Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: проектування інформаційної системи автоматизації реєстрації сигналів моделювання механічних рухів кінцівки на основі об'єктно-орієнтованих технологій

Виконав(ла): студент(ка) 4 курсу, групи СП-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Колодій О. О. (прізвище та ініціали)

Керівник (підпис) Петрик М. Р. (прізвище та ініціали)
Нормоконтроль (підпис) Стоянов Ю. М, (прізвище та ініціали)
Завідувач кафедри (підпис) Петрик М. Р. (прізвище та ініціали)
Рецензент (підпис) Жаровський Р.О. (прізвище та ініціали)

Тернопіль
2023

РЕФЕРАТ

Кваліфікаційна робота бакалавра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2023. Сторінок 71, таблиць 30, рисунків --, презентація

Тема: Проектування інформаційної системи автоматизації реєстрації сигналів моделювання механічних рухів кінцівки на основі об’єктно-орієнтованих технологій

Метою кваліфікаційної роботи було створення програмного забезпечення для автоматичної діагностики есенціального тремору. Були використані методи математичного та комп’ютерного моделювання для обробки даних, які будуть зібрані в результаті відповідних досліджень. Спеціально для цього завдання будуть розроблені моделі та алгоритми, щоб точніше та ефективніше діагностувати це захворювання.

У результаті було розроблено та реалізовано набір математичних моделей та комп’ютеризованих методів обробки вхідної інформації спірального тесту, що дозволяє отримати якісні та кількісні характеристики досліджуваного захворювання.

Для автоматичного визначення ступеня есенціального тремору реалізована бібліотека з набором необхідних алгоритмів у вигляді модулів програмної системи. Розробка виконується з використанням мови програмування Java та програмного комплексу MatLab.

Ключові слова: комп’ютерне моделювання, математична модель, проектування та розробка, автоматизація, тремор, діагностика, алгоритми, когнітивні зв’язки, програмно-апаратний комплекс, спеціалізоване програмне забезпечення.

ANNOTATION

Bachelor's qualification work. Ivan Pul'uj Ternopil National Technical University, Department of Software Engineering, Specialty 121 "Software Engineering". TNTU, 2023. Pages 71, tables 0, figures 30, presentation.

Topic: Design of an information system for automating the registration of motion modeling signals of a limb based on object-oriented technologies.

The aim of the qualification work was to develop software for automatic diagnosis of essential tremor. Mathematical and computer modeling methods were used to process the data collected as a result of relevant research. Specifically for this task, models and algorithms were developed to diagnose this condition more accurately and efficiently.

As a result, a set of mathematical models and computerized methods for processing input information of a spiral test were developed, which allows obtaining qualitative and quantitative characteristics of the investigated condition.

For the automatic determination of the degree of essential tremor, a library of necessary algorithms in the form of modules of the software system was implemented. The development is carried out using the Java programming language and the MatLab software complex.

Keywords: computer modeling, mathematical model, design and development, automation, tremor, diagnosis, algorithms, cognitive connections, hardware-software complex, specialized software.

ЗМІСТ

ВСТУП

Есенціальний тремор є найпоширенішим захворюванням, яке впливає на рухову систему. Його також називають доброякісним, спадковим або ідіопатичним тремором. У людей з цим захворюванням може виникати неконтрольоване тремтіння рук, голови або інших частин тіла.

Зазвичай цей стан розвивається у дорослих і з часом може погіршуватись. Тремор найбільше помітний, коли людина тримає руки перед собою або робить прості рухи, наприклад, тримає чашку, їсть ложкою або пише. Однак, якщо руки повністю розслаблені, наприклад, коли вони спочивають на колінах, тремор зазвичай припиняється. Стрес також може погіршувати тремор на певний час.

Застосування сучасних інформаційних технологій в медицині є важливим для покращення точності діагностики захворювань. Тремор, який може мати різну причину і механізм виникнення, є однією з найпоширеніших порушень рухової функції у людей. Його діагностика є актуальною міждисциплінарною проблемою для неврологів, нейрофізіологів та нейрохірургів. За даними Всесвітньої організації охорони здоров'я, в Україні близько 6% населення віком від 65 років і старше і 3,8% осіб до 40 років мають прояви патологічного тремору.

Метою моєї роботи є створення програмного забезпечення, яке автоматично діагностує есенціальний тремор.

Завдання полягає у зборі інформації, її систематизації та приведенні до зручної однакової форми, розміщення її в застосунку і коректного відображення.

Встановлення діагнозу «есенціальний тремор» може бути складним через наявність різних видів тремтіння та їх подібні прояви при різних порушеннях нервової системи. Використання системного підходу, який включає класифікацію тремору, визначення його причин та опис його характеристик на нейрофізіологічному рівні, може покращити якість діагностики та допомогти встановити ефективний лікування.

Предметом дослідження є використання різних методів для збору та аналізу даних про рухи кінцівок людини, створення математичних моделей для цифрового відтворення та аналізу цих даних, а також використання комп'ютерної обробки даних для розробки технології автоматизованої діагностики есенціального тремору у пацієнтів.

Технологічний прогрес дозволяє отримувати результати досліджень швидше, точніше та з більшою наочністю, а також візуалізувати їх. Завдяки розробці ефективних алгоритмів та вибору відповідної архітектури, такі комп'ютерні системи стають необхідними для наукових та технологічних процесів.

Автоматизація діагностики захворювань, зокрема через тест спіралі, включає аналіз цифрових вхідних даних. З використанням розробленої програмної системи та комп'ютерної моделі можна отримати якісне аналітичне та чисельне представлення рухів рук пацієнтів.

1 АНАЛІЗ ВИМОГ ДО ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Проблематика та способи виявлення, діагностування та збору даних про стан пацієнтів з симптомами тремору

Збереження здоров'я людини є найважливішим завданням, на яке зобов'язані звертати увагу всі. Сучасне суспільство все більше використовує комп'ютерні методики для діагностики та лікування різних хвороб. Завдяки новітнім науковим розробкам і автоматизації, ці процеси стають надзвичайно ефективними, швидкими та корисними. Також важливим є дослідження та математичне моделювання цих методик. Комп'ютерне моделювання відіграє велику роль у вивченні різних галузей науки і техніки і має незамінний внесок у ці дослідження.

Тремор є поширеною хворобою, для діагнстування якої дуже корисним є використанння автоматизованих методів збору і обробки зібраних даних.

Тремор – це мимовільне, інколи ритмічне скорочення м’язів, яке спричиняє поступальні рухи, коливання та сіпання частин тіла. Подальший розвиток даної хвороби, в ретші-решт призводить до параліачу нервової системи або хвороб Паркінсона чи Альцгеймера [1]. У медичній та біотехнічній галузях вже довго займаються методикою визначення, аналізу і оцінки захворювання на есенціальний тремор.

В процесі вивчення проблематики мною було визначено такі основні проблеми автоматизованої ідентифікації захворювання:

- неактуальні методи проведення тестування хворих;
- необ’єктивне оцінювання результатів;
- невисока точність встановлення важкості хвороби;
- високовартісність та складність втілення сучасних високо технологічних способів діагностики в лікарнях.

Цілий ряд причин говорить про високу потребу в реформуванні методів встановлення хвороб цього типу. Складність виявлення самого захворювання та

ступеню його важкості виникає через наявність недосконалих методів його встановлення.

Одним з ефективних методів діагностики є тест з використанням рисунку спіралі, який дає змогу одержати кількісні властивості тремору, аналізуючи відмінності малюнка пацієнта від ідеального шаблону.

Для того, щоб результат був більш точним, можна скористатися комп'ютерним аналізом та математичним моделюванням. Ці методи дозволяють одержати додаткові показники, такі як частотні властивості та амплітуда коливань.

Для тесту спіралі необхідно порівняти взірцеву спіраль з тою, яку намалював пацієнт на планшеті зі стилусом. Він є більш якісною оцінкою різних типів захворювань на тремор [12,13].

Спіралі, отримані в результаті тестування пацієнтами, можуть бути аналізовані двома способами: візуальним спостереженням та комп'ютерним аналізом з використанням математичних моделей.

Комп'ютерний аналіз є більш ефективним та точним у визначенні ступеня тремора. Хочу зазначити, що математичне моделювання дає змогу одержати частотні властивості, амплітуду коливань, відхилення від взірця, а також всі інші показники, які є важливими при діагностуванні за допомогою цього методу дослідження.

1.2 Складність встановлення точного ступеня хвороби тремор

Результатом використання недосконалих та застарілих методик встановлення діагнозу в лікарнях наявні проблеми складності оцінювання хвороби тремтіння та її ідентифікації.

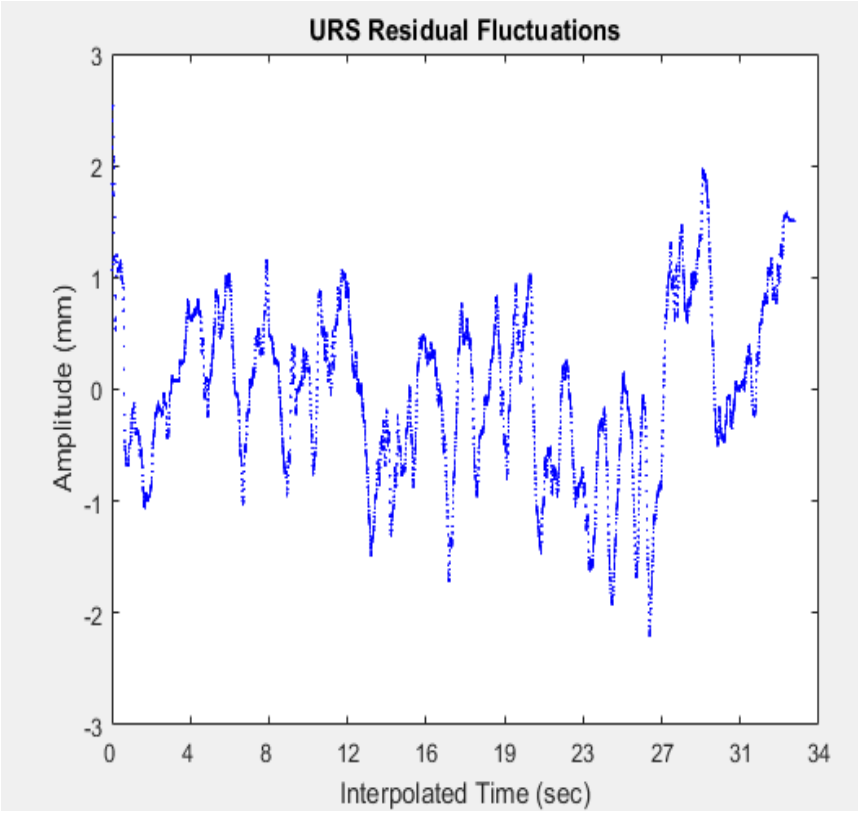
Дуже довго для спостереження за розвитком такої хвороби використовується рисунок спіралі. В результаті виконання тесту пацієнтами, ці спіралі можуть бути проаналізовані візуальним спостереженням або комп’ютерним аналізом. Останній є набагато ефективнішим та точнішим в оцінці ступеню хвороби. В свою чергу, математичне моделювання дає змогу одержати частотні властивості, амплітуду коливань, відхилення від взірця, а також всі інші показники.

Етапи виконання діагностування наступні:

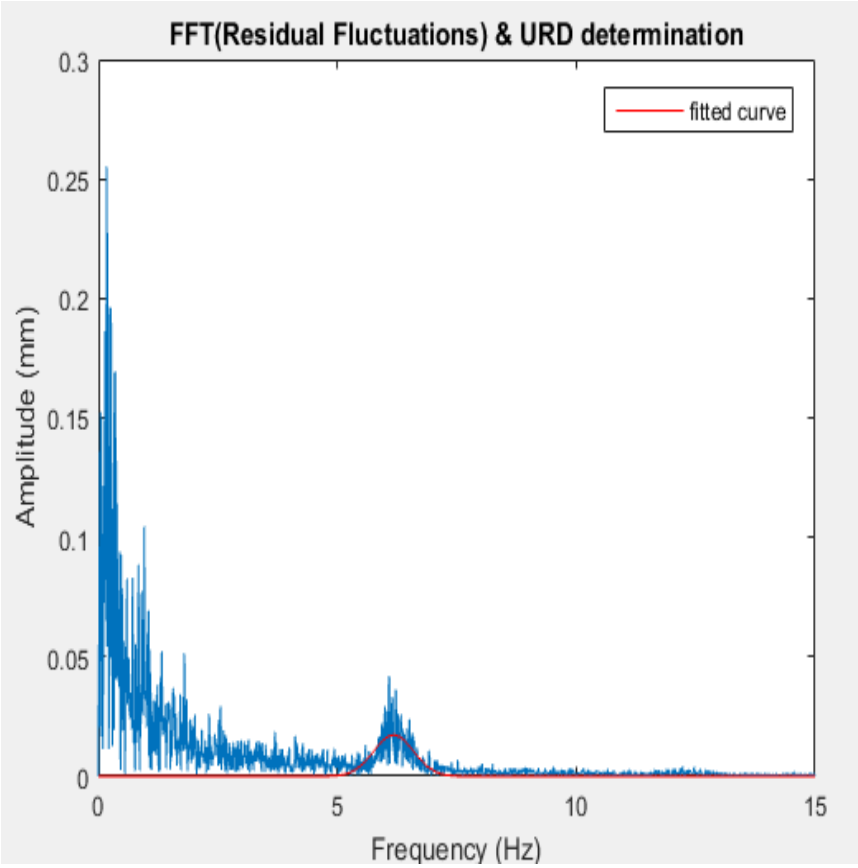
- Тестування спіраллю з пацієнтом, отримання вхідних даних,

- Опрацювання вхідних масивів даних, використовуючи розроблювані методи математичного і статистичного аналізу. Отримання кількісних характеристик (амплітудної та частотної, рисунок 1.1).

- Отримання оцінки і висновку про стан захворювання пацієнта за допомогою шкали Фанн-Толосса-Марін [3].



а)



б)

Рисунок 1.1 - Амплітудна(а) та частотна(б) характеристики

Методи обробки даних включають розгортання рисунку спіралі пацієнта у форму кривої, порівняння цієї кривої з усередненим значенням кривої відносно ідеального шаблону Архімедової спіралі, вимірювання величини відхилення кривої пацієнта від шаблону та отримання частотних характеристик за допомогою перетворення Фур'є. Проблема кількісної оцінки коливань під час тремору вирішується шляхом аналізу окремих сегментів даних. На жаль, ці рейтингові оцінки забезпечують лише суб'єктивні результати значень амплітуди тремору. Вони не становлять жодної грунтовної кількісної оцінки частоти тремору.

Основними елементами, що використовуються в моєму дослідженні є:

- Сукупні методи діагностування
- Розроблювані технології та методики.

- Математичні та комп’ютерні моделі.
- Спеціалізовані програмні та технічні засоби.

Зазвичай, під час процесу встановлення необхідних параметрів, покращується якість та зрозумілість в поведінці системи.

В процесі аналізу предметної області я вияв складові, характеристику і методи опрацювання яких необхідно розглянути:

- методологія аналізу даних;
- структура вхідних даних для аналізу;
- перетворення, що найбільше підходять для ефективної обробки даних;
- шукані характеристики та підсумкові оцінки;
- важлива для користувача інформація.

Для системи діагностування властиві такі терміни та поняття: діагноз, медичний працівник, лабораторія, пацієнт, діагностичний тест, діагностичне обладнання, результати, оцінювання ступеню захворювання.

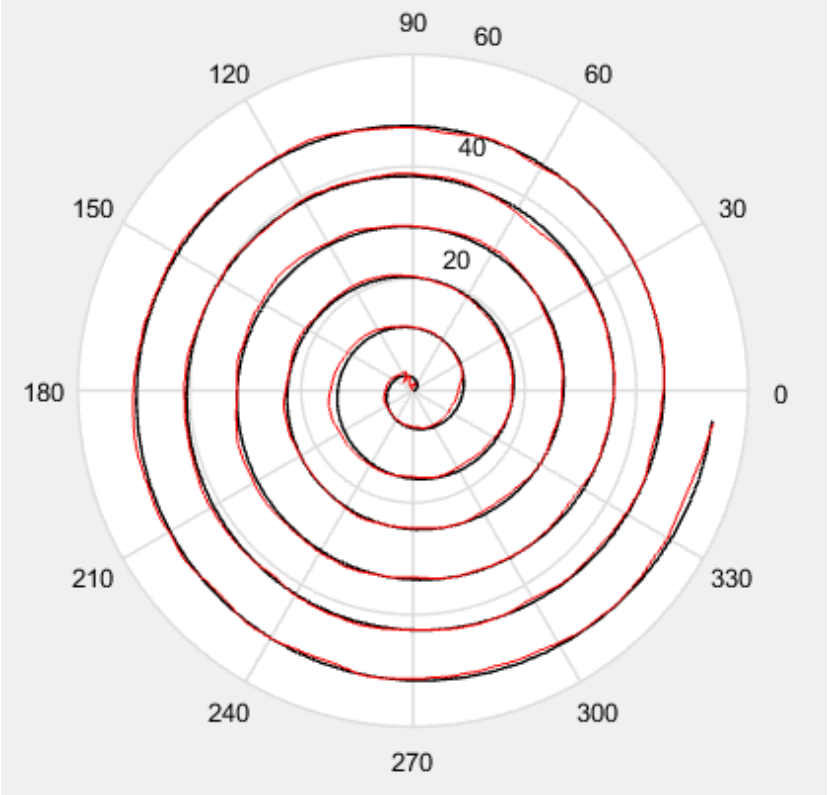
1.3 Методика вимірювання показів Архімедовою спіраллю на планшеті

Одним з відомих способів встановлення захворювання тремор з допомогою розпізнавання рисунку Архімедової спіралі, який виконується на планшеті з пером. Даний тип тестування є напівякісною оцінкою різних видів хвороби тремор, а полягає у виявленні відхилень малюнка хворого відносно взірцевої спіралі. Пацієнт розпочинає виконувати рисунок в середині Архімедової спіралі і старається найбільш точно повторити спіраль згідно шаблону.

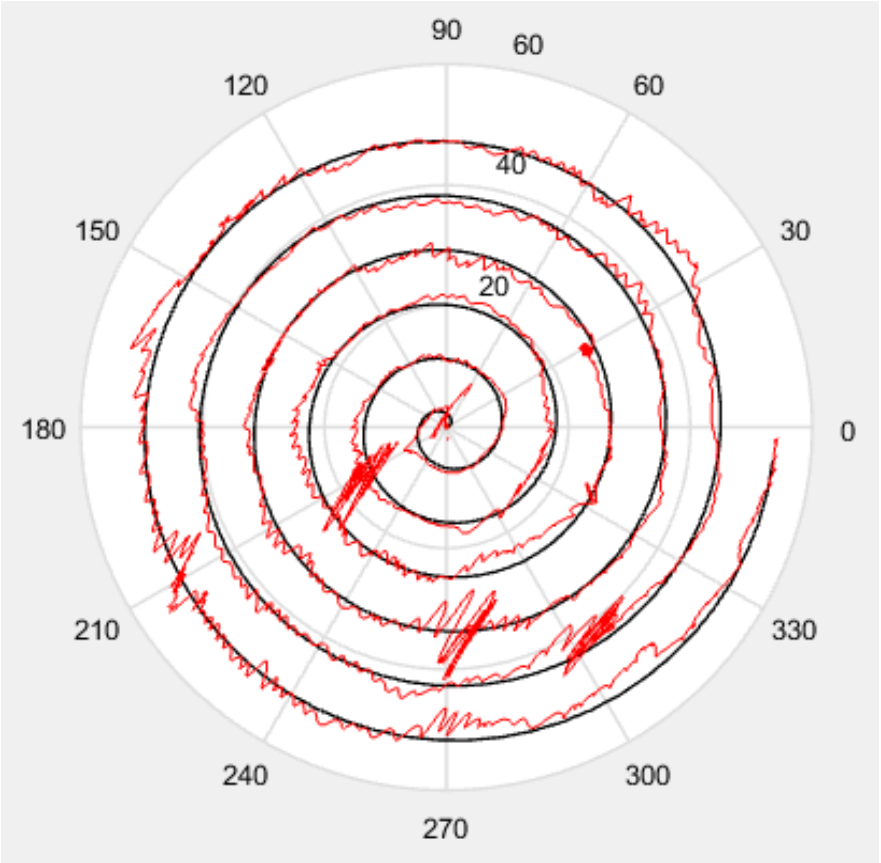
Процес проведення аналізу кривої, створеної хворим під час тестування спіральною моделлю, надає можливість отримати об'єктивні характеристики тремору. Такий тест дозволяє виявити непередбачувану активність м'язів та їх реакцію на рухи у різних площинах простору [4].

Збільшення найбільшого відхилення тремтіння з присутністю інерційної складової, частотні характеристики та залежність від інших ритмічних рухів можуть служити додатковими показниками для підтвердження діагнозу у підозрілих випадках. Крім того, спостерігається збільшення змінливості основних характеристик тремору протягом короткого періоду часу.

Частини тіла можуть коливатись нерегулярно або виникати у окремі моменти часу, а амплітуда цих коливань може часто змінюватися. На рисунку 1.2 наведено приклади результатів тесту спіралі від здорової людини та особи з проблемами тремтіння руки.



а)



б)

Рисунок 1.2 - Зразки виконання тесту здоровим (а) та хворим на тремор (б)

У рисунку 1.2 (а) показано випадок, коли здорова людина виконує тест, і намальована спіраль майже точно співпадає з шаблоном. Це означає, що тремору у цієї особи немає. Звісно, неможливо провести лінію точно по шаблону, але мінімальні відхилення є низькочастотними і свідчать про відсутність захворювання у пацієнта. У рисунку 1.2 (б) видно помітне відхилення спіралі пацієнта від шаблону. Це вже можна вважати наявністю деякого рівня тремору. На цьому прикладі можна побачити високу частоту відхилень - коливання відбуваються в середньому 6 чи 7 разів за одну секунду.

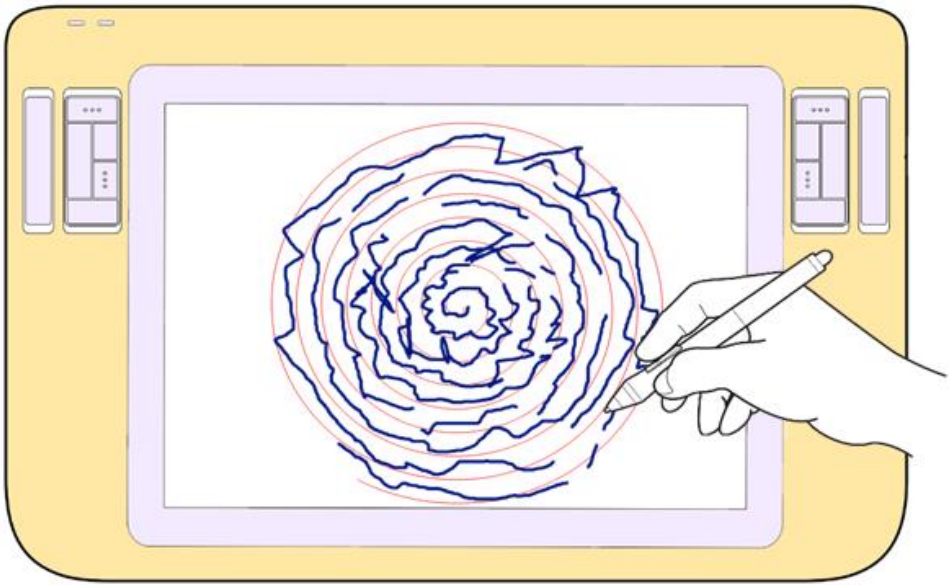


Рисунок 1.3 - Приклад прототипу пристрою

Шаблоном для цього тесту є Архімедова спіраль з шістьма обертами та міжпетлею шириною близько одного сантиметра. Цей шаблон відображається на екрані планшета, що дає можливість пацієнту виконувати рисунок за допомогою стилуса.

Створення сучасного ПЗ для автоматизації діагностування є актуальним для лабораторій і медзакладів, які намагаються вирішити проблеми неврології: захворювання Паркінсона, есенціальний чи фізіологічний тремор і інші.

При виконанні цього проєкту були використані методи математичного моделювання та автоматичної обробки даних Архімедової спіралі, встановлення параметрів даної системи, кількісних властивостей для оцінки есеніального тремору. Необхідним для досягнення цього є:

- вхідна статистична інформація;
- комп’ютерні системи;
- апаратне забезпечення для виконання тесту;
- апаратне забезпечення для діагностування хвороби.

Результати роботи створеного програмного забезпечення можна використати в якості об’єктивного джерела даних для встановлення ступеня захворювання.

2 ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1 Постановка завдання розробки

Кожна система це певний перетворювач, поведінку та властивості якого я будую в процесі створення програмної системи.

Програмна система — це певна «машина», яка вводиться у світ для того, щоб впливати на нього [6].

У цій роботі виконана розробка програмного забезпечення для методів і моделей прийняття певних рішень при невизначеності в процесі створення систем різноманітного призначення, що автоматизуються, теоретичних та прикладних засад, що стосуються побудови, а також впровадження інформаційних інтелектуальних технологій для створення сучасних систем аналізу інформації і системи управління.

Я розробляв інформаційні технології для обробки та синтезу інформаційних, структурних і функціональних моделей всіх об’єктів та процесів, що автоматизуються. Це потрібно для побудови та подальшого впровадження автоматизованих систем для технічної ідентифікації захворювання.

Моя мета - розробити програмне забезпечення для встановлення діагнозу есенціального тремору, за допомогою математичного моделювання та перетворень вхідних даних. За використання статистичної інформації і результатів попередніх експериментів створено метод для оцінки перебігу хвороби у пацієнтів.

Внаслідок проведеного мною аналізу було вирішено, що розроблюване рішення має виглядати як бібліотека, що містить програмні засоби для майбутнього впровадження даних класів моделей і алгоритмів до комплексної системи встановлення діагнозів. Ще одним завданням моєї розробки посталено розробити механізм тестування, а згодом і відлагодження методики проведення аналізу отриманих даних зі спіралі. В результаті виконання мого останнього

завдання може бути створена певна комп’ютеризована модель, виконана в пакеті розробки MatLab.

Для виконання поставлених завдань, програмна бібліотека має отримувати дані і параметри для обробки, коригувати параметри математичних перетворень, розгортати спіралі у форму кривих, визначати залишкове відхилення коливального сигналу, визначати амплітуду відхилення, обчислювати оцінку на основі амплітудних характеристик, визначати частотні характеристики за допомогою перетворення Фур'є, обчислювати оцінку тремору на основі частотної характеристики, а також надавати користувачеві всі методи та модливі програмні засоби для отримання результатів обчислень з функцією візуалізації їх на графіках.

2.2 Визначення актантів та варіантів використання

Після аналізу переліку завдань, було встановлено основних актантів системи та їх варіанти використання. Виходячи з вимог, ця робота передбачає розробку бібліотеки для використання у прикладному програмному забезпеченні з метою отримання результатів обчислень на основі певної моделі. Система орієнтована на конкретну аудиторію, що займається специфічними розробками. Таким чином, основним актором системи буде медичний працівник, який виконуватиме всі можливі завдання, що відповідають варіантам використання даної програмної бібліотеки.

Єдиним користувачем цієї розробки буде медичний чи науковий працівник, який матиме повний доступ до всіх функцій, а також можливостей системи, а також буде встановлювати попередні та плановані параметри моделювання.

Дані варіанти використання представлені на рисунку 2.1:

Рисунок 2.1 - Діаграма варіантів використання системи для актанта «Мед.працівник»

Користувач системи має наступні варіанти використання: вибір даних для аналізу, зміну параметрів обчислення математичної моделі, експорт поточних параметрів моделювання, імпорт раніше збережених результатів моделювання, визначення параметрів представлення для порівняння моделей (наприклад, обмеження на дані або циклічні обчислення).

Серед найважливіших варіантів використання системи можна виділити можливість отримання наступних результатів обчислень: спіралі Архімеда, спіралі пацієнта, розгортання спіралі в криву, амплітудної характеристики спіралі, частотної характеристики спіралі, оцінки ступеня тремору URS (заснована на статичному перетворенні амплітудної характеристики) і оцінки ступеня тремору URD (заснована на динамічній характеристиці системи спіралі, тобто частотній).

Такі можливості системи спрямовані на надання користувачу необхідних результатів перетворень та оцінок, що дозволяють аналізувати тремор і встановлювати ступінь його прояву.

У системі доступні декілька методів представлення результатів обчислень, а також можливість змінювати коефіцієнти моделі. Це дозволяє порівнювати різні варіанти і експериментувати з параметрами комп'ютерної моделі, щоб змінювати характеристики і поведінку системи. Основні параметри, які відповідають значенням математичної моделі, можна змінювати відповідно до розширення діаграми для варіанту використання "Виконати корекцію параметрів дослідження". Це дозволяє користувачу впливати на модель і налаштовувати її параметри залежно від потреб та вимог дослідження.

Кожен варіант використання, зображений на рисунку 2.1 відповідає поставленим завданням на етапі проектування рішення для виконання аналізу даних і отримання оцінки тремору.

Можливість використання даних в розроблюваній програмній бібліотеці здійснюється шляхом її інтеграції в комплексну систему діагностики. Це включає створення об'єктів і виклик методів, які реалізовані в класах, інтегрованих у бібліотеку. Таким чином, дані варіанти використання можуть бути передані до бібліотеки та оброблені з використанням доступних функцій та алгоритмів для отримання результатів діагностики.

2.3 Вибір архітектури та розробка програмного продукту

Управління розробкою будь-якого ПЗ відрізняється винятковою складністю і неможливістю передбачити кінцевий результат. Існує дуже багато різних підходів і методів для розробки, в яких передбачається застосування різноманітних видів моделювання даного процесу.

Для створення цього проекту я обрав еволюційну модель, яка є одним з підходів до управління процесом розробки програмного забезпечення. Ця модель базується на ідеї поступового ітеративного вдосконалення програмного продукту з часом. Основні принципи еволюційної моделі включають:

- Поступові зміни - розробка програмного забезпечення відбувається шляхом послідовного впровадження нових функцій та вдосконалення вже існуючих. Кожна ітерація розширює функціонал програми, при цьому при ній враховуються змінені вимоги та відгуки від користувачів.
- Ітеративність - розробка відбувається у вигляді серії ітерацій. Кожна ітерація має чітко визначені цілі, завдання та терміни виконання. Після завершення кожної ітерації отриманий функціонал може бути випробуваний та оцінений.
- Зацікавлені сторони - участь зацікавлених сторін, таких як замовник, користувачі та розробники, є важливим елементом еволюційної моделі. Вони співпрацюють на протязі всього процесу, надаючи фідбек та встановлюючи пріоритети.
- Перегляд та оцінка - кожна ітерація завершується оцінкою результатів та переглядом, щоб забезпечити відповідність вимогам та якості. На основі отриманої інформації можуть бути внесені зміни до подальших ітерацій[7].

Розширенням даної моделі є модель еволюційного прототипування, яка впроваджується протягом всього періоду розробки програмного забезпечення (зображена на рисунку 2.2). Ця модель часто відома як модель швидкої розробки програмного забезпечення (Rapid Application Development, RAD) [8].

У рамках даної моделі виокремлюються дві основні ітерації розробки: функціонального прототипування та проектування та реалізації системи. Основна ідея цієї моделі полягає в тому, що окремі функції системи моделюються у вигляді прототипів, а потім поступово розвиваються шляхом

поетапного вдосконалення, з метою виконання всіх заданих функціональних вимог.

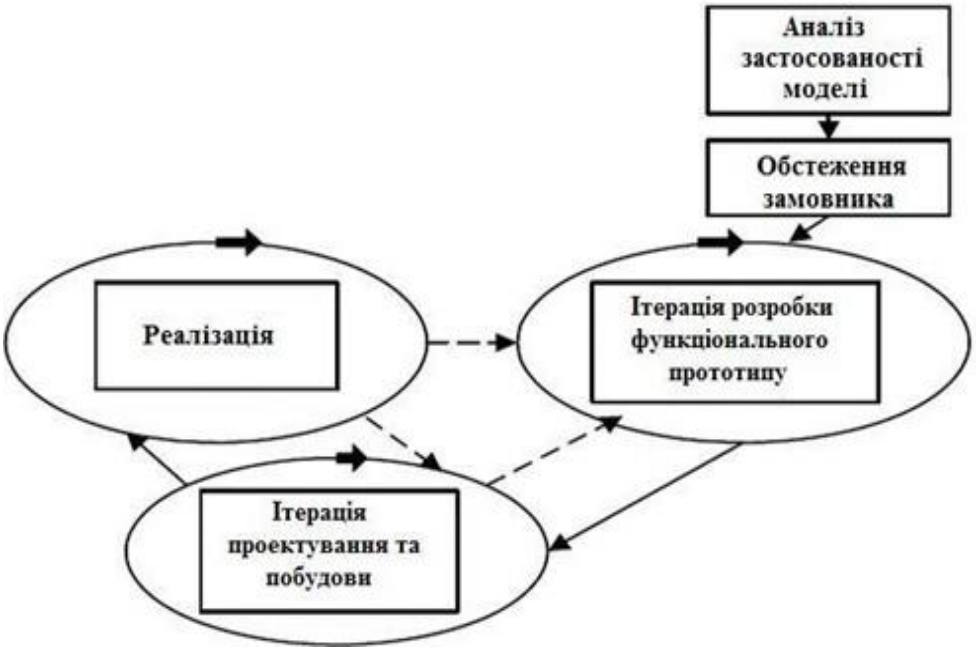


Рисунок 2.2 – Модель еволюційного прототипіювання

У моїй роботі використовується об'єктно-орієнтований підхід у програмуванні, що ґрунтується на об'єктно-орієнтованій архітектурі. Дана парадигма представляє програмну структуру в якості множини об’єктів, що об’єднуються в певні модулі. Таким чином, коли ми створюємо один модуль чи клас, чи можемо ми його використовувати повторно під час проектування та розробки будь-якої іншої системи або моделі, а також при представленні результатів у різних ситуаціях.

Архітектура модель нашої системи це монолітний застосунок, в якому є розділена частина алгоритмічного будування моделей та візуального представлення даних. Перша з цих частин є бібліотекою з наборами класів та їх методів для обробки інформації, а друга в свою чергу виводить інформацію користувачеві.

Складне та багатокомпонентне програмне забезпечення включає в себе декілька відносно самостійних у виконанні та функціональному наповненню

підсистем. Ці підсистеми можна представити у вигляді діаграми компонентів. Реалізація математичної складової - java-бібліотеки, є однією з таких підсистем.

Комплекс для встановлення діагнозу включає в себе модулі та підсистеми призначені для – авторизації, управління записами, виконання тестування спіраллю Архімеда, обробки вхідної інформації.

На рисунку 2.3 зображено схему взаємодії між цими компонентами:

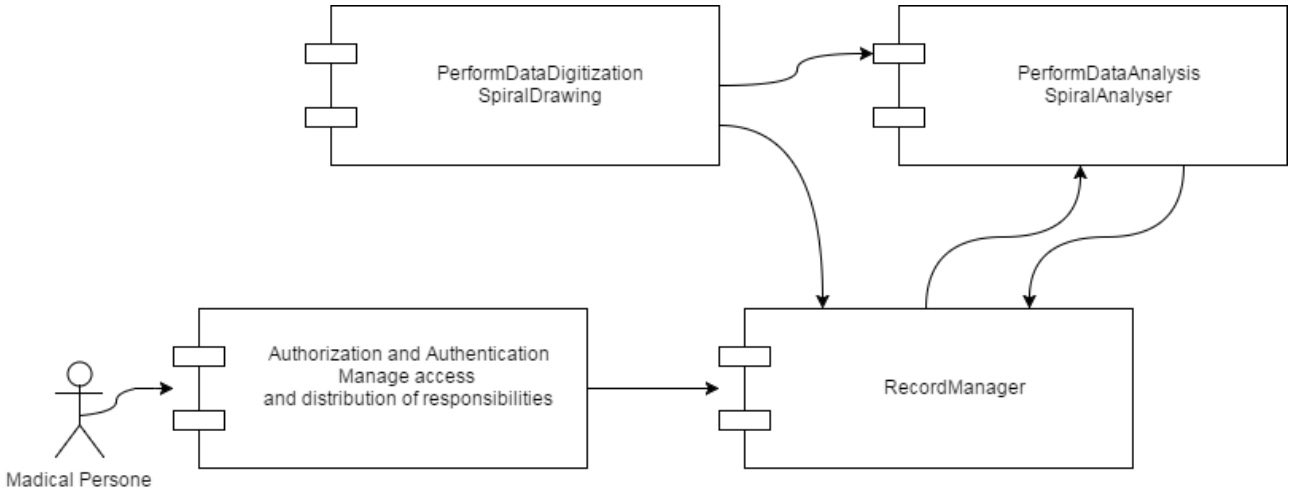


Рисунок 2.3 - Діаграма модулів комплексної системи встановлення діагнозу

Підсистему обробки інформації виглядає як один модуль алгоритмічної частини реалізації механізму моделювання. В ній є реалізації класів та методів взаємодії, які мають мету ініціалізації стартових параметрів, реалізації математичних моделей та алгоритмів обчислення на базі вхідних даних - результатів досліду, методів вводу та виводу атрибутів класів. В загальному всі модулі алгоритмічної частини скомпільовані та зібрані в бібліотеку java-програм. Наступна діаграма візуалізує процес обміну повідомленнями всередині програми під час виконання аналізу спіралі:

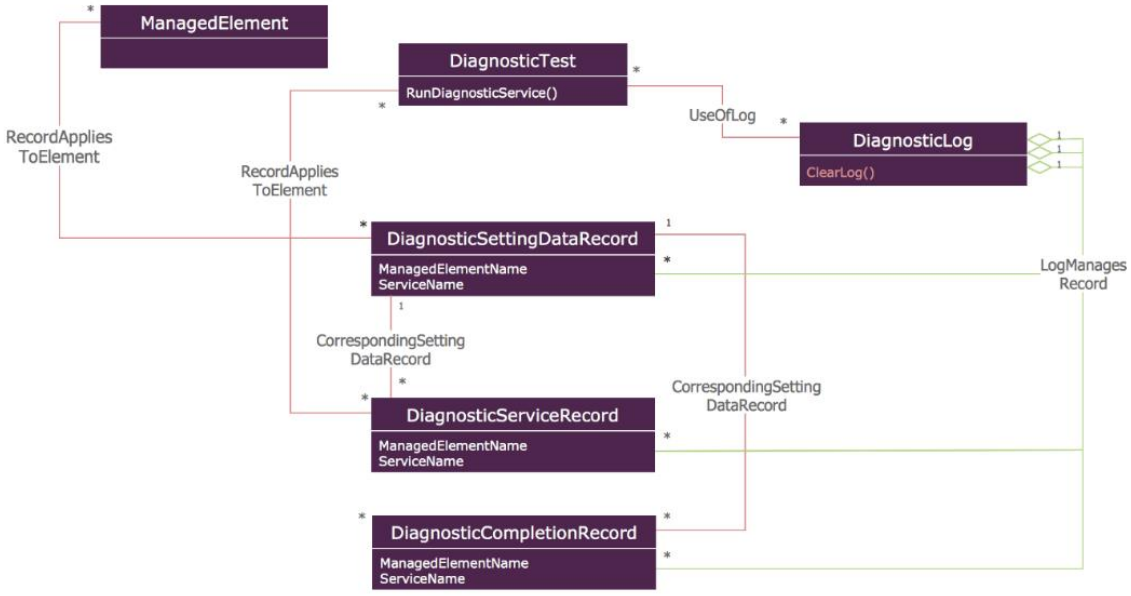


Рисунок 2.4 - Діаграма обміну повідомленнями між компонентами всередині бібліотеки автоматичної ідентифікації параметрів

Разом зі всіма засобами для розробки на мові програмування Java є бібліотеки вбудованих класів, що значно спрощують процес розроблення програми - віртуальна бібліотека протестованого коду з готовими рішеннями багатьох повсякденних задач. Тобто є можливість використовувати реалізований функціонал з бібліотек. Це економить час і кошти для реалізації програмного проекту, бо треба все робити «з нуля». Потрібно лише підключити готові пакети бібліотеки[10]. Використання модульного способу конструювання програмного забезпечення дає змогу гнучко та продуктивно розробляти програмні системи.

У цьому програмному продукті зпроектовано інтерфейси та класи що реалізують створені інтерфейси, класи що наслідують інші, їх методи , що виконують різні операції та обчислення. Такий підхід дає можливість уникнути будь-яких помилок чи суперечностей, надлишковості коду або його повторень, і загалом зробити його більш зрозумілим та доступним для читання, що означає легкість під час його написання та подальшої підтримки.

2.4 Абстрактна модель розроблюваного програмного забезпечення

Для розробки цієї системи мною було вибрано мову програмування Java. Проектую розроблювану програму згідно з об’єктно-орієнтованим підходом, правильним рішенням є проаналізувати основні сутності та класи в ній.

Виходячи з цього, було створено програмне забезпечення з використанням класів та їх наслідування, а також об’єктів та інтерфейсів. Способом для розробки доцільно вважати зворотню інженерію, що означає розробку та реалізацію всіх ключових елементів, що були перелічені вище та їх взаємодію між собою. Після чого , на основі вже частково реалізованого коду буде збудована UML-діаграма класів.

UML (Unified Modeling Language) — є стандартизованою мовою для моделювання програмного забезпечення, систем та бізнес-процесів. Вона надає нотацію та правила для створення графічних моделей, які допомагають розуміти, проектувати, впроваджувати та документувати системи [9].

Вона широко використовується за використання парадигми об'єктно-орієнтованого програмування та є майже обов’язковою частиною уніфікованого процесу розробки програмного забезпечення.

За допомогою UML розробники створюють модель структури та поведінки системи за допомогою різних графічних позначень для представлення загальної інформації про різні об’єкти системи і дає змогу сконцентруватися на саме архітектурі та проєктуванні.

Таким чином, архітектура системи передбачає використання класів, інтерфейсів та зв'язків між ними. Діаграма класів для модуля алгоритмів системи представлена на рисунку 2.5.

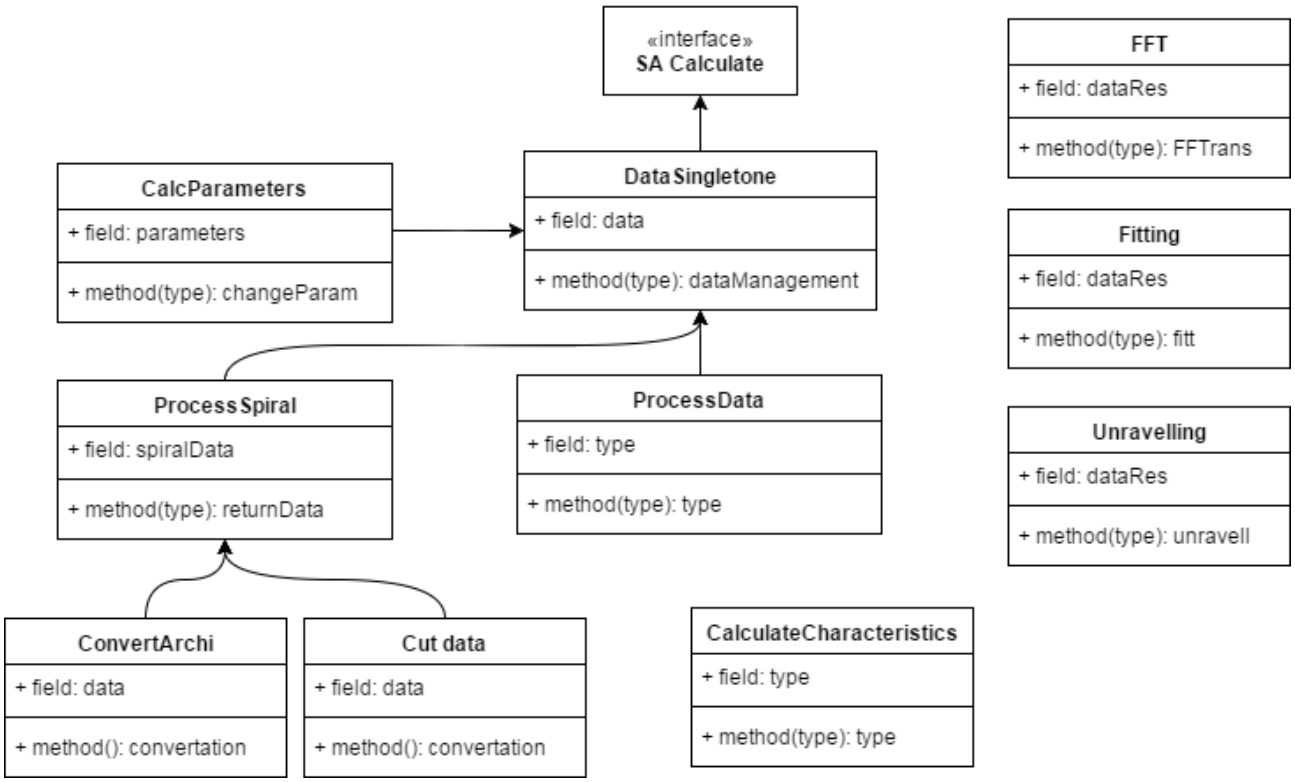


Рисунок 2.5 - UML-діаграма класів бібліотеки з алгоритмами математичних обчислень

Згідно рисунку 2.5, імплементація алгоритму обчислення включає в себе основний абстрактний клас SA Calculate, який в подальшому використовується для створення відповідної моделі, яка представляє дані. Це дозволяє використовувати алгоритм з мінімальними зусиллями, залежно від потреб обчислень, не глибоко занурюючись у структуру бібліотеки та її класів. Кінцевому користувачеві важливий лише набір методів роботи і можливість їх швидкого та простого виклику у власній програмі. Для досягнення цього використовуються інтерфейсні класи. Вони забезпечують можливість взаємодії в середині внутрішніх механізмів, що здійснюють обчислення.

Клас Data Singleton містить масив потрібних змінних і масивів обрахованої інформації, які доступні для всіх класів та методів програмного модуля. Він є статичним та глобально доступним класом, який містить методи, що виконують функцію передачі чи отримання значень і параметрів атрибутів. Завдяки такій архітектурі класу можна доволі ефективно зберігати загальнодоступні змінні.

Виконання математичних перетворень здійснюється у класах FFT, Fitting та Unravelling. Всі ці класи створюють об'єкт з властивими йому параметрами та атрибутами математичної моделі.. Методи в цих класах містять реалізації алгоритмів обчислення перетворень, повертаючи значення отриманих результатів. Виклики та маніпуляції з об’єктами та перетвореннями їх атрибутів відбуваються в класі ProcessData.

Клас CalcParameters виконує функцію конфігураційного механізму для ініціалізації та зберігання початкових змінних, та змінних середовищ системи.

Клас Process Spiral містить набір методів та операцій для обробки даних спіралей, що надходять з зовнішнього середовища. В цьому класі реалізовано механізм перетворення координат спіралі з декартових в полярні, обмеження вибірки даних, а також маніпуляції з вхідною інформацією.

Для найпростішого відтворення процесу аналізу даних можна скористатися UML-діаграмою активності. Ця діаграма крок за кром показує сценарій виконання процесу аналізування даних, починаючи з отримання даних на вході у систему і закінчуючи отриманням кількісних характеристик і оцінок ступені перебігу захворювання тремор.

На діаграмі активності присутні стартова позиція. З неї починається процес виконання поставлених завдань.

Діяльність передбачає завершення у двох випадках: при успішному виконанні завдань, що були поставлені, або у випадку невдачі при виконанні на важливому кроці.

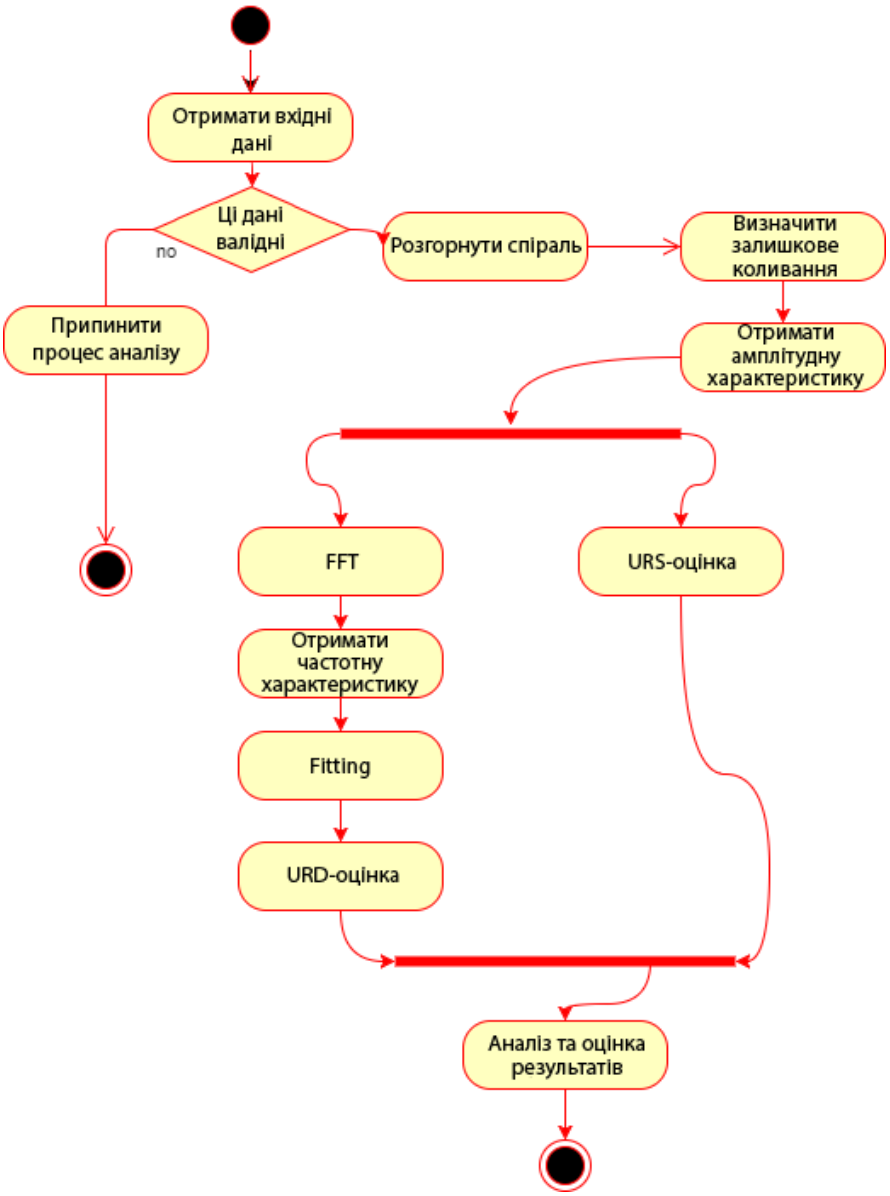


Рисунок 2.6 - Діаграма активності для аналізу даних спіралі

На етапі початку роботи системи, вхідними даними є дані Архімедової спіралі та дані про рисунки хворих.

Наступними кроками виконання завдання аналізу даних та ідентифікації ступеню тремору є: розгортання спіралей, виокремлення залишкового коливального сигналу, обчислення амплітуди відхилення, отримати URS-оцінку на основі характеристики амплітуди, виконання FFT з даними амплітудної характеристики, отримання частотної характеристики, виконання Gaussian-fitting, визначення результуючої оцінки тремору пацієнта.

3 РОЗРОБКА ПРОГРАМНО-АПАРТНОГО КОМПЛЕКСУ

3.1 Конструювання і розробка системи

Вибір мови та середовища розробки програмного забезпечення є важливою рішенням, яке може вплинути на увесь життєвий цикл проекту. Мова програмування визначає синтаксис та можливості програми, тоді як середовище розробки надає інструменти для написання, тестування та налагодження коду.

Вибір правильної мови та середовища залежить від багатьох факторів, таких як тип проекту, потреби бізнесу, навички команди розробників та екосистема підтримки. Правильний вибір може сприяти продуктивності, якості програмного продукту та забезпечити швидкість розробки.

Мова програмування повинна відповідати потребам проекту, мати підтримку необхідних функцій та бібліотек, а також мати активну спільноту розробників. Середовище розробки повинно надавати зручний інтерфейс, функції автодоповнення, налагодження та підтримку контролю версій.

Невдалі вибори можуть призвести до низької продуктивності, високих витрат часу та ресурсів на розробку, складнощів у розумінні коду та підтримці програмного продукту. Тому, вибір мови та середовища розробки є важливим кроком для успішного виконання проекту.

Для розробки програмного забезпечення в межах обраної нами архітектури підходять мови з підтримкою парадигми ООП - C#, Java, Python, C++. Варто зазначити увагу на наявності засобів для створення потрібних методів і можливості кожної з цих мов, наявність бібліотек та підтримка запуску на різних платформах. Враховуючи те, що бібліотека буде використовуватись у різних середовищах та повинна працювати на більшості пристроїв, було обрано мову програмування Java, через її універсальність та кроссплатформенність.

Існує багато різних інтернет-ресурсів з наведеними прикладами програмного коду, бібліотеками, що містять готові функції, що є надзвичайно корисним в процесі написання програмного коду. Окрім цього Java -

стандартизована і доступна мова програмування. Для запуску в різних операційних системах програм, які написані на Java, потрібно лише встановити JRE (Java Runtime Environment), яке є безкоштовним середовищем та доступним для на офіційному сайті Oracle.

Тому, серед всіх мов програмування, для розробки було обрано саме Java. Головною її особливістю є використання байт-коду, який виконується віртуальною машиною Java (JVM – Java Virtual Machine). Вона входить в вищезгаданий пакет JRE, що забезпечує функціонування написаних цією мовою програм всередині багатьох різних операційних системах [10].

Серед переваг мови програмування Java є:

- Переносимість: Java є платформонезалежною мовою, що означає, що програми, написані на Java, можуть працювати на різних операційних системах без необхідності в перекомпіляції. Це дозволяє розробникам створювати одну версію програми, яка може бути виконана на різних пристроях.
- Об'єктно-орієнтоване програмування: ця мова підтримує об'єктно-орієнтоване програмування, що сприяє модульності, перевикористанню коду та забезпечує зручний підхід до розвитку програмного забезпечення.
- Багата бібліотека: Java має велику стандартну бібліотеку, яка надає різноманітні функції для розробки програмного забезпечення. Це включає бібліотеки для роботи з мережами, базами даних, графікою, шифруванням, паралельним програмуванням та багато іншого. Вона дозволяє розробникам ефективно виконувати завдання без необхідності писати все з нуля.
- Безпека: Java має вбудовану систему безпеки, яка дозволяє запобігати небезпечним операціям, таким як доступ до пам'яті, переповнення буфера та іншим вразливостям. Це робить Java популярним вибором для розробки безпечних програм та веб-додатків.

- Підтримка багатопотоковості: Java надає вбудовану підтримку для роботи з потоками, що дозволяє створювати багатопотокові програми. Це дозволяє ефективно використовувати ресурси пристрою та забезпечує більш високу продуктивність.
- Велика спільнота розробників: Java має велику та активну спільноту розробників, яка надає підтримку, ресурси та відповіді на питання. Це сприяє швидкому вирішенню проблем та обміну знаннями.

Загалом, Java є потужною та надійною мовою програмування, яка використовується для розробки різноманітних програмних продуктів, від мобільних додатків до корпоративних систем. Її переносимість, об'єктно-орієнтований підхід та багата бібліотека роблять її привабливим вибором для багатьох розробників.

Ключовим принципом розробки програмного забезпечення на цій мові є захист від будь-якого стороннього доступу. Програми , що написані мовою Java не можуть здійснювати виклик глобальних функцій чи одержувати доступ до всіх ресурсів системи, що створює в обраній мові програмування дуже високий рівень безпеки, який є недоступним для інших [5].

Вирішено обрати середовище програмування NetBeans. Яке має високу гнучкість налаштувань, можливість розширення, простий інтерфейс та доступність користувачеві. NetBeans є безкоштовним для завантаження та користування.

Щоб виконати комп’ютерне моделювання реалізованого програмного забезпечення, використовуючи спроектовану технологію встановлення діагнозу, потрібно використати один із пакетів для виконання чисельних обчислень і візуального відтворення результатів цих обчислень. У випадку модельованого програмного забезпечення, буде ефективним та зручним застосування пакету прикладних програм MatLab.

MatLab є мовою програмування і інтерактивним середовищем, яке надає інструментарій для аналізу даних, розробки алгоритмів, створення моделей і додатків [11]. Використовуючи MatLab, можна проводити дослідження різних підходів, отримувати рішення і розв'язки для складних задач.. Також в ньому є технологія компілювання та збирання готових програмних рішень.

До складу пакету входять багато функцій, які призначені для побудови двовимірних та тривимірних графіків та візуального аналізу інформації. Вбудоване середовище розробки надає можливість створювати графічні інтерфейси з різноманітними елементами керування - кнопками, полями для введення та багатьма іншими. Дані графічні інтерфейси можуть бути перетворені в незалежні додатки, використовуючи компонент MatLab-Compiler.

3.2 Реалізація Java-бібліотеки

У даній частині роботи варто зосередити увагу на практичній реалізації класів та методів, описаних на діаграмах класів. Основні методи для обчислення та візуалізації надалі будуть наведені.

Алгоритми обчислення математичних методів реалізовані у вигляді набору класів, що містять методи, які моделюють потрібну функціональність. Створені класи та взаємодію їх між собою було об’єднано єдиний модуль-бібліотеку.

Метод побудови шаблону Архімедової спіралі в Декартовій системі координат реалізовано методом makeArchimedesSpiral() класу ProcessSpiral.

```
private ArrayList<Point.Double> makeArchimedesSpiral() {
    ArrayList<Point.Double> spiral = new ArrayList<Point.Double>();
    double x, y, r;
    for (double a = 0; a < 12 * Math.PI; a += 0.1) {
        r = a * 9 / (2 * Math.PI); // 9 mm between cicle
        x = r * Math.cos(a);
        y = -r * Math.sin(a);
        Point.Double point = new Point.Double(x, y);
        spiral.add(point);
    }
    return spiral;
}
```

Рисунок 3.1 – реалізація методу побудови спіралі Архімеда

Напевно найбільш цінними та важливими методами цієї розробки є такі з них, що обчислюють амплітудні та частотні характеристики. Далі приведено лістинг одного алгоритму для обрахунку параметрів FFT-перетворення:

```
realOut[yWave][xWave]+=(
    inputData[ySpace][xSpace] * Math.cos(2 * Math.PI * (
        (1.0 * xWave * xSpace / width) + (1.0 * yWave * ySpace / height))
    )
) / Math.sqrt(width * height);

imagOut[yWave][xWave] -= (
    inputData[ySpace][xSpace] * Math.sin(2 * Math.PI * (
        (1.0 * xWave * xSpace / width) + (1.0 * yWave * ySpace / height))
    )
) / Math.sqrt(width * height);

amplitudeOut[yWave][xWave] = Math.sqrt(
    realOut[yWave][xWave * realOut[yWave][xWave] + (
        imagOut[yWave][xWave] * imagOut[yWave][xWave]
    )
);

return imagOut[yWave][xWave];
```

Рисунок 3.2 – обчислення параметрів FFT-перетворення

Дана функція повертає змінну imageOut, що включає в себе відношення амплітуди тремору та частотної характеристики.

Перетворення Архімедової спіралі з Декартової системи координат у полярну виконується перетворенням, описаним на рисунку 3.3. У ньому як параметр використовуються координати зафіксованих даних з цифрового

дігітайзера у вигляді координат декартової системи. Але для виконання будь-яких наступних дій над даними спіралі, потрібно представити їх у системі полярних координатах.

```
public static Point getCartesianPoint(double x, double y) {
    Point p = new Point();
    p.x = x;
    p.y = y;
    p.radius = sqrt(x * x + y * y);
    p.angleR = atan2(x, y);
    p.angleD = toDegrees(p.getAngleRadians());
    return p;
}
```

Рисунок 3.3 – обчислення полярних координат моделі спіралі

Показник стандартного квадратичного відхилення (STD), який також використовується для обчислень, реалізовано в коді на рисунку 3.4:

```
//std dev function for good measure
public static double getStandardDeviation(double[] data) {
            final double mean = getMean(data);
            double sum = 0;
            for (int index = 0; index != data.length; ++index) {
                sum += Math.pow(Math.abs(mean - data[index]), 2);
            }
            return Math.sqrt(sum / data.length);
        }
        public static double getMean(double[] data) {
    if (data.length == 0) {
            return 0;
        }
        double sum = 0.0;
        for (int index = 0; index != data.length; ++index) {
            sum += data[index];
        }
        return sum / data.length;
```

Рисунок 3.4 – обчислення полярних координат моделі спіралі

Для проведення математичних обчислень можуть бути використані функціональні можливості, що вбудовані у бібліотеку java. Math, що поставляється разом з пакетом розробки. Вона містить набір готових методів, функцій, сталих та констант.

3.3 Реалізація комп'ютерної моделі

В даній розробці реалізовано простий інтерфейс користувача для взаємодії з ним та візуалізації опрацьовуваної інформації у пакеті MatLab-guide.

Завантажити вхідні дані Архімедових спіралей та рисунків хворого даний програмний код дозволяє по виклику застосунку запустити діалогове вікно додатку та обрати текстовий файл, що має розширення .txt з файлової системи.

```
function pushbutton_openfile_Callback(hObject, eventdata, handles)
global data;
%Open new file with record
[FileName,PathName]=uigetfile({'*.txt','All.txt files';'*.*','All Files' },'mytitle');
if isequal(FileName,0) || isequal(PathName,0)
        disp('User pressed cancel')
else
disp(['User selected ', fullfile(PathName, FileName)])
    data = dlmread(fullfile(PathName, FileName),'',1,0);
    set(handles.textLabelOfFile, 'String', FileName);
    pushbutton_compute_Callback(handles.pushbutton_compute, eventdata, handles);
end
```

Рисунок 3.5 – виклик функції для завантаження інформації з файлу

У коді, приведеному далі описана можливість змінних отримати значення з масиву, що містить вхідні дані. Тут застосовуються дані шаблона Архімедової спіралі, і в змінні X і Y присвоюються значення з відповідних масивів інформації. Користуючись цим, виконується перетворення з метою одержання моделі Архімедової спіралі в полярних координатах.

```
Archi = dlmread('Archi.txt',' ',1,0);
        X=Archi(:,1);
        Y=Archi(:,2);
        Theta=atan2(Y,X);
        Rho=sqrt(X.^2+Y.^2);
        S=cat(2,unwrap(Theta),Rho);
        % take care that angle is in radian
```

Рисунок 3.6 – ініціалізація змінних в досліджуваному середовищі

Здатність побудувати графіки функцій і вивести візуальне відображення масивів інформації реалізовані функцією plot(). ЇЇ виклик наведений в коді, що зображений на рисунку 3.7.

```
global f Y NFFT D;
figure
plot(f,2*abs(Y(1:NFFT/2+1)))
title('FFT(Residual Fluctuations) & URD determination')
xlim([0 15]);
xlabel('Frequency (Hz)')
ylabel('Y(f)')
```

Рисунок 3.7 – код побудови графіку функції URD

Тут приведено взірець побудови функції, що призначена для одержання частотної характеристики за використання перетворення Фур'є. Методи `title(),xlabel(), ylabel(), xlim()` призначені для форматування виведення інформації на графіки та створення підписів для осей та позначок шкали.

Наступним кроком у здійсненні аналізу є змінення параметрів перетворення за допомогою фільтра Гауса. Спосіб одержання значень і їх перетворення до необхідного виду зображено в коді на рисунку 3.8.

```
% Collect fitting Upper and Lower limits parameters from edit_text_boxes
fitt_Low_a=str2double(get(handles.edit_Lower_fitting_a,'String'));
fitt_Low_w=str2double(get(handles.edit_Lower_fitting_w,'String'));
fitt_Low_Xc=str2double(get(handles.edit_Lower_fitting_Xc,'String'));
fitt_Up_a=str2double(get(handles.edit_Upper_fitting_a,'String'));
fitt_Up_w=str2double(get(handles.edit_Upper_fitting_w,'String'));
fitt_Up_Xc=str2double(get(handles.edit_Upper_fitting_Xc,'String'));
cut_loop_points=str2double(get(handles.edit_cut_loop_points,'String'));
```

Рисунок 3.8 – зразок зчитування нових параметрів системи

Цей спосіб для одержання значення в текстових полях інтерфейсу користувача вмикає певний параметр доступу до елемента вікна

`get(handles.edit_`. Так збираються дані з поля та присвоюються в змінну з використанням перетворення `=str2double`.

Також тут є можливість статистичного оцінювання ступеню захворювання за допомогою перетворення характеристик в шкалу Фан-Толоза-Марін. Наведений далі код програми виконує математичне перетворення щоб привести кількісну частотну характеристики URD до значення згідно шкали FTM:

```
% in mm
% Score using the Rivals's scale range: (0-10)
URD_Score=9.20773+3.7515*log10(URD);
URD_Score_moy = mean(URD_Score);
URD_mm = mean(URD);
```

Рисунок 3.9 – зразок зчитування нових параметрів системи

Найбільш комплексною функцією є `function calculation(Xi, Yi, T)`. В ній відбувають всі перетворення інформації, результати цих перетворень присвоюються в глобальні змінні, а потім вони будуть використані для побудови графіків або для виводу результату оцінки ідентифікації ступеню досліджуваного захворювання. Програмний код описаної функції є об’ємним, в зв’язку з чим він наведений в додатку А.

Щоб створити механізм для оновлення обчислюваних даних після зміни будь-якого з параметрів, створено кнопку Compute, після натискання на яку або виклику відповідного методу всередині програмного коду, буде виконане повторне обчислення та ініціалізація змінних в системі.

3.4 Тестування розробленої системи

Як відомо, немає програм без помилок. Програмне забезпечення необхідно перевірити на відповідність критеріям розробки та вірність реалізованих алгоритмів.

Спочатку програма перевіряється на помилки. Тобто при неправильному введенні параметра дані не вводяться. Так було випробувано внесення невірних вхідних даних при використанні. У результаті в журналі виникає помилка java.lang.NumberFormatException, коли дані вводяться з невідповідним значенням формату представлення даних. Тобто ми знаємо причину помилки, але поки що система не відповіла користувачеві з інформаційної точки зору. Тому доцільно розглянути превентивний підхід до обробки таких подій, а також більш конкретно повідомити користувача про причину помилки.

Автоматичне функціональне тестування перевіряє правильність роботи програмного коду. За потреби розробники можуть виконувати ці тести час від часу. У комп'ютерному програмуванні модульне тестування програмного забезпечення використовується для перевірки одного або кількох програмних модулів, методів класу, щоб визначити, чи вся бізнес-логіка правильна та відповідає очікуваним цілям і результатам.

JUnit — це бібліотека тестування програмного забезпечення для мови Java. [27]. Досвід, отриманий під час використання JUnit, важливий для розробки концепцій тестування програмного забезпечення.

```
public class SimplyMeasuredSDTTest {
private static final Logger LOGGER = Logger
    .getLogger(SimplyMeasuredCloudTest.class);
    @Test
public void simplymeasured() throws IOException {
final ProcessData frequencyAnalyzer = new ProcessData ();
frequencyAnalyzer.setFrequenciesToReturn(1000);
frequencyAnalyzer.setMinTLength(3);
frequencyAnalyzer.setStopT(loadStopT());
assertNotNull(frequencyAnalyzer.getParamT());
actual = frequencyAnalyzer.performCalc(T);
assertThat(actual,is(expected));
String shouldNotExist = configurable.getShouldNotExist();
assertNull(shouldNotExist);
}
```

Рисунок 3.10 – модульний тест методу fAnalyse() класу ProcessData

3.5. Демонстрація використання створеного програмного продукту

Внаслідок виконання даної роботи було реалізовано інтерфейс користувача і методи взаємодії та демонстрації роботи програмного забезпечення для побудови моделей.

Було задіяно програмний пакет MatLab з метою побудови, компіляції та перевірки на помилки застосунку SpiralAnalyser. Щоб запустити програми треба перейти до директорії, що містить проект і виконати .exe файл. Після цього появляється відкрите стартове вікно програми SpiralAnalyser.

Як можна спостерігати на рисунку 3.11, вікно цієї програми розділено на різні функціональні елементи. При натисканні на програмну кнопку 1 користувачу надається доступ до нового діалогового вікна, де можна вибрати один із попередньо збережених діагностичних тестів спіралей. Після підтвердження вибору, в самій програмі автоматично промальовуються шаблонна Архімедова спіраль та спіраль рисунку хворого.

Рисунок 3.11 - Вікно програми SpiralAnalyser з відритим записом спіралі

1 – кнопка вибору вхідних даних для тесту;

2 – шаблон спіралі Архімеда та спіралі пацієнта ;

3 – оцінка тремору за амплітудною характеристикою;

4 – оцінка тремору за частотною характеристикою;

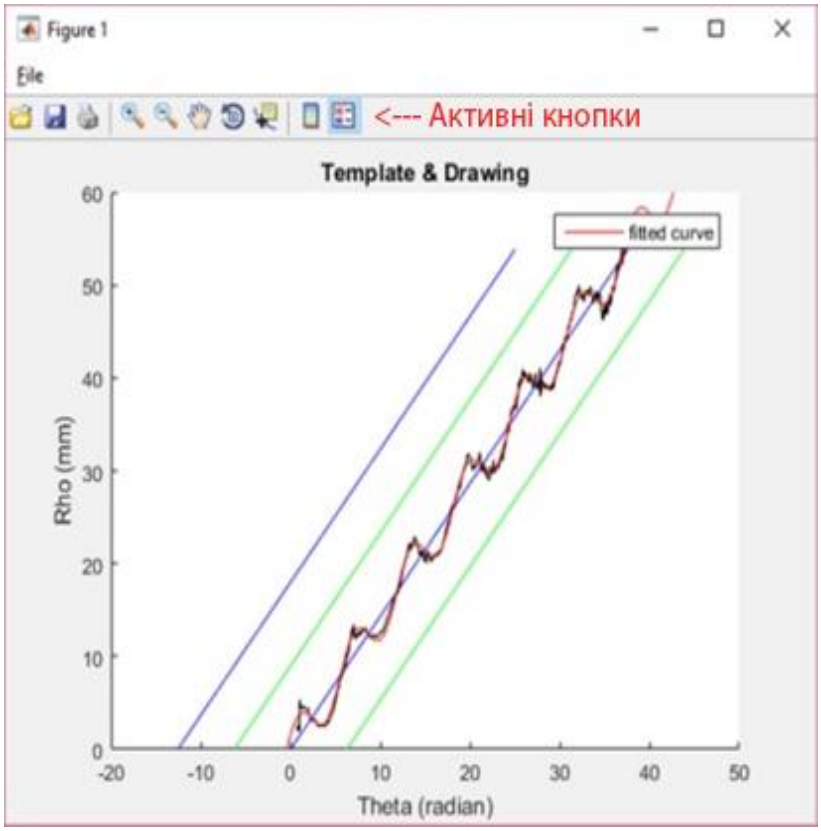
5 – кнопки для виводу вікон графіків обчислюваних характеристик;

6 – панель для зміни параметрів обмеження вибору значень;

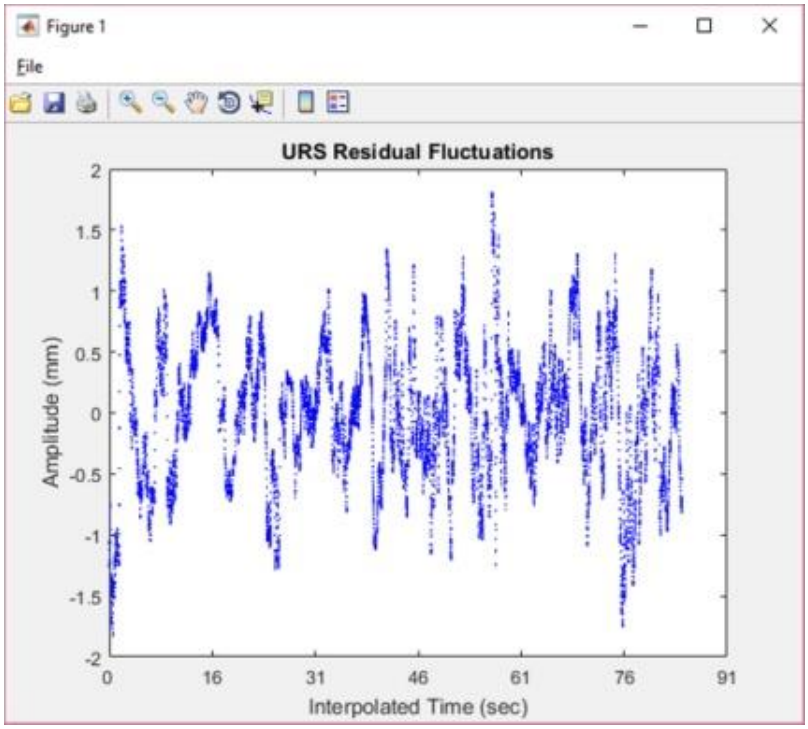
7 – панель зміни параметрів функції Гауса.

Автоматично у головному вікні програми висвітлюється інформація аналізу вхідних даних спіралі, яку намалював пацієнт (поля 3 і 4), а саме: URS – статична оцінка хвороби на основі відхилень залишкового колихання, URD – значення динаміки руху, характеристика частоти, створена перетворенням Фур’є. Точніші і достовірніші дані та оцінки ступеню тремору хворого можна

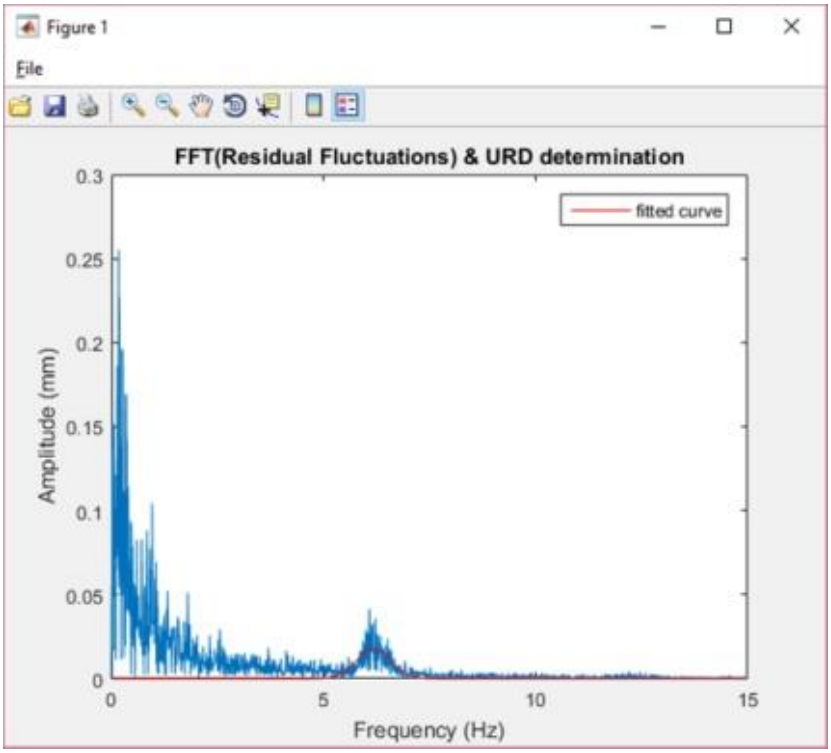
отримати, змінюючи налаштування параметрів обчислення, відкидаючи хибні значення початку та кінця спіралі. Можна внести зміни в функціоналі в полях 6 та 7 основного меню.



a) Unravelling



б) URS Fluctuations



в) URD Fitting

Рисунок 3.12 - Візуалізація різних математичних перетворень спіралі

При виборі однієї з кнопок поля 5, що зображено на рисунку 3.12, користувач має можливість відобразити окремий етап перетворення даних та побудуви результатів і порівнювати їх. Три перші кнопки поля 5 демонструють наступні можливості користувача:

- користувач може візуалізувати процес розгортання спіралі в криву з використанням методики unravelling;

- користувач може отримати візуальне представлення амплітуди залишкового відхилення на всій аналізованій часовій осі;

- користувач може переглянути графічну частотну характеристику динамічного значення амплітуди відхилення, яка формується за допомогою перетворення Фур'є.

Щоб здійсненити точне налаштування системи дослідження внесено ряд полів з конкретними парамерами, що відповідають змінним у функціях

перевтворення Гауса і кількості паналізованих даних. Можна виконувати аналіз даних спіралі, з покроковим зменшенням необхідної кількості часових міток з обох кінців. Існує ручний режим налаштування виключення з аналізованого спектру точок як з початку масиву даних, так і з кінця:

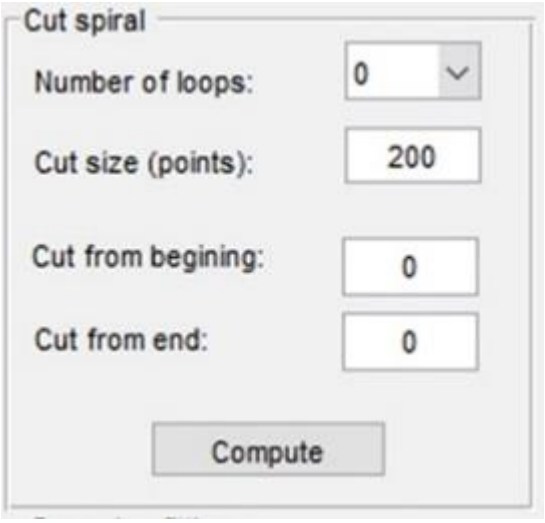


Рисунок 3.13 - Можливість змінювати параметри кількості аналізованих даних

Викидаючи з масиву потрібних для проведення аналізу даних по 200 точок з кожного кінця, можна провести порівняння отриманих результатів, а також їх зміну як реакцію на фільтрацію всіх вхідних даних.

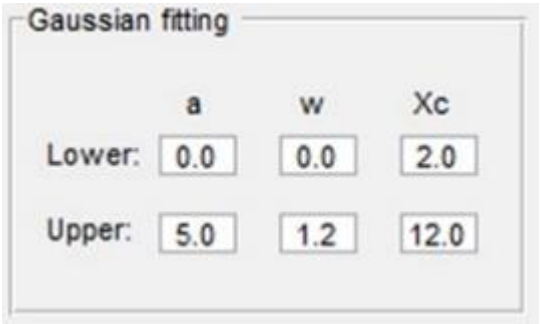
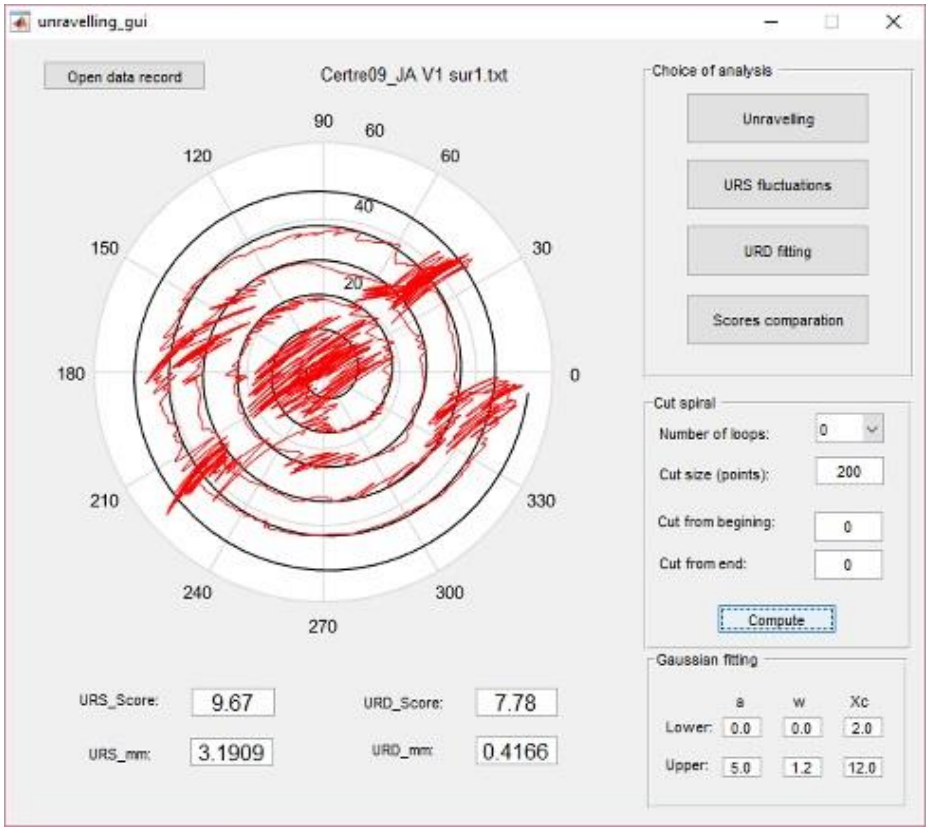


Рисунок 3.14 - Налаштування перетворення методом Гауса

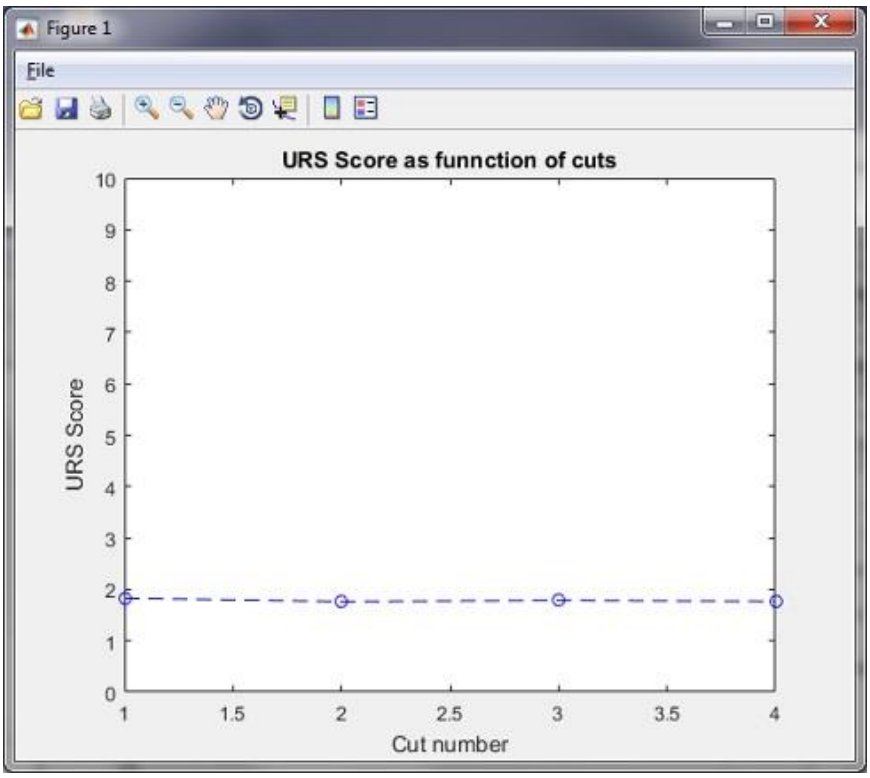
Перетворення Гауса виконану на статичній кількісній характеристиці аналізу за використанням цієї формули:

$$\frac{a}{w\sqrt{\pi/2}}\exp\left(-2\left(\frac{x-Xc}{w}\right)^2\right) \tag{1.1}$$

Ця таблиця дозволяє змінювати значення нижнього і верхнього обмеження функції перетворення.



а)



б)

Рисунок 3.15 - Зразок обмеження вхідних даних (а) та отримані оцінки (б)

Для ефективного та корисного аналізу отриманих результатів є можливість виконувати циклічні обчислення, використовуючи пропорційне зменшення аналізованих масивів даних. На кожній ітерації обмежується обсяг аналізованих даних, і отримані оцінки порівнюються. Такий підхід дозволяє отримувати ітеративні оцінки та порівнювати їх для кожної ітерації обмеження аналізованих даних.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Соціальне значення охорони праці

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров’я і працездатності людини під час трудової діяльності.

Питання охорони праці регулюються конкретними законодавчими та нормативно-правовими актами, які, визначають обов’язки роботодавця із забезпечення працівникам комфортних та безпечних умов для здійснення роботи. Ці обов’язки, а також права робітників на таких умовах праці передбачені частиною 2 ст.2 і частиною 1 ст.21 КЗпП, а також ст.13 Закону України «Про охорону праці» [15], в яких визначаються основні положення з реалізації конституційного права робітників. Існує певний ряд вимог, які визначають специфіку заходів з охорони праці при роботі з персональним комп’ютером.

Законодавчі та нормативно-правові акти, які за участі відповідних органів державної влади регулюють відносини між роботодавцем та робітником з питань безпеки, гігієни праці та виробничого середовища, а також встановлюють єдиний порядок організації охорони праці в Україні. На їх основі розроблені чисельні документи: правила, інструкції, норми, державні санітарні правила та інші нормативно-правові документи, якими мають керуватись роботодавці та які регламентують певні питання щодо конструкції електронно-обчислювальної техніки, а також особливостей їх розміщення.

Сьогодні основними документами, які регламентують питання охорони праці при використанні працівниками персональних комп’ютерів, є закон України «Про охорону праці» та наступні підзаконні акти:

– НПАОП 0.00-7.15-18 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [16];

– ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [27]. У відповідності з цими нормативними актами, при розробці будь-якого програмного забезпечення, в тому числі при розробці інформаційної системи автоматизації реєстрації сигналів моделювання механічних рухів кінцівки, необхідно вжити всіх необхідних заходів з охорони праці та розробити відповідні документи.

Окрім цього, в залежності від кількості працівників в ІТ компанії та від кількості працівників задіяних в проекті служба охорони праці виглядає наступним чином:

– В ІТ компанії або проекті з кількістю працюючих 50 і більше осіб, роботодавець створює службу охорони праці відповідно до типового положення, що затверджується центральним органом виконавчої влади, що забезпечує формування державної політики у сфері охорони праці.

– В ІТ компанії або проекті з кількістю працюючих менше 50 осіб, функції служби охорони праці можуть виконувати в порядку сумісництва особи, які мають відповідну підготовку.

– В ІТ компанії або проекті з кількістю працюючих менше 20 осіб, для виконання функцій служби охорони праці можуть залучатися сторонні спеціалісти на договірних засадах, які мають відповідну підготовку.

Підпорядковується служба охорони праці згідно із законодавством безпосередньо роботодавцеві [15]. Варто зазначити, що роботодавець може доручити функціональне управління (кураторство) діяльністю служби іншій посадовій особі, наприклад, головному інженерові, заступникові директора з охорони праці тощо. Покладення таких обов’язків необхідно закріпити наказом або відобразити в посадовій інструкції уповноваженої особи. Робота служби охорони праці підприємства має здійснюватися відповідно до плану роботи та графіків обстежень, затверджених роботодавцем.

У служби охорони праці можна виділити такі їх функції:

– Підготовка проектів наказів (розпоряджень) з питань охорони праці і внесення їх на розгляд роботодавцю. Проведення спільно з представниками інших 48 структурних підрозділів і за участю представників професійної спілки підприємства або, за її відсутності, уповноважених найманими працівниками осіб із питань охорони праці перевірок дотримання працівниками вимог нормативно-правових актів з охорони праці.

– Проведення з працівниками вступного інструктажу з питань охорони праці та супутніх інструктажів.

– Ведення обліку та проведення аналізу причин виробничого травматизму, професійних захворювань, аварій на виробництві, заподіяної ними шкоди.

– Забезпечення належного оформлення і зберігання документації з питань охорони праці.

– Складання звітності з охорони праці за встановленими формами.

– Складання за участю керівників підрозділів підприємства переліків професій, посад і видів робіт, на які повинні бути розроблені інструкції з охорони праці, що діють в межах підприємства, надання методичної допомоги під час їх розроблення.

– Інформування працівників про основні вимоги законів, інших нормативно-правових актів та актів з охорони праці, що діють в межах підприємства.

– Розгляд питань про підтвердження наявності небезпечної виробничої ситуації, що стала причиною відмови працівника від виконання дорученої роботи відповідно до законодавства (у разі необхідності).

– Організація - забезпечення підрозділів нормативно-правовими актами з охорони праці та актами з охорони праці, що діють в межах підприємства, посібниками, навчальними матеріалами з цих питань [15].

Дотримання даних пунктів є необхідним при роботі з програмним комплексом інформаційної системи автоматизації реєстрації сигналів моделювання механічних рухів кінцівки.

Отже, соціальне значення охорони праці є надзвичайно важливим аспектом діяльності всіх людей, незалежно від їх професій. Адже охорона праці дозволяє працівникам виконувати свою роботу та не турбуватись про захист їх прав при виникненні надзвичайних ситуацій у робочому процесі: відшкодування моральних, фізичних та матеріальних збитків.

4.2 Вимоги ергономіки до організації робочого місця оператора

Безпека життєдіяльності - це системний підхід до забезпечення безпеки людей у різних сферах їхнього життя і діяльності. Вона охоплює заходи, які призначені для запобігання небезпекам, зменшення ризиків та забезпечення захисту людей від шкідливих впливів, що можуть призвести до травм чи хвороб.

Приміщення клініки чи лабораторії, де перебувають медпрацівники, відносяться до приміщень без підвищеної небезпеки ураження електричним струмом. Обладнання, що використовується в цих приміщеннях є споживачем електроенергії, що живиться від змінного струму 220В від мережі з заземленою нейтраллю, та відноситься до електроустановок до 1000В закритого виконання. За способом захисту людини від ураження електричним струмом відповідає згідно з ДСТУ 3585-97 "Електробезпека. Загальні вимоги" або ДСТУ 2947-94 "Електроустановки незалежно від напруги. Загальні вимоги безпеки".

Згідно «Правилам улаштування електроустановок» (далі «ПУЕ») виконані такі групи заходів з електробезпеки: Конструктивні заходи забезпечують захист від випадкового дотику до струмопровідних частин за допомогою їх ізоляції та захисних оболонок. Згідно з ДСТУ 3747-98 "Електробезпека. Терміни та визначення" [17] у приладах II класу захисту використовується подвійна ізоляція - електрична ізоляція, що складається з робочої і додаткової ізоляції.

Для запобігання статистичного навантаження при користуванні ПК рекомендовано робити перерви по 10 хв. через кожні дві години. Синдром зап’ястного каналу, або тунельний синдром зап’ястя, який може бути наслідком хронічної травми, трапляється у людей внаслідок тривалої роботи з комп'ютерною мишею: постійні напруга і здавлювання приводить до мікротравм, стискання нерву прилеглими оточуючими тканинами, через що виникає набряк.

Щоб тунельний синдром вас не турбував, потрібно дотримуватися кількох правил організації робочого місця:

– оптимальна висота клавіатури від підлоги – 65-75 см;

– наявність ергономічних і зручних особисто для вас миші і клавіатури;

– можливість регулювання висоти, а також нахилу клавіатури (відстань від поверхні стола до середини клавіатури – не більше 30 мм, кут підйому клавіатури від 2° до 15°);

– наявність у клавіатури підставки для рук;

– наявність килимка для миші з захистом від тунельного синдрому

– наявність стільця або крісла з підлокітниками [19].

При роботі з мишкою і клавіатурою також слід дотримуватися певних правил. Коли ви набираєте текст, рука повинна бути зігнута в лікті під прямим кутом (90°), а при роботі з мишкою стежте, щоб кисть була прямою і лежала на столі якнайдалі від краю.

Для того, щоб попередити тунельний синдром потрібно робити спеціальні вправи для кистей – чим частіше, тим краще. Дані вправи допоможуть поліпшити кровообігу в м'язах і розтягнути їх. Комплекс вправ необхідно повторювати приблизно кожні 45 хвилин, тривалість однієї вправи – 1-2 хв [18].

Нервове напруження впливає на серцево-судинну систему, збільшуючи артеріальний тиск і частоту пульсу, а ще на терморегуляцію організму та емоційні стани працівника.

Особливу роль у запобіганні втомі відіграють професійний відбір, організація робочого місця, правильне робоче положення, ритм роботи,

раціоналізація трудового процесу, використання емоційних стимулів, впровадження раціональних режимів праці і відпочинку тощо.

Аби запобігти будь-яким травмам спини внаслідок тривалої роботи за комп’ютером необхідно також правильно сидіти за робочим столом, згідно ДСТУ 8604:2015 "Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги", а саме - тримати спину рівно, не нахиляти корпус тіла під час роботи, ноги тримати на підлозі повною ступнею, якщо руки на клавіатурі, то лікоть повинен бути зігнутий під кутом 90 градусів, між людиною та столом має бути відстань мінімум 5 см.

Рисунок 4.1 - Правильне положення тіла при роботі за комп’ютером

Боротьба зі втомою, в першу чергу, зводиться до покращення санітарно-гігієнічних умов виробничого середовища (ліквідація забруднення повітря, вібрації, шуму, нормалізація мікроклімату, належне освітлення тощо) [20].

ВИСНОВКИ

Результатом виконаної роботи стала розроблена математична модель методики ідентифікації есенціального тремору за допомогою спірального тесту Архімеда. Використовували методику визначення кількісних характеристик коливань внаслідок тремтіння кінцівок – амплітуди решти коливань та частотної характеристики за перетворенням Фур’є. За допомогою методів статистичної оцінки треморний стан пацієнта можна ідентифікувати за шкалою Фана-Толосса-Маріна.

Проектування та реалізація програмного забезпечення базується на розробленій математичній моделі. У процесі проектування аналізується предметна область, описуються поставлені завдання, обґрунтовується архітектура системи, описуються варіанти використання та реалізації програми. Під час реалізації програми доступні всі заплановані функції та можливості роботи.

Для дослідження характеристик захворювання було розроблено та реалізовано бібліотеку, в які необхідні є функції та алгоритми. Такоє додано опис ключових можливостей, що обчислюють параметри та функції моделі. Мною створено модель для проведення тестування та налагодження методу визначення конкретних параметрів системи з використанням рисунка Архімедової спіралі, яку намалював пацієнт, як вхідної інформації. Для більш глибокого вивчення та впровадження необхідних методів розглядаються основні системні параметри, досліджуються тестові дані та пов’язаний із ними прогрес, а також аналізуються наявні, високоефективні способи встановлення досліджуваної хвороби.

Розвиток комп'ютерного обладнання та технологій в сучасній медицині суттєво сприяє більш точному встановленню діагнозів, зокрема у випадку есенціального тремору. Сучасні комп'ютери та програмне забезпечення дозволяють обробляти великі обсяги даних і проводити складний аналіз симптомів та характеристик хвороби.

Завдяки цьому, лікарі отримують доступ до інструментів, які допомагають виявити навіть найменші відхилення в коливаннях тремору та визначити їх параметри з високою точністю. Комп'ютерні алгоритми та аналітичні моделі сприяють виявленню особливостей тремору, включаючи його амплітуду, частоту та форму.

Крім того, розвиток обладнання для вимірювання та моніторингу коливань тремору дозволяє отримати об'єктивні дані про стан пацієнта. Сенсори, акселерометри та інші пристрої забезпечують точне вимірювання та реєстрацію параметрів тремору в реальному часі. Ці дані потім можуть бути оброблені за допомогою комп'ютерних алгоритмів, що дозволяє встановити характеристики тремору з високою точністю та об'єктивністю.

Все це спільно робить можливим більш точне встановлення діагнозів, зокрема есенціального тремору, що дає лікарям засоби для більш ефективного лікування та контролю стану пацієнтів. Розроблені математичні моделі та програмне забезпечення, в поєднанні з сучасним обладнанням, відкривають нові можливості у дослідженні та діагностиці есенціального тремору,

Програмне забезпечення повинно істотно зменчити час та розходи на виконнання експериментів та досліджень. Виконана робота з розробки методу діагностики есенціального тремору допоможе покращити діагностичні умови хворих.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Elble R.J., Brilliant M., Leffler K., Higgins C. Quantification of Essential Tremor in Writing and Drawing. Movement Disorders. 1996; II(l):70-8.
2. Elble R.J., Pullman S.L., Matsumoto J.Y., Raethjen J., Deuschl G., Tintner R. Tremor Research Group Tremor amplitude is logarithmically related to 4- and 5-point tremor rating scales. Brain. 2006;129: 2660–2666. doi: 10.1093/brain/awl190. [PubMed] [Cross Ref].
3. Legrand A.P., Rivals I., Richard A., Apartis E., Roze E., Vidailhet M., Meunier S., Hainque E. New insight in spiral drawing analysis methods – application to action tremor quantification. J Clinical Neurophysiology. October 2017,Volume 128, Issue 10, P. 1823–1834. DOI: 10.1016/j.clinph.2017.07.002.
4. Kraus P.H., Hoffmann A. Spiralometry: Computerized Assessment of Tremor Amplitude on the Basis of Spiral Drawing. Movement Disorders 2010; 25(13):2164-2170.
5. Pullman S.L. Spiral analysis: a new technique for measuring tremor with digitizing tablet. Movement Disorders 1998;3:85-89.
6. Padgaonkar, A.J., Krieger, K.W. and King, A.I. (1975) Measurement of angular acceleration of a rigid body using linear accelerometers, J. Appl. Mech. (Trans. ASME), 42: 552-556.
7. Чурпій І. К., Чурпій Н. В., Скрипко В. Д. Сучасний стан інформатизації в медицині. Буковинський медичний вісник. 2011. Т. 15, N 1. С. 171-173.
8. Ковальчук О. Я., Іваницький Р. І. Експертні системи в медицині. – Тернопіль: Тернопільська державна медична академія імені І. Я. Горбачевського, 2014.
9. EMCMED. Медична інформаційна система. URL: http://mcmed.ua/ (дата звернення 23.03.2023).

10. Динамічні ігри з розривними траєкторіями / А. О. Чикрій, Ю. Г. Кривонос, І. І. Матичин . Наук. думка, 2005. 220с.
11. Ленюк М.П., Петрик М.Р. Методи інтнгральних перетворень Фур’є-Бесселя в задачах математичного моделювання масопереносу в неоднорідних середовищах. Київ: Наукова думка. 2000. 372 с.
12. Pullman, S L. Spiral analysis: a new technique for measuring tremor with digitizing tablet. Movement Disorders 1998;3:85-89.
13. André Pierre Legrand, Isabelle Rivals, Aliénor Richard, Emmanuelle Apartis, Emmanuel Roze, Marie Vidailhet, Sabine Meunier, Elodie Hainque. New insight in spiral drawing analysis methods – Application to action tremor quantification. J Clinical Neurophysiology. October 2017 Volume 128, Issue 10, Pages 1823–1834.
14. Мудрик І. Я. Автоматизовані системи діагностування стану пацієнтів, хворих на есенціальний тремор / Мудрик Іван Ярославович, 2021.
15. Закон України «Про охорону праці». [Електронний ресурс] URL: https://zakon.rada.gov.ua/laws/show/2694-12
16. Закон України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». [Електронний ресурс] URL: https://zakon.rada.gov.ua/laws/show/z0508-18
17. «Система технічних засобів і заходів електробезпеки». [Електронний ресурс] URL: https://cpo.stu.cn.ua/Oksana/posibnik/1110.html
18. «Як технології змінюють ваше тіло». [Електронний ресурс] URL: https://ua.112.ua/mnenie/yak-tekhnolohii-zminiuiut-nashe-tilo-537194.html
19. «Що таке синдром зап'ястного каналу?». [Електронний ресурс] URL: https://www.bodysport.com.ua/ua/statti/tunnel-syndrome
20. «Як правильно сидіти за комп'ютером». [Електронний ресурс] URL: https://jysk.ua/blog/yak-pravilno-siditi-za-robochim-stolom

ДОДАТКИ

ДОДАТОК А

Лістинг коду MatLab-додатку комп'ютерної моделі

```
function varargout = unravelling_gui(varargin)
% UNRAVELLING_GUI MATLAB code for unravelling_gui.fig
%      UNRAVELLING_GUI, by itself, creates a new UNRAVELLING_GUI or raises the existing
%      singleton*.
%          UNRAVELLING_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%       function named CALLBACK in UNRAVELLING_GUI.M with the given input
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn',    @unravelling_gui_OpeningFcn, ...
                   'gui_OutputFcn',  @unravelling_gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before unravelling_gui is made visible.
function  unravelling_gui_OpeningFcn(hObject,  eventdata,  handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to unravelling_gui (see VARARGIN)
```

```
    % Choose default command line output for unravelling_gui
    handles.output = hObject;
    % Update handles structure
    guidata(hObject, handles);

    % UIWAIT makes unravelling_gui wait for user response (see UIRESUME)
    % uiwait(handles.figure1);

    % --- Outputs from this function are returned to the command line.
    function varargout = unravelling_gui_OutputFcn(hObject, eventdata,
handles)
    varargout{1} = handles.output;

    % --- Executes on button press in pushbutton_Unravelling.
    % Plot Unravelling analysis
    function    pushbutton_Unravelling_Callback(hObject,    eventdata,
handles)
    global Theta Rho Thet Rh A;
    % Pullmann analysis
    figure
    hold on
    plot(unwrap(Theta),Rho,'blue');
    plot(unwrap(Theta)+2*pi,Rho,'green');
    plot(unwrap(Theta)-2*pi,Rho,'green');
    plot(unwrap(Theta)-4*pi,Rho,'blue');
    plot(unwrap(Thet),Rh,'black');
    plot(A,'red');
    ylim([0 60]);
    title('Template & Drawing')
    xlabel('Theta (radian)')
    ylabel('Rho (mm)')

    % --- Executes on button press in pushbutton_URS_fluctuations.
    % Plot URS_fluctuations
    function  pushbutton_URS_fluctuations_Callback(hObject,  eventdata,
handles)
    global C;
```

```
figure
plot(C, 'b.', 'markersize', 4);
title('URS Residual Fluctuations');
xlabel('Interpolated Time (sec)');
ylabel('Amplitude (mm)');
ax = gca;
xTicks = get(gca, 'xtick');
ax.XTickLabel = ceil(xTicks * 7.503 / 1000);

% --- Executes on button press in pushbutton_URD_fitting.
% Plot URD_fitting
function pushbutton_URD_fitting_Callback(hObject, eventdata,
handles)
global f Y NFFT D;
% Plot single-sided amplitude spectrum
figure
plot(f,2*abs(Y(1:NFFT/2+1)))
title('FFT(Residual Fluctuations) & URD determination')
xlim([0 15]);
xlabel('Frequency (Hz)')
ylabel('Y(f)')
hold on;
plot(D)
xlim([0 15]);
xlabel('Frequency (Hz)');
ylabel('Amplitude (mm)');

% Openfile button callback
function pushbutton_openfile_Callback(hObject, eventdata, handles)
global data;
%Open new file with record
[FileName,PathName] = uigetfile({'*.txt','All .txt files';'*.*','All
Files' },'mytitle');
if isequal(FileName,0) || isequal(PathName,0)
            disp('User pressed cancel')
        else
            disp(['User selected ', fullfile(PathName, FileName)])
            data = dlmread(fullfile(PathName, FileName),'',1,0);
```

```
                set(handles.textLabelOfFile, 'String', FileName);

pushbutton_compute_Callback(handles.pushbutton_compute,        eventdata,
handles);
     end
     % Recompute all calculations with new parameters of cutting
     function pushbutton_compute_Callback(hObject, eventdata, handles)
     global  Theta  Rho  S  data  Thet  Rh  URS  URD  Y  URS_Score  URD_Score
fitt_Low_a
     global   fitt_Low_w   fitt_Low_Xc   fitt_Up_a   fitt_Up_w   fitt_Up_Xc
cut_loop_points
     %-------------------------------------------------------------------
-------
                Archi = dlmread('Archi.txt',' ',1,0);
                X=Archi(:,1);
                Y=Archi(:,2);
                Theta=atan2(Y,X);
                Rho=sqrt(X.^2+Y.^2);
                S=cat(2,unwrap(Theta),Rho);
                % take care that angle is in radian

                % URS values to be collected
                URS=[];
                URD=[];

                % Collect fitting Upper and Lower limits parameters from
edit_text_boxes
             fitt_Low_a                                                 =
str2double(get(handles.edit_Lower_fitting_a,'String'));
             fitt_Low_w                                                 =
str2double(get(handles.edit_Lower_fitting_w,'String'));
             fitt_Low_Xc                                                =
str2double(get(handles.edit_Lower_fitting_Xc,'String'));
             fitt_Up_a                                                  =
str2double(get(handles.edit_Upper_fitting_a,'String'));
             fitt_Up_w                                                  =
str2double(get(handles.edit_Upper_fitting_w,'String'));
```

```
            fitt_Up_Xc                                                  =
str2double(get(handles.edit_Upper_fitting_Xc,'String'));
         cut_loop_points                                                =
str2double(get(handles.edit_cut_loop_points,'String'));

                % initialization of loop and cutting variable
                loop = get(handles.popupmenu_loop,'Value')-1;
                cut_begin                                               =
str2double(get(handles.cut_from_begin,'String'));
                cut_end                                                 =
str2double(get(handles.cut_from_end,'String'));
                if isnan(cut_begin)
                    set(handles.cut_from_begin,'string','0');
                    %warndlg('Input must be numerical');
                    return
                end
                if isnan(cut_end)
                    set(handles.cut_from_end,'string','0');
                    %warndlg('Input must be numerical');
                    return
                end

                if loop == 0
                    % data is the patient drawing record
                    Xi=data(cut_begin+1:end-cut_end,2);
                    Yi=data(cut_begin+1:end-cut_end,3);
                    T=data(cut_begin+1:end-cut_end,1);
                    calculation(Xi, Yi, T);

                elseif loop >= 1
                    for i=1:loop;
                        low=cut_loop_points*i;
                        Xi=data(low+cut_begin:end-low-cut_end,2);
                        Yi=data(low+cut_begin:end-low-cut_end,3);
                        T=data(low+cut_begin:end-low-cut_end,1);
                        calculation(Xi, Yi, T);
                    end
                end
```

```
                % in mm
                % Score using the Rivals's scale range: (0-10)
                URS_Score=4.52694+10.2136*log10(URS);
                URS_Score_moy = mean(URS_Score);
                URS_mm = mean(URS);
                %URS_mm = geomean(URS);

                % in mm
                % Score using the Rivals's scale range: (0-10)
                URD_Score=9.20773+3.7515*log10(URD);
                URD_Score_moy = mean(URD_Score);
                URD_mm = mean(URD);
                %URD_mm = geomean(URD);
                %-------------------------------------------------------
-------------
                % Set editText fields with value from URS and URD
        set(handles.URS_score_label,                          'String',
sprintf('%2.2f',URS_Score_moy));
        set(handles.URD_score_label,                          'String',
sprintf('%2.2f',URD_Score_moy));
        set(handles.URS_mm_label, 'String', sprintf('%2.4f',URS_mm));
        set(handles.URD_mm_label, 'String', sprintf('%2.4f',URD_mm));

                % Plot spiral and template on the main screen
                cla()
                p=polar(Theta, Rho);
                set(p,'Color','black','LineWidth',1)
                hold on
                polar(Thet,Rh,'red');

    function calculation(Xi, Yi, T)
    global Thet Rh A B C URS URD f Y NFFT D fitt_Low_a fitt_Low_w
fitt_Low_Xc fitt_Up_a fitt_Up_w fitt_Up_Xc;

    Thet=atan2(Yi,Xi);
```

```
      Rh=sqrt(Xi.^2+Yi.^2);
      % take care that angle is in radian
      Patient=cat(2,T,unwrap(Thet),Rh);
      % fitting to suppress paralaxe error
      op=fitoptions('Methods','NonlinearLeastSquares','StartPoint',[1  1  1
1]);
      ftype=fittype('a+b*x+c*cos(x)+d*sin(x)','options',op);

      % to overcome the NaN values in the data record
      x = Patient(:,2);
      l = ~isnan(x);
      x = x(l);
      y = Patient(:,3);
      k = ~isnan(y);
      y = y(k);
      t = Patient(:,1);
      m = ~isnan(t);
      t = t(m);
      % fitting with linear sinus
      [A,B]=fit(x, y, ftype);

      % evaluation of the URS
      C=y-A(x);
      URS=[URS;std(C)];

      % FFT of C
      NFFT=2^nextpow2(length(C));
      Y=fft(C,NFFT)/length(C);
      f=(1/0.0075)/2*linspace(0,1,NFFT/2+1);

      % URD file for Gaussian fitting
      op=fitoptions('Methods','NonlinearLeastSquares','StartPoint',[1    1
1]);
      op.Lower=[fitt_Low_a fitt_Low_w fitt_Low_Xc];
      op.Upper=[fitt_Up_a fitt_Up_w fitt_Up_Xc];
      ftype=fittype('a/(w*sqrt(pi/2))*exp(-2*((x-
xc)/w)^2)','options',op);
      % gaussian fitting
```

```
[D,E]=fit(f.',2*abs(Y(1:NFFT/2+1)),ftype);
F=coeffvalues(D);
URD=[URD;F(1)];

% Plot Scores comparation figures
function compare_Scores_comparation_Callback(hObject, eventdata,
handles)
global URS_Score URD_Score
figure
plot(URS_Score,'b--o');
title('URS Score as funnction of cuts');
xlabel('Cut number');
ylabel('URS Score');
ylim([0 10]);
%hold on
figure
plot(URD_Score,'r--o');
title('URD Score as funnction of cuts');
xlabel('Cut number');
ylabel('URD Score');
ylim([0 10]);
```

Лістинг коду Java-бібліотеки для програми «SpiralAnalyser»

1. Клас FFT

```
public class FFT
{
    static void twoDfft(double[][] inputData, double[][] realOut,
            double[][] imagOut, double[][] amplitudeOut)
    {
        int height = inputData.length;
        int width = inputData[0].length;

        // Two outer loops iterate on output data.
        for (int yWave = 0; yWave < height; yWave++)
        {
            for (int xWave = 0; xWave < width; xWave++)
            {
                // Two inner loops iterate on input data.
                for (int ySpace = 0; ySpace < height; ySpace++)
                {
                    for (int xSpace = 0; xSpace < width; xSpace++)
                    {
                    // Compute real, imag, and ampltude.
```

```
          realOut[yWave][xWave]    +=    (inputData[ySpace][xSpace]    *
Math.cos(2 * Math.PI * ((1.0 * xWave * xSpace / width) + (1.0 * yWave *
ySpace / height)))) / Math.sqrt(width * height);
          imagOut[yWave][xWave]    -=    (inputData[ySpace][xSpace]    *
Math.sin(2 * Math.PI * ((1.0 * xWave * xSpace / width) + (1.0 * yWave *
ySpace / height)))) / Math.sqrt(width * height);
     amplitudeOut[yWave][xWave]   =   Math.sqrt(realOut[yWave][xWave   *
realOut[yWave][xWave] + imagOut[yWave][xWave] * imagOut[yWave][xWave]);
}
           return imagOut[yWave][xWave];
                     }
                }
             }
         }
```

2. Клас ProcessSpiral

```
package geom;

import static java.lang.Math.atan2;
import static java.lang.Math.cos;
import static java.lang.Math.pow;
import static java.lang.Math.sin;
import static java.lang.Math.sqrt;
import static java.lang.Math.toDegrees;
import static java.lang.Math.toRadians;

public class ProcessSpiral {

    /*
     * This variable represents the state of the output.
     * It is static because any change to this variable is reflected
in all the
     * instances.
     */
    private static State state;
    /**
     * The Point object representing the reference point at the
intersection of
     * the coordinate axes. Essentially, it is a point whose
coordinates are
     * <pre>(0,0)</pre>.
     */
    public static final Point ORIGIN = getCartesianPoint(0, 0);

    /**
     * This factory method returns a {@code Point} object with the
(Cartesian)
     * coordinates passed as parameters.
     *
     * @param x The X-coordinate of the {@code Point} object.
     * @param y The Y-coordinate of the {@code Point} object.
     * @return The required {@code Point} object.
     */
    public static Point getCartesianPoint(double x, double y) {
        Point p = new Point();
        p.x = x;
```

```
            p.y = y;
            p.radius = sqrt(x * x + y * y);
            p.angleR = atan2(x, y);
            p.angleD = toDegrees(p.getAngleRadians());
            return p;
        }

        /**
         * This factory method returns a {@code Point} object with the
(Polar)
         * coordinates passed as the parameters.
         *
         * @param radius The distance of the required {@code Point} from
the origin
         * (0,0)
         * @param degrees The angle between the line joining the required
         * {@code Point} and the origin and the X-axis (in degrees i.e.
from 0 to
         * 360).
         * @return The required {@code Point} object.
         */
        public static Point getPolarDegreesPoint(double radius, double
degrees) {
            Point p = new Point();
            p.radius = radius;
            p.angleD = degrees;
            p.angleR = toRadians(degrees);
            initPolar(p);
            return p;
        }

        public static Point getPolarRadiansPoint(double radius, double
radians) {
            Point p = new Point();
            p.radius = radius;
            p.angleR = radians;
            p.angleD = toDegrees(radians);
            initPolar(p);
            return p;
        }

        /*
         * This method is common to both the 'getPolar_______Point()'
methods.
         */
        private static void initPolar(Point point) {
            double angle = point.getAngleRadians();
            point.x = point.getRadius() * cos(angle);
            point.y = point.getRadius() * sin(angle);
        }

        /**
         * This method is used to change the form of representation of
<em>ALL</em>
         * {@code Point} objects.
         *
         * @see State
```

```
         * @param state The {@code State} constant to set.
         */
        public static void setState(State state) {
            Point.state = state;
        }

        /*
         * The coordinates of this Point in the Cartesian system.
         */
        private double x;        // The perpendicular distance from the
Y-axis.
        private double y;        // The perpendicular distance from the
X-axis.

        /*
         * The coordinates of this Point in the Polar System.
         */
        private double radius;  // The distance from the Origin (0,0).
        private double angleR;  // The angle in Radians.
        private double angleD;  // The angle in Degrees.

        public double distanceFrom(Point other) {
            return other.equals(Point.ORIGIN) ? radius
                    : sqrt(pow(this.getX() - other.getX(), 2)
                            + pow(this.getY() - other.getY(), 2));
        }

        public Point reflectionXAxis() {
            return getCartesianPoint(getX(), -getY());
        }


        public Point reflectionYAxis() {
            return getCartesianPoint(-getX(), getY());
        }


        public Point reflectionOrigin() {
            return getCartesianPoint(-getX(), -getY());
        }


        public Point reflectionFrom(Point other) {
            if (other.equals(Point.ORIGIN)) {
                return reflectionOrigin();
            }
            return getCartesianPoint(
                    2 * other.getX() - this.getX(),
                    2 * other.getY() - this.getY());

        }

        public double getX() {
            return x;
        }

        public double getY() {
```

```
    return y;
}

public double getRadius() {
    return radius;
}

public double getAngleRadians() {
    return angleR;
}

public double getAngleDegrees() {
    return angleD;
}

public Quadrant getLocation() {
    if (location == null) {
        if (this.equals(Point.ORIGIN)) {
            location = Quadrant.ON_ORIGIN;
        } else if (x == 0) {
            location = Quadrant.ON_Y_AXIS;
        } else if (y == 0) {
            location = Quadrant.ON_X_AXIS;
        } else if (x > 0 && y > 0) {
            location = Quadrant.FIRST_QUADRANT;
        } else if (x < 0 && y > 0) {
            location = Quadrant.SECOND_QUADRANT;
        } else if (x < 0 && y < 0) {
            location = Quadrant.THIRD_QUADRANT;
        } else if (x > 0 && y < 0) {
            location = Quadrant.FOURTH_QUADRANT;
        }
    }
    return location;
}

@Override
public boolean equals(Object o) {
    if (o instanceof Point) {
        Point p = (Point) o;
        return this.hashCode() == p.hashCode();
    }
    return false;
}

@Override
public int hashCode() {
    int hash = 0;
    hash += (int) (Double.doubleToLongBits(this.getX())
            ^ (Double.doubleToLongBits(this.getX()) >>> 32));
    hash += (int) (Double.doubleToLongBits(this.getY())
            ^ (Double.doubleToLongBits(this.getY()) >>> 32));
    return hash;
}

@Override
public String toString() {
```

```
            Thread t = new Thread();
            String summary = "\tCartesian:\t( x\t: " + x + ", y\t: " + y
+ " )";
            if (state == null) {
                setState(State.SHORT_SUMMARY);
            }
            if (!state.equals(State.NO_SUMMARY)) {
                summary  +=  "\n\tPolar:\n\t\tDegrees\t(  radius\t:  "  +
radius + ", angle\t: "
                        + angleD + " )\n";
                summary  +=  "\t\tRadians\t(  radius\t:  "  +  radius  +  ",
angle\t: "
                        + angleR + " )\n";
            }
            if (state.equals(State.LONG_SUMMARY)) {
                summary += "\tQuadrant\t: " + getLocation();
                // summary += "\n\t" + Integer.toHexString(hashCode());
            }
            return summary;
        }

        /**
         *
         */
        @SuppressWarnings("PublicInnerClass")
        public static enum State {
            LONG_SUMMARY,
            SHORT_SUMMARY,
            NO_SUMMARY;
        }

        @SuppressWarnings("PublicInnerClass")
        public static enum Quadrant {

            FIRST_QUADRANT,
            SECOND_QUADRANT,
            THIRD_QUADRANT,
            FOURTH_QUADRANT,
            ON_X_AXIS,
            ON_Y_AXIS,
            ON_ORIGIN;
        }
    }
```

Код класу для виконання GaussianFitting

```
    public class Fit {

        // return pdf(x) = standard Gaussian pdf
        public static double pdf(double x) {
            return Math.exp(-x*x / 2) / Math.sqrt(2 * Math.PI);
        }

        // return pdf(x, mu, signma) = Gaussian pdf with mean mu and
stddev sigma
        public static double pdf(double x, double mu, double sigma) {
            return pdf((x - mu) / sigma) / sigma;
```

```
        }

        // bisection search
        private static double inverseCDF(double y, double delta, double
lo, double hi) {
            double mid = lo + (hi - lo) / 2;
            if (hi - lo < delta) return mid;
            if (cdf(mid) > y) return inverseCDF(y, delta, lo, mid);
            else              return inverseCDF(y, delta, mid, hi);
        }


        // return phi(x) = standard Gaussian pdf
        @Deprecated
        public static double phi(double x) {
            return pdf(x);
        }
        // return phi(x, mu, signma) = Gaussian pdf with mean mu and
stddev sigma
        @Deprecated
        public static double phi(double x, double mu, double sigma) {
            return pdf(x, mu, sigma);
        }

        }

      public CurveFitter (double[] xData, double[] yData) {
             this.xData = xData;
             this.yData = yData;
             numPoints = xData.length;
      }
      public void doFit(int fitType) {
             doFit(fitType, false);
      }
```

ДОДАТОК Б

Диск з кваліфікаційною роботою бакалавра