (384 шейдерних ядер / 6 обчислювальни ядер) і двоканальним контролером пам'ят DDR4-18600	2000
Системна плата	GIGABYTE B450M S2H V2, 1 x PCI-E 2. x1, 1 x PCI-E 3.0 x16, 4 x USB 3.1 Gen1, 1 VGA, 1 x RJ45, 2 x USB 2.0, 2 x PS/2, 1 DVI-D, 1 x HDMI, 1 x optical SPDIF out, 3 Audio, M.2 2280, 4 x Sata 6.0 Gb/s	1350
Відеокарта	Інтегрована AMD Radeon R7	-
Жорсткий диск	480 GB SSD	1490
Оперативна пам’ять	DDR4 8GB 2666 MHZ KINGSTO (KVR26N19S8/8)	834
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	416
Корпус	Logicpower 8702 – 550w 12cm	1411
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	220
інше	Клавіатура, мишка	Подаруно
Монітор	Монітор BenQ GL2450HM Black	3600

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	Powercom BNT-600AP USB	1400

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межа до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складем таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	9	12721	11448,9	125937,9
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	147569,4

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
Група 3			
1. Будівлі	1440000	-	-
2. Передавальні пристрої	144000	-	-
Всього по групі	1584000	5	79200
Група 4			
3. Обчислювальна техніка	147570	-	-
Всього по групі	147570	40	59028
Нематеріальні активи			
4. Нематеріальні активи	17000	10	1700
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	121875	20	24375
7. Господарський інвентар	31500	25	7875
Всього по групі	162406	-	34507,75
Разом	$К_р$ = 1910976		$А_р$ = 174435,75

Примітка: вартість автомобіля Daewoo Lanos 2010 взята по даним електронного ресурсу https://auto.ria.com/uk/auto_daewoo_lanos_33592444.html та складає 121875 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 490 \cdot 228/17 = 6579 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 6579 \cdot 10 \cdot 0{,}01 = 658 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(6579+658) = 1592 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 5 пачек паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=206 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 206 \cdot 5 \cdot 1 = 1030 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 17):

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 35,6 грн./шт., DVD-R box – 55 грн./шт.

$$З_{М2} = 17 \cdot 55 = 935 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 500+508+1155 = 2163 \text{ грн.}$$

$$З_М = (1030 + 935 + 2163)/17 = 243 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 6579 \cdot 15 \cdot 0{,}01 = 987 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 17 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 174436 \cdot 2/(17 \cdot 12) = 1710 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	6579
2. Додаткова зарплата виконавців	$З_д$	658
3. Відрахування на соціальні потреби	$С_{оц}$	1592
4. Загальногосподарські витрати	$Г_{осп}$	987
5. Витрати на матеріали	$З_М$	243
6. Освоєння нових операційних систем, мов програмування	$О_п$	987
7. Амортизація основних фондів	$А_м$	1710
8. Повна собівартість програмного забезпечення	$С_п$	12756
9. Плановий прибуток	$П_р$	6378
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	19134
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3826,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	22960,8

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 6579+658+1592+987+243+987+1710 = 12756 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 12756 = 6378 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	22961
Всього капітальних витрат	–	22961

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	173923	57974
2. Витрати на електроенергію	$З_{ел}$	562	389
3. Витрати на амортизацію	$З_{ам}$	0	5740
Всього витрат за рік	$І$	174485	64103

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \quad (7.23)$$

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 80 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\ баз} = 240 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 \cdot 9 = 173923\ грн,$$

до:

$$З_{р\ нов} = 80 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 \cdot 9 = 57974\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 9 \cdot 0{,}15 \cdot 130 \cdot 3{,}2 = 562 \text{ грн.}$$

$$З_{ел\ нов} = 9 \cdot 0{,}15 \cdot 90 \cdot 3{,}2 = 389 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	22961	–	5740,25
Всього відрахувань	-	–	22961	–	5740,25

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.24)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (19134 - 12756) \cdot 17 - (0,05 \cdot 1584000 + 0,4 \cdot 147570 + $$
$$+0,2 \cdot 121875 + 0,25 \cdot 40531 + 0,1 \cdot 17000) \cdot 2/12 = 79353 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.25)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{326976}{(19134 - 12756) \cdot 17 \cdot 12 / 2} = 0,5 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	17
2. Повна собівартість розробленої програми	Грн.	12756
3. Ціна розробленої програми	Грн.	19134
4. Плановий прибуток від реалізації розробленої програми	Грн.	6378
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1910976
7. Загальний прибуток від реалізації програмної продукції	Грн.	108426
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	79353
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	22961
11.Величина економічного ефекту у користувача програмної продукції	Грн.	104642
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,2

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (174485\text{-}64103)\text{-}0{,}25 \cdot 22961 = 104642 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \quad (7.28)$$

$$T_{сп} = \frac{22961}{174485 - 64103} = 0{,}2 \text{ року}.$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Наявний в даний час в нашій країні комплекс розроблених організаційних заходів та технічних засобів захисту, накопичений передовий досвід роботи ряду обчислювальних центрів показує, що є можливість домогтися значно більших успіхів у справі усунення впливу на працюючих небезпечних і шкідливих виробничих факторів. Проте стан умов праці та його безпеки в ряді обчислювальних центрів (ОЦ) та підприємств ще не задовольняють сучасним вимогам. Оператори ЕОМ, оператори підготовці даних, програмісти та інші працівники ОЦ та підприємств ще стикаються з впливом таких фізично небезпечних і шкідливих виробничих факторів, як підвищений рівень шуму, підвищена температура зовнішнього середовища, відсутність або недостатня освітленість робочої зони, електричний струм, статична електрика і інші.

Багато працівників ОЦ та підприємств пов'язані з впливом таких психофізичних факторів, як розумова перенапруга, перенапруження зорових і слухових аналізаторів, монотонність праці, емоційні перевантаження. Вплив зазначених несприятливих факторів призводить до зниження працездатності, викликане розвиваються втомою. Поява і розвиток втоми пов'язане зі змінами, які виникають під час роботи в центральній нервовій системі, з гальмівними процесами в корі головного мозку. Наприклад сильний шум викликає труднощі з розпізнаванням колірних сигналів, знижує швидкість сприйняття кольору, гостроту зору, зорову адаптацію, порушує сприйняття візуальної інформації, зменшує на 5 – 12% продуктивність праці. Тривала дія шуму з рівнем звукового тиску 90 дБ знижує продуктивність праці на 30 – 60%.

Медичні обстеження працівників ОЦ та підприємств показали, що крім зниження продуктивності праці високі рівні шуму призводять до погіршення

слуху. Тривале перебування людини в зоні комбінованого впливу різних несприятливих факторів може призвести до професійного захворювання. Аналіз травматизму серед працівників ВЦ показує, що в основному нещасні випадки відбуваються від впливу фізично небезпечних виробничих факторів при заправці носія інформації на обертовий барабан при знятому кожусі, під час співробітниками невластивих їм робіт. На другому місці випадки, пов'язані з дією електричного струму.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машина (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5
Довжина	5,95
Висота	2,8

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	7,4
Об'єм, V	$м^3$	не менше 20.0	20,8

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють четверо людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце програміста не відповідають

нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно **Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, р**обота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми україни "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи

приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значенн;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання.

Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Освітлення приміщень, де працюють спеціалісти з IT, є важливим фактором для забезпечення комфорту, продуктивності та здоров'я працівників. Недостатнє або надмірне освітлення може призводити до втоми, стресу, головного болю, порушення зору та інших проблем. Для того, щоб уникнути цих проблем, необхідно дотримуватися таких рекомендацій:

– Вибирати джерела світла, які мають природний спектр, не мерехтять і не випромінюють шкідливе випромінювання. Наприклад, LED-лампи або енергозберігаючі лампи.

– Регулювати яскравість і контрастність монітора так, щоб вони не були надто яскравими або тьмяними. Застосовувати режим "Night Shift" або "Dark Mode" для зменшення навантаження на очі.

– Розташовувати монітор так, щоб він не відбивав світло від вікна або лампи. Уникати прямого сонячного світла на екрані або обличчі.

– Забезпечувати рівномірне освітлення робочого місця, уникаючи тіней і блисків. Використовувати локальне освітлення для покращення видимості документів або інших предметів.

– Дотримуватися норм освітленості для роботи з комп'ютерами, які становлять 300-500 лк. Вимірювати освітленість за допомогою спеціальних приладів або мобільних додатків.

– Робити перерви в роботі з комп'ютером кожні 40-60 хвилин і робити вправи для очей. Дивитися на віддалені об'єкти, моргати часто, закривати очі на кілька секунд.

Для розрахунку штучного освітлення приміщень, де працюють спеціалісти ІТ, можна використовувати такі методи:

– Метод коефіцієнта використання світлового потоку – цей метод дозволяє врахувати як прямий, так і відбитий світловий потік від стелі, стін і робочої поверхні. Для цього методу потрібно знати нормовану освітленість, площу приміщення, коефіцієнти запасу, нерівномірності і використання світлового потоку.

– Метод питомої потужності – цей метод полягає в тому, що для кожного типу приміщення і роботи визначається питома потужність освітлювальної установки за таблицями. Для цього методу потрібно знати площу приміщення, коефіцієнт запасу і питомий світловий потік ламп.

– Точковий метод – цей метод базується на розрахунку умовної освітленості, яка створюється одним світильником з лампою світловим потоком 1000 лм. Для цього методу потрібно знати геометричні розміри приміщення, висоту підвісу світильників, коефіцієнти запасу і нерівномірності, а також користуватися ізолюксами умовної горизонтальної освітленості

Виконаємо розрахунок освітлення методом коефіцієнта використання світлового потоку – це метод розрахунку штучного освітлення, який дозволяє визначити кількість і потужність світильників, необхідних для забезпечення заданої освітленості на робочих поверхнях.

Для розрахунку освітлення за методом коефіцієнта використання світлового потоку потрібно знати такі параметри

– Площа робочої поверхні S, м^2 – для приміщення 50 м^2

– Нормована освітленість E, лк – Нормована освітленість E_N – це рівень освітленості, який встановлюється за європейським стандартом EN 12464-1:2011 для різних типів внутрішніх робочих місць. Цей стандарт визначає мінімальну освітленість, яка повинна бути забезпечена на поверхні завдання, на поверхні безпосереднього оточення і на фоновій поверхні – 100 лк.

Світлодіодні лампи мають найвищу світлову віддачу, тобто вони видавлюють найбільше світла на одиницю споживаної енергії. Так, світлодіодна лампа потужністю 10 Вт має світловий потік близько 800 лм. Приймаємо 800 лм.

– Коефіцієнт використання світлового потоку K – Коефіцієнт використання світлового потоку залежить від геометрії приміщення, характеристик світильників і коефіцієнтів відбиття поверхонь. Чим ближче світильники до стелі і стін, тим менше світла доходить до робочої поверхні. Чим ближче робоча поверхня до підлоги, тим більше світла вона отримує. Чим більші коефіцієнти відбиття стелі, стін і підлоги, тим більше світла повертається до робочої поверхні.

– Коефіцієнт запасу Z – Коефіцієнт запасу залежить від типу і частоти обслуговування світильників, ламп і поверхонь, а також від режиму роботи освітленн. Зазвичай коефіцієнт запасу приймається в межах 1,1-1,5. Приймаємо 1,1.

Для розрахунку кількості світильників в приміщенні, вам потрібно врахувати рівень освітлення, світловий потік кожного світильника, коефіцієнт використання і площу приміщення.

Загальний алгоритм розрахунку:

– **Визначте бажаний рівень освітлення (E):** Це рівень освітлення, який вам потрібно в приміщенні. Він вимірюється в люксах (лк). Для офісних приміщень зазвичай рекомендується рівень освітлення близько 300-500 лк, залежно від конкретних потреб.

– **З'ясуйте світловий потік кожного світильника (F):** Це кількість світла, яке виділяє кожен світильник і вимірюється в люменах (лм). Інформацію про світловий потік зазвичай надає виробник світильника.

– **Визначте коефіцієнт використання (K_c):** Цей коефіцієнт вказує на частину світла, яка досягає поверхні, де працюють співробітники. Виробники світильників також надають інформацію про цей параметр.

– **Визначте виправлення світлового потоку (K):** Виправлення світлового потоку враховує втрати світла через пил, забруднення та відбиття від поверхонь. Для офісних приміщень, де чистота зазвичай добра, цей коефіцієнт може бути близько 0,9.

– **Виміряйте площу приміщення (S):** Виміряйте площу приміщення, в якому планується розташування світильників, в квадратних метрах (м²).

– **Використовуйте формулу для розрахунку кількості світильників (N):** Використовуючи вище наведені значення, використовуйте наступну формулу:

$$N = \frac{E \cdot S \cdot Z}{F \cdot K \cdot Kc} \textbf{ (1).}$$

Визначемо кількості світильників:

Результат розрахунку дозволить визначити кількість світильників, яку потрібно встановити в приміщенні, щоб досягти бажаного рівня освітлення.

Якщо бажаний рівень освітлення E = 500 лк, світловий потік кожного світильника F = 800 лм, коефіцієнт використання K = 0,8, виправлення світлового потоку Кс = 0,9, і площа приміщення S = 50 м², то кількість світильників буде:

$$N = \frac{500\,лм \cdot 50\,м2 \cdot 1.1}{800 \cdot 0.8 \cdot 0.9} = 47.7.$$

Отже, вам потрібно встановити приблизно 48 світильників з світлодіодними лампами потужністю 10 Вт, які мають світловий потік приблизно 800 лм світильників, щоб досягти бажаного рівня освітлення у цьому приміщенні.

Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність роботи ІТ спеціалістів вцілому.

У розділі здійснено аналіз загальних умов праці, призначеного для праці програмістів, розглянуто та проаналізовано небезпечні та шкідливі фактори, які негативно впливають на ІТ спеціалістів під час професійної діяльності.

Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я ІТ спеціаліста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи статистичного аналізу та фільтрації даних зі змінних носіїв.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів статистичного аналізу та фільтрації даних зі змінних носіїв.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем статистичного аналізу та фільтрації даних зі змінних носіїв.

– Досліджена система статистичного аналізу та фільтрації даних зі змінних носіїв.

– На основі отриманих результатів досліджень створена програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання статистичного аналізу та фільтрації даних зі змінних носіїв.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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. 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 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 DSA.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 104642 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Одинцов Є.Д. Дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Vijay Kumar Velu. Mastering Kali Linux for Advanced Penetration Testing. Packt Publishing Ltd. 2022. 573 p.
3. Josh Armitage. Cloud Native Security Cookbook. O'Reilly Media. 2022. 516 p.
4. Massimo Bertaccini. Cryptography Algorithms. Packt Publishing. 2022. 358 p.
5. Alyssa Miller. Cybersecurity Career Guide. Manning Publications. 2022. 368 p.
6. Awais Rashid, Howard Chivers, George Danezis, Emil Lupu, Andrew Martin. CyBOK The Cyber Security Body of Knowledge. The National Cyber Security Centre. 2019. 854 p.
7. Loren Kohnfelder. Designing Secure Software. No Starch Press. 2022. 332 p.
8. Samir Kumar Rakshit. Ethical Hacker's Penetration Testing Guide. BPB Online. 2022. 509 p.
9. Corey J. Ball. Hacking APIs. No Starch Press. 2022. 353 p.
10. Kevin Beaver. Hacking for Dummies. John Wiley & Sons. 2022. 419 p.
11. Mark S. Merkow. Practical Security for Agile and DevOps. CRC Press. 2022. 236 p
12. Derek Fisher. Application Security Program Handbook. Manning Publications. 2021. 155 p.
13. Cameron Wyatt PH.D. Kali Linux Tutorial. Independently published. 2021. 60 p.

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

15. 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,

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

17. 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.

18. 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

19. 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.

20. 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.

21. 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.*

22. 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.

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

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

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

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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.

45. Smirnov, O., Kuznetsov, A., Kiian, A., Babenko, B., Zhosan, H., Prokopovych-Tkachenko, D., «Soft Decoding Method for Turbo-Productive Codes», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT 2019*, Lviv, Ukraine, 2-6 July, 2019, P. 129-134.

46. 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.

47. 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.

48. 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.

49. 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.

50. Smirnov A.A., Kuzne tsov 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.

51. Вінтенко Б.Ю., Смірнов О.А., Коваленко О.В., Смірнов С.А., Коваленко А.С. «Дослідження нормативних документів та галузевих стандартів розробки програмного забезпечення комп'ютерних систем управління АЕС, важливих для безпеки». *Системи управління, навігації та зв'язку*, 2023, вип. 2(72), С. 170-178.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи статистичного аналізу та фільтрації даних зі змінних носіїв.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 32-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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 з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи статистичного аналізу та фільтрації даних зі змінних носіїв;

– 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.

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 архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

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#.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 114 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 11.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Коваленко А.С.

Дослідження та програмна реалізація системи статистичного аналізу та фільтрації даних зі змінних носіїв

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 98

Літера: РП

Кропивницький – 2023 року

// st_analysis.cs – головний модуль програми, що реалізує статистичний аналіз даних

```
#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;
```

```
    }

    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)
    {
        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++)
            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);
```

```
    }
    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];
        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]);
    }
```

```
    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;
```

```
        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;
```

```
                    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
                    //
```

```
                    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);
```

```
                    //
                    // 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
```

```
                    / / * 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
```

```
            { 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;
```

```
            //
            // 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 )
```

```
                        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
```

```
            { 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);
```

```
                    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 )
```

```
                        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 )
```

```
                        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}};
```

```
                    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);
```

```
                    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 )
```

```
                        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);
```

```
                    _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
```

```
            //      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 )
```

```
                        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;
```

```
            //
            // 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 )
```

```
                        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;
```

```
                    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
```

```
            //
            //
            _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");
```

```
            _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
            //
```

```
            //
            _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;
```

```
                    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;
```

```
                    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};
```

```
                    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 )
```

```
                        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;
```

```
                    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 )
```

```
                    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);
```

```
                    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);
```

```
                    _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);
```

```
                    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;
```

```
                    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;
```

```
                    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 )
```

```
                        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
```

```
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);
```

```
                    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; }
```

```
            }
            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;
```

```
                    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);
```

```
                                        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);
                }
```

```
            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);
```

```
                    _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);
```

```
                    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 )
```

```
                    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;
```

```
                    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 )
```

```
                        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;
```

```
                    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
            //
```

```
            //
            _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 )
```

```
                        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
```

```
            { 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};
```

```
                    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 )
```

```
                        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;
```

```
                    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;
```

```
                    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 )
```

```
                        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);
```

```
                    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 )
```

```
                        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);
```

```
                    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);
```

```
                    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);
```

```
                    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 )
```

```
                        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;
```

```
                    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;
```

```
            //
            // 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
            //
            //
```

```
            _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
                {
```

```
                    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);
```

```
                    _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 )
```

```
                    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; }
```

```
            }
            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;
```

```
                    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 )
```

```
                        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);
```

```
                    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 )
```

```
                        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
            //
```

```
            //
            _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); }
```

```
                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);
```

```
                    _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);
```

```
                    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
                {
```

```
                    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};
```

```
                    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 )
```

```
                        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;
```

```
            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 )
```

```
                        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};
```

```
                    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;
```

```
                    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);
```

```
                    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;
```

```
            //
            // 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);
```

```
                    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 )
```

```
                    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
            //
            _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);
                _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;
  }
}
```