Національна академія наук України
Інститут телекомунікацій і глобального інформаційного
простору НАН України

Кваліфікаційна наукова
праця на правах рукопису

БУЦІЙ РОМАН АНДРІЙОВИЧ

УДК 519.65

ДИСЕРТАЦІЯ

МОДЕЛЮВАННЯ ТА МЕТОДИ ЕФЕКТИВНОГО ОПРАЦЮВАННЯ ЦИКЛІЧНИХ СИГНАЛІВ В НЕЙРОІНТЕРФЕЙСНИХ ТА КАРДІО-ДІАГНОСТИЧНИХ СИСТЕМАХ

113 – Прикладна математика

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ /Р.А. Буцій/

Науковий керівник:
доктор технічних наук, професор
Лупенко Сергій Анатолійович

Ідентичність усіх примірників дисертації
ЗАСВІДЧУЮ:
Вчений секретар спеціалізованої вченої ради

Київ – 2024

АНОТАЦІЯ

Буцій Р. А. “Моделювання та методи ефективного опрацювання циклічних сигналів в нейроінтерфейсних та кардіодіагностичних системах”. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 113 “Прикладна математика”. – Інститут телекомунікацій і глобального інформаційного простору НАН України, Київ, 2024.

Зміст анотації. Дисертація присвячена розробці та аналізу нових математичних моделей та методів опрацювання циклічних біомедичних сигналів, перш за все, сигналів електроенцефалографії (ЕЕГ) та електрокардіографії (ЕКГ). Значна увага приділяється вдосконаленню методів цифрового опрацювання циклічних біомедичних сигналів, що сприяє підвищенню точності та зниженню часової обчислювальної складності опрацювання ЕЕГ сигналів в кардіо-діагностичних системах, системах біометричної аутентифікації особи та опрацювання ЕЕГ сигналів в неінвазивних нейроінтерфейсних системах.

Одним із основних результатів дисертації є розробка нової математичної моделі векторної ЕЕГ, зареєстрованої в умовах багатократного повторення ментального керуючого впливу оператора нейроінтерфейсу, у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів, яка має більший теоретичний та практичний потенціал у порівнянні з відомими моделями ЕЕГ сигналів. Ця модель враховує циклічність і стохастичність синхронно зареєстрованих біосигналів, змінність та спільність їх ритму, надаючи нові можливості для підвищення точності, інформативості методів статистичного цифрового опрацювання ЕЕГ в сучасних неінвазивних нейроінтерфейсних системах. Важливість цієї моделі для нейроінтерфейсних систем полягає у вдосконаленні методів ритмоадаптивного статистичного опрацювання ЕЕГ сигналів та алгоритмів ідентифікації (детекції) ментальних керуючих впливів оператора нейроінтерфейсної системи, що значно підвищує ефективність взаємодії між людиною та комп’ютером.

Значна частина дисертаційного дослідження присвячена розробці ефективних методів ритмоадаптивного опрацювання ЕКГ сигналів в системах медичної діагностики та біометричної аутентифікації особи на основі математичної моделі ЕКГ сигналів у вигляді циклічного випадкового процесу. Розроблені методи та алгоритми забезпечують високу точність та низьку обчислювальну складність алгоритмів медичної діагностики та біометричної аутентифікації за ЕКГ сигналами.

Дисертація також включає розробку програмного забезпечення на мові Python, яке реалізує вищезгадану математичну модель. Це програмне забезпечення дозволяє автоматизувати опрацюваня, аналіз та класифікацію ЕЕГ та ЕКГ сигналів.

У вступі обґрунтовано актуальність дослідження, наведено зв'язок роботи з науково-дослідною темою, поставлено мету та визначено завдання дослідження, об'єкт та предмет дослідження, наведено перелік методів дослідження, що застосовувались для досягнення мети дисертаційної роботи. Сформульовано наукову новизну, практичне значення отриманих результатів та особистий творчий внесок здобувача. Подано відомості щодо апробації та опублікування результатів дослідження.

У першому розділі проведено аналіз сучасних технологій аналізу біосигналів, які є критично важливими для розуміння фізіологічних процесів у живих організмах. Основну увагу приділено циклічним біосигналам, таким як ЕКГ і ЕЕГ, які відіграють ключову роль у медичній діагностиці, біометричній аутентифікації особи та у неінвазивних нейроінтерфейсних системах. Розглянуто напрями покращення методів збору біомедичних даних, що включають носимі технології та технології дистанційного моніторингу, які сприяють більшій зручності та точності в аналізі циклічних біосигналів. Також акцентується увага на використанні цих сигналів для потреб біометричної аутентифікації особи за ЕКГ сигналами.

Досліджено відомі математичні моделі та методи аналізу циклічних біосигналів, які включають ЕЕГ та ЕКГ. Значна увага приділяється питанню

адекватності та точності цих математичних моделей у задачах медичної діагностики, біометричної аутентифікації та детекції менальних керуючих імпульсів оператора нейроінтерфейсної системи. Розглядаються як стохастичні, так і детерміновані підходи до моделювання циклічних біомедичних сигналів, а також їхнє застосування в клінічних та дослідницьких цілях, з особливим акцентом на розвиток програмного забезпечення для імплементації цих моделей.

Також у розділі проведено порівняльний аналіз відомих математичних моделей ЕКГ та ЕЕГ. Основна увага приділяється напрямам підвищення адекватності моделей та ефективності методів статистичного опрацювання ЕКГ та ЕЕГ в системах медичної діагностики, біометричної аутентифікації особи та в неінвазивних нейроінтерфейсних системах.

У другому розділі розглянуто визначальні для математичного моделювання властивості векторної ЕЕГ. Встановлюється потреба у багатоциклових дослідженнях для навчання нейроінтерфейсних систем з ціллю ефективного виявлення ментальних впливів, а також залучення статистичних методів для моделювання циклічної структури векторної ЕЕГ.

Детально розглядається процедура побудови вектора циклічних ритмічно пов’язаних випадкових процесів, який описує векторний ЕЕГ за умови багаторазового повторення ментальних керуючих впливів оператора нейроінтерфейсу. Описується методологія опрацювання ЕЕГ, яка має засоби ритмоадаптації та забезпечує точне відтворення часових інтервалів між циклами ЕЕГ, що має важливе значення для врахування повторюваності та мінливості ЕЕГ сигналів в методах їх ефективного опрацювання. Наведено ритмоадаптивні статистичні методи для оцінювання ймовірнісних характеристик ЕЕГ, що дозволило точніше досліджувати ймовірнісну структуру векторної ЕЕГ у системах інтерфейс мозок-комп'ютер.

У третьому розділі дисертаційної роботи розглянуто ключові етапи опрацювання ЕЕГ сигналів у нейроінтерфейсних системах, що включають у себе реєстрацію ЕЕГ сигналів за допомогою платформи OpenBCI, їх попереднє

опрацювання, статистичний аналіз на основі нової математичної моделі векторної ЕЕГ та класифікацію.

На етапі попереднього опрацювання ЕЕГ сигналів здійснюється їх фільтрація для видалення шумів і зовнішніх артефактів, що підвищує якість даних для наступних етапів аналізу. Показано, що використання нових статистичних методів та алгоритмів дає змогу з високою точністю оцінити ймовірнісні характеристики ЕЕГ сигналів, що є критично важливим для ефективної класифікації ментального керуючого впливу оператора нейроінтерфейсу.

У розділі описано розроблене програмне забезпечення на базі Python із застосуванням бібліотек таких як Pandas, Numpy, Scipy, Sklearn та Matplotlib, що надає зручні інструменти для автоматизованого опрацювання, аналізу та візуалізації ЕЕГ даних. Описано експериментальну частину дослідження, що включає детальну перевірку розробленої математичної моделі та аналіз ефективності нейроінтерфейсних систем у залежності від наявності чи відсутності досвіду та тренувань операторів нейроінтерфейсної системи.

Завершується розділ дискусією про результати класифікації, де порівнюються різні методики і підходи до опрацювання ЕЕГ сигналів, з особливим акцентом на їх спектральному аналізі, який виявився особливо ефективним для ідентифікації ментальних команд. Висвітлено ключові ідеї та потенціал подальших досліджень у даній області, що відкриває шлях для удосконалення неівазивних нейроінтерфейсних систем.

У четвертому розділі досліджуються методи статистичного ритмоадаптивного опрацювання електроенцефалографічних (ЕКГ) та сейсмокардіографічних (СКГ) сигналів в задачах медичної діагностики та біометричної аутентифікації особи. У розділі основну увагу зосереджено на аналізі характеристик ЕКГ та СКГ сигналів на основі ритмоадаптивних статистичних оцінок їх моментних функцій. Продемонстровано високу ефективність біометричної аутентифікації особи за її ЕКГ сигналами на основі розробленого у дисертації методу, демонструючи зв'язок між біометричними

даними та потенціалом їх використання в безпекових системах. Обговорюються етапи опрацювання ЕКГ сигналів, включаючи їх реєстрацію, сегментацію, нормалізацію, а також класифікацію. Представлено порівняльний аналіз різних класифікаторів на основі їхньої ефективності у біометричній аутентифікації осіб з бази даних Combined Measurement of Electrocardiograms, Breathing, and Seismocardiograms. Демонструється, як статистичні характеристики і час опрацювання сигналів впливають на вибір класифікаторів, враховуючи потреби швидкості та точності в сучасних системах безпеки.

Продемонтровано ефективність розробленого у дисертації підходу до опрацювання ЕКГ сигналів на прикладі вирішення актуального завдання автоматизованої діагностики аритмії у пацієнтів. Досліджено основні характеристики ефективності та часова обчислювальна складність алгоритмів для навчання та тестування класифікаторів у системах медичної діагностик аритмій за ЕКГ.

Ключові слова: математичне моделювання, статистичні методи опрацювання, циклічні сигнали, електроенцефалограма, електрокардіограма, медична діагностика, біометрична аутентифікація, нейроінтерфейс.

SUMMARY

Butsiy R. A. “Modeling and Methods for Effective Processing of Cyclic Signals in Neurointerface and Cardio-diagnostic Systems”. – Qualifying scientific work on the rights of the manuscript.

Thesis for the degree of Doctor of Philosophy in specialty 113 "Applied Mathematics". – Institute of Telecommunications and Global Information Space of the National Academy of Sciences of Ukraine, Kyiv, 2024.

Abstract Content. The thesis is dedicated to the development and analysis of new mathematical models and methods for processing cyclic biomedical signals, primarily electroencephalography (EEG) and electrocardiography (ECG) signals. Significant attention is given to the improvement of digital processing methods for cyclic biomedical signals, which enhances the accuracy and reduces the computational time for processing EEG signals in cardio-diagnostic systems, biometric authentication systems, and processing EEG signals in non-invasive neurointerface systems.

One of the main results of the thesis is the development of a new mathematical model of vector EEG, recorded under conditions of repeated mental control influence by a neurointerface operator, in the form of a vector of cyclic rhythmically connected random processes. This model surpasses the theoretical and practical potential compared to known EEG signal models. It accounts for the cyclicality and stochasticity of synchronously registered biosignals, the variability and commonality of their rhythm, providing new possibilities for enhancing the accuracy and informativeness of statistical digital processing methods of EEG in modern non-invasive neurointerface systems. The significance of this model for neurointerface systems lies in the enhancement of rhythm-adaptive statistical processing methods of EEG signals and identification (detection) algorithms of mental control influences of the neurointerface system operator, significantly improving the interaction between human and computer.

A substantial part of the thesis research is dedicated to developing effective methods for rhythm-adaptive processing of ECG signals in medical diagnostic and biometric authentication systems based on a mathematical model of ECG signals as a

cyclic random process. The developed methods and algorithms ensure high accuracy and low computational complexity of algorithms for medical diagnosis and biometric authentication by ECG signals.

The thesis also includes the development of software in Python, which implements the aforementioned mathematical model. This software enables the automation of processing, analysis, and classification of EEG and ECG signals.

In the introduction, the relevance of the research is justified, the connection of the work with the scientific research topic is presented, the goal is set and the tasks of the research are defined, the object and subject of the research are outlined, and a list of research methods applied to achieve the goals of the thesis is provided. The scientific novelty, practical significance of the obtained results, and the personal creative contribution of the candidate are articulated. Information regarding the validation and publication of the research results is also presented.

In the first section, an analysis of modern technologies for bio-signal analysis, which are critically important for understanding physiological processes in living organisms, is conducted. Special attention is paid to cyclic bio-signals such as ECG and EEG, which play a key role in medical diagnostics, biometric authentication, and non-invasive neurointerface systems. Directions for improving methods of biomedical data collection are considered, including wearable technologies and remote monitoring technologies, which enhance convenience and accuracy in cyclic bio-signal analysis. The use of these signals for biometric authentication purposes based on ECG signals is also emphasized.

Known mathematical models and methods for analyzing cyclic bio-signals, including EEG and ECG, are examined. Considerable attention is given to the adequacy and accuracy of these mathematical models in tasks of medical diagnostics, biometric authentication, and detection of mental control impulses by a neurointerface system operator. Both stochastic and deterministic approaches to modeling cyclic biomedical signals are discussed, as well as their application in clinical and research purposes, with a special focus on the development of software for implementing these models.

The section also conducts a comparative analysis of known mathematical models of ECG and EEG. The main focus is on directions for improving the adequacy of models and the effectiveness of statistical processing methods of ECG and EEG in systems for medical diagnostics, biometric authentication, and in non-invasive neurointerface systems.

In the second section, the defining properties for mathematical modeling of vector EEG are considered. The need for multicycle studies for training neurointerface systems with the aim of effectively detecting mental influences, as well as the involvement of statistical methods for modeling the cyclic structure of vector EEG, is established.

A detailed examination of the process of constructing a vector of cyclic rhythmically connected random processes, which describes vector EEG under conditions of repeated mental control influences by a neurointerface operator, is conducted. The methodology for processing EEG, which has rhythm-adaptation capabilities and ensures accurate reproduction of time intervals between EEG cycles, is described. This is crucial for accounting for the repeatability and variability of EEG signals in their effective processing methods. Rhythm-adaptive statistical methods for assessing the probabilistic characteristics of EEG are presented, allowing for a more accurate study of the probabilistic structure of vector EEG in brain–computer interface systems.

In the third section of the thesis, key stages in the processing of EEG signals in neurointerface systems are examined, which include the registration of EEG signals using the OpenBCI platform, their preliminary processing, statistical analysis based on the new mathematical model of vector EEG, and classification.

In the preliminary processing stage of EEG signals, filtration is carried out to remove noise and external artifacts, which improves the quality of data for subsequent analysis stages. It is shown that the use of new statistical methods and algorithms allows for a highly accurate estimation of the probabilistic characteristics of EEG signals, which is critically important for effective classification of the mental control influence of a neurointerface operator.

The section describes the developed software based on Python using libraries such as Pandas, Numpy, Scipy, Sklearn, and Matplotlib, which provides convenient tools for automated processing, analysis, and visualization of EEG data. The experimental part of the research, which includes a detailed verification of the developed mathematical model and analysis of the effectiveness of neurointerface systems depending on the presence or absence of experience and training of neurointerface system operators, is described.

The section concludes with a discussion on classification results, comparing various methodologies and approaches to EEG signal processing, with a particular emphasis on their spectral analysis, which proved to be particularly effective for identifying mental commands. Key ideas and the potential for further research in this area are highlighted, paving the way for the improvement of non-invasive neurointerface systems.

The fourth section examines the methods of statistical rhythm-adaptive processing of electroencephalographic (EEG) and seismocardiographic (SCG) signals in the tasks of medical diagnostics and biometric authentication of a person. In the section, the main attention is focused on the analysis of the characteristics of ECG and SCG signals on the basis of rhythm-adaptive statistical estimates of their moment functions. High effectiveness of biometric authentication of persons based on their ECG signals using the developed method in the thesis is also demonstrated, highlighting the connection between biometric data and the potential for their use in security systems. The stages of processing ECG signals, including their registration, segmentation, normalization, and classification, are discussed. The section also discusses comparative analysis of different classifiers based on their effectiveness in biometric authentication of persons from the Combined Measurement of Electrocardiograms, Breathing, and Seismocardiograms database. It is demonstrated how statistical characteristics and the time of signal processing influence the choice of classifiers, considering the needs for speed and accuracy in modern security systems.

The effectiveness of the approach to the processing of ECG signals developed in the dissertation was demonstrated on the example of solving the actual task of

automated diagnosis of arrhythmia in patients. The main efficiency characteristics and time computational complexity of algorithms for training and testing classifiers in systems of medical diagnosis of arrhythmias based on ECG were studied.

Key words: mathematical modeling, statistical signal processing, cyclic signals, electroencephalogram, electrocardiogram, medical diagnostics, biometric authentication, neurointerface.

Список публікацій здобувача за темою дисертації

Праці, в яких опубліковано основні наукові результати дисертації:

1. Lupenko, S.; Butsiy, R. Isomorphic Multidimensional Structures of the Cyclic Random Process in Problems of Modeling Cyclic Signals with Regular and Irregular Rhythms. Fractal Fract. 2024, 8, 203. [Article, Scopus, Web of Science, Google Scholar, JCR – Q1, CiteScore - Q1, SJR – Q2, Impact Factor 5.4, ISSN 2504-3110]
doi:https://doi.org/10.3390/fractalfract8040203

2. Lupenko, S.; Butsiy, R.; Shakhovska, N. Advanced Modeling and Signal Processing Methods in Brain–Computer Interfaces Based on a Vector of Cyclic Rhythmically Connected Random Processes. Sensors 2023, 23, 760. [Article, Scopus, Web of Science, Google Scholar, JCR - Q2, CiteScore - Q1, SJR – Q2, Impact Factor 3.9, ISSN 1424-8220]
doi:https://doi.org/10.3390/s23020760

3. Lupenko, S.; Butsiy, R.; Volyanyk, O.; Stadnyk, N. Advanced Signal Processing and Classification of EEG Patterns in Neurointerface Systems. In Proceedings of the 3rd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2023), Ternopil, Ukraine, Opole, Poland, 22–24 November 2023, 3628, 156–164. [Article, Scopus, Web of Science, Google Scholar, ISSN 1613-0073]

4. Butsiy, R.; Lupenko, S. Comparison of Modern Methods of Classification of EEG Patterns for Neurointerface Systems. In: Yang, XS., Sherratt, S., Dey, N., Joshi, A. (eds) Proceedings of Seventh International Congress on Information and Communication Technology. Lecture Notes in Networks and Systems; Springer, Singapore, 2022, 465, 345-354. [Article, Scopus, Google Scholar, ISSN 2367-3370]
doi:https://doi.org/10.1007/978-981-19-2397-5_32

5. Butsiy, R.; Lupenko, S.; Zozulya, A. Comprehensive justification for the choice of software development tools and hardware components of a multi-channel neurointerface system. 2021 IEEE 16th International Conference on Computer

Sciences and Information Technologies (CSIT), 2021, 1, 309-312. [Article, Scopus, Google Scholar, ISSN 2766-3639]
doi:https://doi.org/10.1109/CSIT52700.2021.9648788

6. Butsiy, R.; Lupenko, S. Comparative Analysis of Neurointerface Technologies for the Problem of Their Reasonable Choice in Human-Machine Information Systems. Sci. J. TNTU (Tern.), 2020, 4, 135–148. [Article, Index Copernicus, Google Scholar, ISSN 2522-4433]
doi:https://doi.org/10.33108/visnyk_tntu2020.04.135

7. Lupenko, S.; Butsiy, R. Express Method of Biometric Person Authentication Based on One Cycle of the ECG Signal. Sci. J. TNTU (Tern.), 2024, 113, 100–110. [Article, Index Copernicus, ISSN 2522-4433]
doi:https://doi.org/10.33108/visnyk_tntu2024.01.100

Праці, які засвідчують апробацію матеріалів дисертації:

8. Лупенко, С.; Буцій, Р. Сучасні нейроінтерфейсні технології: актуальність, перспективи та складності. У Матеріалах міжнародної наукової конференції “Іван Пулюй: життя в ім’я науки та України” (до 175-ліття від дня народження), 28-30 вересня 2020 р.; ТНТУ: Тернопіль, Україна, 2020, 81–82. [Conference Paper, Google Scholar]

9. Буцій, Р.; Лупенко, С. Аналіз основних характеристик комерційних нейроінтерфейсів. У Матеріалах IX Міжнародної науково-технічної конференції молодих учених та студентів, 25-26 листопада 2020 р.; ТНТУ: Тернопіль, Україна, 2020, 2, 9–10. [Conference Paper, Google Scholar]

10. Катеринюк, І.; Лупенко, С.; Буцій, Р. Аудіоінтерфейсні та нейроінтерфейсні технології вводу діагностичної інформації в інформаційну систему “Імідж-Терапевт” для народної медицини. У Матеріалах IX Міжнародної науково-технічної конференції молодих учених та студентів, 25-26 листопада 2020 р.; ТНТУ: Тернопіль, Україна, 2020, 2, 24–25. [Conference Paper, Google Scholar]

11. Буцій, Р.; Лупенко, С. Аналіз методів для задач опрацювання сигналів нейроінтерфейсних систем. У Матеріалах VIII науково-технічної конференції “Інформаційні моделі, системи та технології”, 9-10 грудня 2020 р.; ТНТУ: Тернопіль, Україна, 2020, 3. [Conference Paper, Google Scholar]

12. Буцій, Р.; Лупенко, С. Підхід до побудови доступних за ціною дослідницьких нейроінтерфейсних систем. У Матеріалах XX Міжнар. наук.-практ. конф. “Сучасні інформаційні технології управління екологічною безпекою, природокористуванням, заходами в надзвичайних ситуаціях”, 4-8 жовтня 2021 р.; Юстон: Київ, Україна, 2021, 131–134. [Conference Paper]

13. Буцій, Р.; Лупенко, С. Принцип керування роботизованою рукою зі зворотним зв’язком за допомогою нейроінтерфейсу. У Матеріалах IX Науково-технічної конференції “Інформаційні моделі, системи та технології”, 8-9 грудня 2021 р.; Тернопіль, Україна, 2021, 3. [Conference Paper]

14. Лупенко, С; Буцій, Р. Математична модель векторної ЕЕГ у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів. У Матеріалах XXI Міжнар. наук.-практ. конф. "Інформаційно-комунікаційні технології та сталий розвиток", 14–16 листопада 2022 р.; Юстон: Київ, Україна, 2022, 49–52.
[Conference Paper, Google Scholar]

15. Лупенко, С; Буцій, Р. Дослідження класифікаторів для інтерфейсів мозок-комп'ютер на основі сигналів ЕЕГ. У Матеріалах XXII Міжнар. наук.-практ. конф. “Інформаційно-комунікаційні технології та сталий розвиток”, 14–15 листопада 2023 р.; Юстон: Київ, Україна, 2023, 57–59. [Conference Paper]

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

ЕЕГ	Електроенцефалографія
ЕКГ	Електрокардіографія
СКГ	Сейсмокардіографія
ІМК	Інтерфейс мозок-комп'ютер
OpenBCI	Open Source Brain-Computer Interface
GUI	Графічний інтерфейс користувача
ADC	Аналого-цифровий перетворювач
Ag-AgCl electrode	Електрод з покриттям хлориду срібла
SIC	Statistical Interval Classifier
SPC	Statistical Point Classifier
TP	Істинно позитивний
TN	Істинно негативний
FP	Помилково позитивний
FN	Помилково негативний
TPR	Істинно позитивний рівень
TNR	Істинно негативний рівень
FNR	Помилково негативний рівень
FPR	Помилково позитивний рівень
PPV	Позитивно прогностичні значення
NPV	Негативно прогностичні значення
$\boldsymbol{R}$	Множина дійсних чисел
$\boldsymbol{Z}$	Множина цілих чисел
$\boldsymbol{N}$	Множина натуральних чисел
$\boldsymbol{\Omega}$	Множина елементарних подій
N	Кількість електродів (відведень) на поверхні голови оператора

M	Кількість експериментів одного типу, проведених для одного оператора
$\boldsymbol{\gamma}$	Параметр, що позначає (маркує) інваріант з групи інваріантів вектора циклічних ритмічно пов'язаних випадкових процесів
$\boldsymbol{E}\{\cdot\}$	Оператор математичного сподівання
$\boldsymbol{\Theta}_N(\omega, t)$	Вектор циклічних ритмічно пов'язаних випадкових процесів
$\xi_i(\omega, t)$	і-та компонента вектора циклічних ритмічно пов'язаних випадкових процесів
$T(t, n)$	Функція ритму вектора циклічних ритмічно пов'язаних випадкових процесів
$\boldsymbol{\Theta}_{N_\omega}(t)$	Реалізація вектора циклічних ритмічно пов'язаних випадкових процесів
$F_{k_{\xi_{i_1}\dots\xi_{i_k}}}(x_1, \dots, x_k; t_1, \dots, t_k)$	Сумісна k-вимірна функція розподілу вектора циклічних ритмічно пов'язаних випадкових процесів
$f_{k_{\xi_{i_1}\dots\xi_{i_k}}}(u_1, \dots, u_k; t_1, \dots, t_k)$	Сумісна k-вимірна характеристична функція вектора циклічних ритмічно пов'язаних випадкових процесів
$C_{p_{\xi_{i_1}\dots\xi_{i_k}}}(t_1, \dots, t_k)$	Змішана початкова моментна функція вектора циклічних ритмічно пов'язаних випадкових процесів
$R_{p_{\xi_{i_1}\dots\xi_{i_k}}}(t_1, \dots, t_k)$	Змішана центральна моментна функція вектора циклічних ритмічно пов'язаних випадкових процесів
$\boldsymbol{W}_{m,1}$	Часова область, що відповідає стану пасивності оператора в m-му циклі електроенцефалографічного сигналу

$\boldsymbol{W}_{m,2}$	Часова область, що відповідає стану активності оператора в m-му циклі електроенцефалографічного сигналу
$\boldsymbol{W}_{c_m}$	Часова область визначення m-го циклу електроенцефалографічного сигналу, що відповідає m-му експерименту, проведеному для одного оператора
$\boldsymbol{W}_{mk} = \left[\tilde{t}_{mk}, \tilde{t}_{m,k+1}\right]$	Часова область, що відповідає k-тій зоні в m-му циклі електроенцефалографічного сигналу
$\boldsymbol{W}_{m,1}$	Індикаторна функція часової області, що відповідає стану пасивності оператора в m-му циклі електроенцефалографічного сигналу
$\boldsymbol{W}_{m,2}$	Індикаторна функція часової області, що відповідає стану активності оператора в m-му циклі електроенцефалографічного сигналу

ВСТУП

Актуальність теми.

Створення високоточних неінвазивних нейроінтерфейсних систем, систем медичної діагностики функціонального стану серця людини, а також систем динамічної біометричної аутентифікації особи за її біомедичними циклічними сигналами є важливим та актуальним завданням сучасної науки та інженерії. Зокрема, створення ефективних інтерфейсів мозок-комп’ютер (ІМК), які забезпечують безпосередній зв'язок між людським мозком та комп'ютерними системами через електроенцефалографічні (ЕЕГ) та інші нейронні сигнали, стає все більш актуальним. Розвиток ІМК вимагає підвищення точності та зниження обчислювальної складності алгоритмів розпізнавання патернів та класифікації ЕЕГ сигналів, що є важливо для забезпечення точності комунікації та контролю технічних систем.

Незважаючи на значний прогрес у цій галузі, існують численні невирішені питання, які потребують подальших досліджень. Серед них — розробка більш адекватних моделей та більш точних методів для розпізнавання та класифікації біомедичних сигналів, зокрема ЕЕГ та ЕКГ сигналів, які мають складну динамічну циклічну стохастичну часову структуру. Сучасні моделі та методи часто не в змозі повною мірою враховувати стохастичність та циклічність, мінливість та спільність ритму досліджуваних біосигналів, що призводить до неточностей у функціонуванні відповідних інформаційних систем. Тому значну увагу слід приділити ритмоадаптивним методам аналізу, які можуть значно підвищити точність роботи систем моніторингу здоров'я, біометричної ідентифікації особи та нейроінтерфейсних систем. Ці методи дозволяють більш адекватно враховувати зміни ритму в досліджуваних біомедичних сигналах, що є критично важливим для забезпечення високої точності функціонування нейроінтерфейсних систем, систем медичної діагностики та біометричної аутентфікації.

Отже, вирішення цих проблем є основою для обґрунтування актуальності даної дисертаційної роботи, а саме, зумовлює необхідність розробки адекватних математичних моделей та ефективних методів опрацювання циклічних

стохастичних біомедичних сигналів зі змінним ритмом в області неінвазивних нейроінтерфейсів, кардіодіагностики та систем біометричної аутентифікації особи.

Зв'язок роботи з науковими програмами, планами, темами:

Дисертаційне дослідження пов'язане з виконанням науково-дослідної теми "Кібер-фізичне моделювання в дослідженнях медико-біологічних процесів" (№ держреєстрації 0119U000509). Розроблена дисертантом нейроінтерфейсна система включає вибір та адаптацію програмних та апаратних компонентів для ефективного взаємодії мозку з комп'ютером з використаням ЕЕГ сигналів. Використання платформи OpenBCI дозволило експериментувати з різними методами аналізу сигналів та створювати ІМК з відкритим програмним кодом, що розширило можливості для дослідження та розвитку в області біомедичних технологій.

Мета і завдання дослідження. Метою дисертаційного дослідження є розвиток математичних моделей та методів ефективного опрацювання циклічних біомедичних сигналів у сучасних неінвазивних нейроінтерфейсах, системах біометричної аутентифікації особи та системах медичної діагностики.

Для досягнення сформульованої мети в дисертаційній роботі необхідно розв'язати наступні завдання:

1) провести компаративний аналіз відомих наукових та технологічних досягнень в галузі моделювання та опрацювання циклічних біомедичних сигналів, з метою виявлення їх недоліків та задля формулювання наукового завдання дисертаційної роботи;

2) розробити нову математичну модель сукупності ЕЕГ сигналів, зареєстрованих в умовах багаторазового повторення ментальних керуючих впливів, яка б враховувала стохастичність, циклічність, мінливість та спільність ритму досліджуваних біосигналів, з метою підвищення ефективності методів опрацювання ЕЕГ сигналів в неінвазивних нейроінтерфейсних системах;

3) верифікувати нові методи опрацювання сукупності ЕЕГ сигналів для виявлення ментальних керуючих впливів в неінвазивних нейроінтерфейсних

системах, зосередившись на покращенні характеристик точності та часової обчислювальної складності;

4) розробити ритмоадаптивний метод біометричної аутентифікації особи за її ЕКГ, що б забезпечила високу точність аутентифікації;

5) обґрунтувати оптимальні (з точки зору характеристик точності та часової обчислювальної складності) вектори інформативних ознак для нейроінтерфейсних систем та систем біометричної аутентифікації особи за ЕКГ;

6) імплементувати розроблені математичні моделі та методи в систему комп'ютерних програм на мові Python, що дозволить автоматизувати процеси фільтрації завад, оцінювання ймовірнісних характеристик, проведення спектральних розкладів та класифікації циклічних біомедичних сигналів в неінвазивних нейроінтерфейсних системах, системах медичної діагностики та системах біометричної аутентифікації особи.

Об’єктом дослідження є процес побудови математичної моделі та методів ефективного опрацювання циклічних біомедичних сигналів в сучасних неівазивних нейроінтерфейсах, системах медичної діагностики та системах біометричної аутентифікації.

Предметом дослідження є математичні моделі і методи ефективного опрацювання нейроінтерфейсних та кардіодіагностичних сигналів.

Методи дослідження. Методи теорії випадкових процесів та векторів для моделювання ЕЕГ та ЕКГ сигналів, методи математичної статистики, а саме, методи статистичного точкового та інтервального оцінювання для побудови методів опрацювання біомедичних циклічних та верифікації їх математичних моделей. Методи спектрального аналізу сигналів для зменшення розмірності діагностичного простору у нейроінтерфейсних та кардіодіагностичних системах. Методи машинного навчання в задачах класифікації сигналів.

Наукова новизна отриманих результатів.

Вперше, розроблено та верифіковано нову математичну модель сукупності електроенцефалографічних (ЕЕГ) сигналів із різних відведень (електродів), зареєстрованих в умовах багаторазового повторення ментальних керуючих впливів

оператора неінвазивного нейроінтерфейсу, у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів.

Ґрунтуючись на розробленій математичній моделі сукупності ЕЕГ сигналів, зареєстрованих в умовах багаторазового повторення ментальних керуючих впливів оператора неінвазивного нейроінтерфейсу, *вперше*, обґрунтовано методи їх ритмоадаптивного статистичного опрацювання, що дало змогу сформувати множину потенційно чутливих до ментального впливу оператора нейроінерфесу інформативних характеристик ЕЕГ сигналів.

Вперше, на основі математичної моделі ЕКГ у вигляді циклічного випадкового процесу, розроблено високоефективний ритмоадаптивний метод біометричної аутентифікації особи за її ЕКГ, зокрема, за лише одним циклом ЕКГ.

Вперше, в рамках ритмоадаптивного підходу до статистичного опрацювання циклічних біомедичних сигналів, обґрунтовано оптимальні (з точки зору характеристик точності та часової обчислювальної складності) вектори інформативних ознак (перших 40 спектральних коефіцієнтів розкладу статистичних оцінок початкових моментних функцій ЕЕГ та ЕКГ сигналів у ряд Фур’є) в нейроінтерфейсних системах та в системах біометричної аутентифікації особи за ЕКГ.

Практичне значення отриманих результатів. Розроблена у дисертації нова математична модель сукупності ЕЕГ сигналів у вигляді вектора циклічних ритмічно повязаних випадкових процесів та ефективні методів статистичного опрацювання ЕЕГ та ЕКГ сигналів удосконалюють відоме математичне забезпечення сучасних неінвазивних нейроінтерфейсів, систем медичної діагностики та систем біометричної аутентифікації особи за її фізіологічними сигналами.

На основі отриманих у дисертації нових теоретичних та прикладних результатів, на мові Python розроблено програмне забезпечення для попереднього та основного (статистичного та спектрального) ритмоадаптивного опрацювання циклічних біомедичних сигналів, а також для проведення процедури класифікації із використанням типових засобів машинного навчання (k-Nearest Neighbors, Linear SVM, Decision Tree, Random Forest, Multilayer Perceptron, Adaptive

Boosting, Naive Bayes) в неінвазивних нейроінтерфейсних системах та системах біометричної аутентифікації особи за ЕКГ.

Результати дисертаційного дослідження впроваджено у навчальний процес Тернопільського національного технічного університету імені Івана Пулюя та в науково-дослідну роботу Тернопільського національного медичного університету імені І. Я. Горбачевського (Додаток Б).

Особистий внесок здобувача. Всі наукові результати дисертаційної роботи сформульовані та отримані автором самостійно. З наукових робіт, опублікованих у співавторстві, у дисертації використані результати особистих досліджень здобувача. У наведених працях (Додаток А), опублікованих із співавторами, здобувачеві належать: у [1] — проведення серії експериментальних досліджень ефективності методів ритмоадаптивного та неритмоадаптивного статистичного опрацювання ЕКГ сигналів в системах біометричної аутентифікації особи, у [2, 14] — розроблено нову математичну модель векторної ЕЕГ у вигляді вектора циклічних ритмічно пов'язаних випадкових процесів, у [3, 15] — проведено детальну оцінку ряду сучасних класифікаторів для аналізу сигналів ЕЕГ в системах інтерфейсу мозок-комп'ютер, використавши розроблену математично модель вектора циклічних ритмічно пов'язаних випадкових процесів, у [4] — проведено експериментальне дослідження із забезпеченням однакових умови експерименту та порівняно точність методів класифікації, які широко використовуються у сучасних нейроінтерфейсах, у [5] — розглянуто стратегії вибору програм та обладнання для нейроінтерфейсних систем, з акцентом на розробку власного програмного забезпечення для аналізу ЕЕГ, у [6, 9, 11] — виконано порівняльний аналіз існуючих нейроінтерфейсів, зокрема, різних цінових сегментів ринку та продукції різних виробників систем нейроінтерфейсів, а також розроблено підхід до обґрунтованого вибору методів та апаратно-програмного забезпечення при розробці систем нейроінтерфейсів у різних областях застосування, у [7] — розроблено ефективний експрес-метод біометричної аутентифікації особи на основі одного циклу ЕКГ сигналу, що відрізняється високою ефективністю аутентифікації, у [8] — проведено аналіз сучасних напрямків розвитку

нейроінтерфейсних технологій, описано їх актуальність та перспективи, розглянуто різні сфери застосування, потенційні можливості та складності, пов'язані з впровадженням та розвитком нейроінтерфейсів, у [10] — розглянуто різні інтерфейси для вводу діагностичної інформації в інформаційну систему “Імідж-Терапевт”, включаючи аудіоінтерфейс для контролю та корекції даних через голосові команди, нейроінтерфейс, що уможливлює невербальну взаємодію з системою, у [12] — розглянуто підхід до створення доступних за ціною дослідницьких нейроінтерфейсних систем з використанням платформи OpenBCI, у [13] — описано розробку принципів керування роботизованою рукою за допомогою нейроінтерфейсу, увагу зосереджено на вдосконаленні нейроінтерфейсу через прототип, який дозволяє користувачу управляти віртуальним курсором і реальною роботизованою рукою.

Апробація результатів дисертації. Апробація результатів дисертації. Наукові та практичні результати дисертаційного дослідження доповідалися та обговорювалися на міжнародних конференціях, а саме, на: Міжнародній науковій конференції "Іван Пулюй: життя в ім’я науки та України" (до 175-ліття від дня народження), Тернопіль, 2020 р.; IX Міжнародній науково-технічній конференції молодих учених та студентів, Тернопіль, 2020 р.; VIII науково-технічній конференції "Інформаційні моделі, системи та технології", Тернопіль, 2020 р.; XX Міжнародній науково-практичній конференції "Сучасні інформаційні технології управління екологічною безпекою, природокористуванням, заходами в надзвичайних ситуаціях", Київ, 2021 р.; IX Науково-технічній конференції "Інформаційні моделі, системи та технології", Тернопіль, 2021 р.; XXI Міжнародній науково-практичній конференції "Інформаційно-комунікаційні технології та сталий розвиток", Київ, 2022 р.; XXII Міжнародній науково-практичній конференції "Інформаційно-комунікаційні технології та сталий розвиток", Київ, 2023 р.

Публікації. За результатами дисертаційної роботи опубліковано 15 наукових праць, зокрема: 5 статей у закордонних наукових періодичних виданнях, які індексуються міжнародною наукометричною базою Scopus. Серед них, 2 статті, які опубліковані в міжнародних наукових журналах квертелів 1 та 2 за

наукометричними базами JCR, CiteScore та SJR. Також опубліковано 2 статті у наукових фахових періодичних виданнях України, які індексуються в Index Copernicus. 8 публікацій – це матеріали міжнародних та вітчизняних наукових конференцій.

Структура та обсяг дисертації. Дисертаційна робота містить вступ, чотири розділи, висновки, список використаних джерел із найменувань, додатків. Загальний обсяг дисертаційної роботи складає 196 сторінки, з них 135 сторінок основного тексту, де наведено 49 рисунків та 15 таблиць.

РОЗДІЛ 1
КОМПАРАТИВНИЙ АНАЛІЗ МАТЕМАТИЧНИХ МОДЕЛЕЙ ТА МЕТОДІВ ОПРАЦЮВАННЯ ЦИКЛІЧНИХ СИГНАЛІВ В НЕЙРОІНТЕРФЕЙСНИХ ТА КАРДІОДІАГНОСТИЧНИХ СИСТЕМАХ

У цьому розділі проведено компаративний аналіз доступних на ринку інфораційних технологій опрацювання ЕЕГ та ЕКГ сигналів. Особлива увага приділена аналізу існуючих математичних меделей та методів опрацювання ЕКГ та ЕЕГ сигналів з точки зору їх адекватності та ефективності використання в неівазивних нейроінтерфейсних системах, системах біометричної аутентифікації особи та медичної діагностики. Визначено обмеження та недоліки існуючих моделей та методів опрацювання ЕЕГ та ЕКГ сигналів, що дало змогу сформулювати мету дисертаційного дослідження.

Результати, отримані в рамках цього розділу, відображені у публікаціях [1, 2].

1.1. Роль інформаційних технологій в опрацюванні біосигналів циклічної просторово-часової структури

Циклічні біосигнали відіграють важливу роль у відображенні комплексних процесів, що відбуваються в живих організмах, відображаючи ритмічну діяльність систем, які підтримують їх життя. Ці сигнали, маючи циклічну просторово-часову структуру, несуть у собі важливу інформацію про стан здоров'я людини, дозволяючи ідентифікувати потенційні порушення та аномалії у функціонуванні різноманітних фізіологічних систем організму людини. Вони включають у себе широкий спектр сигналів, починаючи від серцевого ритму та дихання до більш складних процесів, таких як нейронна активність мозку та м'язові контракції. Особливістю циклічних біосигналів є те, що вони можуть варіюватися від простих до високо складних патернів, залежно від стану організму та зовнішніх впливів [3].

На рисунку 1.1 подано реєстрограми типових циклічних сигналів. Графіки ілюструють динамічну природу серцевої електричної активності разом із записами дихання та сейсмокардіографії протягом 8-секундного інтервалу.

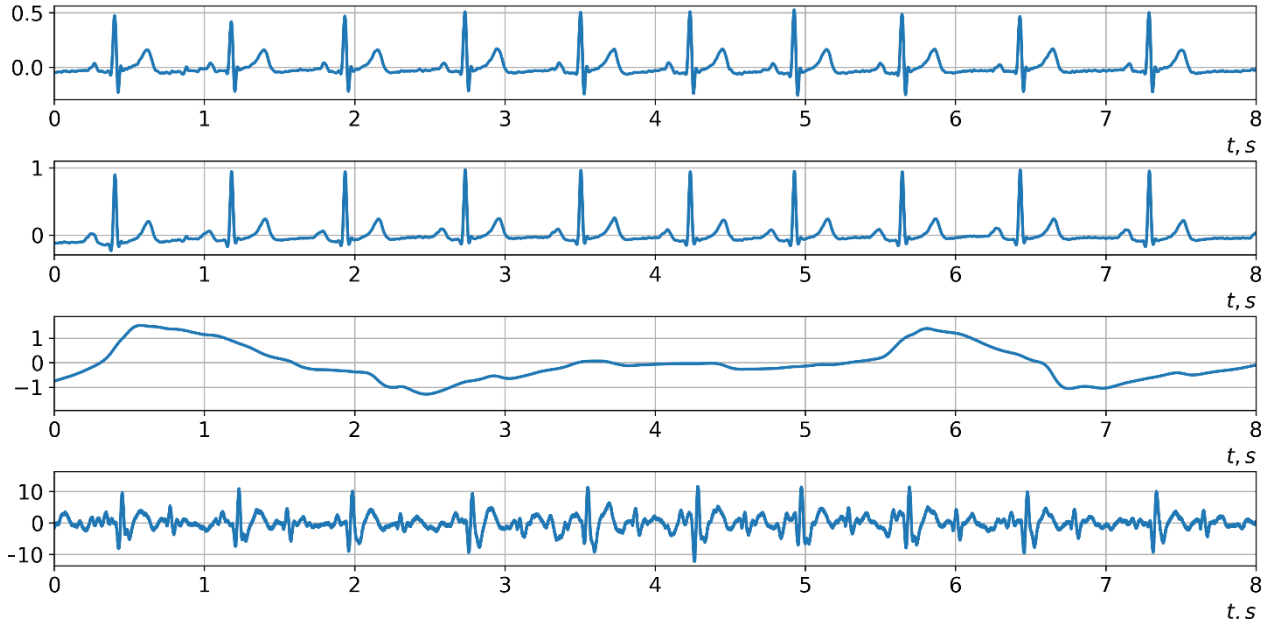


Рисунок. 1.1. Графіки реалізацій синхронно зареєстрованих електрокардіограми з відведень I та II, дихальної активності (RESP) та сейсмокардіограми (SCG)

Дослідження циклічних біосигналів за дапомогою цифрових технологій відкриває шлях для розвитку інноваційних медичних технологій і систем моніторингу здоров'я, що відкриває перспективи для своєчасного діагностування та лікування різноманітних захворювань. З іншої сторони, різноманітні циклічні біологічні сигнали, такі як ЕКГ, використовуються не тільки у медичних цілях, але й для задач аутентифікації та ідентифікації особи [4-7], що відкриває нові можливості в області безпеки та біометричної ідентифікації особи.

Інший важливий біологічний сигнал, ЕЕГ, традиційно використовується у медицині для діагностики різноманітних захворювань, зокрема епілепсії та інших неврологічних розладів. Проте, за останні десятиліття, ідея інтерфейсу мозок-комп'ютер здобула популярність [8], розширюючи сферу застосування ЕЕГ далеко за межі традиційної медицини [9-13].

Прогрес у сфері інформаційних технологій значно розширив можливості медичного моніторингу та діагностики. Сучасні інформаційні системи, що використовуються в охороні здоров'я, дозволяють зібрати, опрацювати та аналізувати великі обсяги медичних даних з високою точністю та швидкістю, що відкриває нові шляхи для своєчасного виявлення захворювань, їх прогнозування та розробки індивідуальних підходів до лікування кожного пацієнта.

Інформаційні системи в медицині, засновані на передових технологіях, таких як штучний інтелект та машинне навчання, здатні аналізувати складні біомедичні сигнали, як-от ЕЕГ або ЕКГ, виявляючи структурні нюанси, які можуть залишитися непоміченими при традиційному аналізі. Вони також сприяють розвитку телемедицини та дистанційного моніторингу стану здоров'я, забезпечуючи пацієнтам доступ до високоякісної медичної допомоги незалежно від їхнього місцеположення.

1.2. Сучасні інформаційні технології опрацювання ЕКГ та ЕЕГ сигналів

Технології аналізу ЕКГ є фундаментальним діагностичним інструментом у кардіології, що вимірює електричну активність серця за допомогою сукупності електродів, уможливлюючи діагностику стану серцево-судинної системи організму людини.

Розвиток методик збору ЕКГ у біометричних дослідженнях ознаменувалася значними досягненнями, особливо з метою підвищення прийнятності ЕКГ як біометричної ознаки. Перші дослідження в основному використовували стандартну конфігурацію з 12 відведень для розробки біометричних алгоритмів [14, 15]. З часом спостерігалася помітна зміна в бік вибіркового використання конкретних відведень, особливо відведення I та II, які віддають перевагу через більшу прийнятність завдяки розміщенню електродів на зап'ясті [16, 17]. Недавні розробки ще більше розширили ці можливості, інтегруючи збір ЕКГ у носимі технології та предмети повсякденного вжитку, прокладаючи шлях до більш універсальних та зручних для користувача

біометричних рішень [18-24], дозволяючи користувачам здійснювати моніторинг свого серцевого ритму в домашніх умовах.

Класифікація аритмій є критично важливою для сучасної медицини, оскільки правильна діагностика може значно покращити результати лікування серцевих захворювань. Аритмії, які включають неправильні серцебиття, можуть варіюватися від легких неприємностей до серйозних станів, які загрожують життю, тому їх вчасне виявлення є надзвичайно важливим. Сучасні технології дозволяють використовувати автоматизовані системи для моніторингу ЕКГ сигналів і класифікації потенційних аритмій, що забезпечує їх надійну та швидку діагностику.

Завдяки розвитку біомедичного інжинірингу та комп'ютерних технологій, класифікація аритмій стала більш точною та ефективною. Використання алгоритмів машинного навчання та статистичних методів дозволяє аналізувати великі обсяги даних ЕКГ для ідентифікації візерунків, які можуть вказувати на наявність аритмій. Це допомагає медичним фахівцям швидше реагувати на потенційні проблеми та планувати лікування, спираючись на детальний аналіз серцевої діяльності пацієнта.

ЕКГ використовується не лише для медичного моніторингу та діагностики, але й у сферах аутентифікації та ідентифікації, завдяки унікальності серцевих ритмів кожної людини. Ця особливість дозволяє використовувати ЕКГ як біометричний ідентифікатор, підвищуючи рівень безпеки в системах контролю доступу та цифрових ідентифікаційних технологіях, надаючи надійний спосіб перевірки особи без необхідності використання традиційних паролів або фізичних ключів.

Дослідження, описане у статті [16], демонструє можливість використання ЕКГ як надійного біометричного ідентифікатора шляхом аналізу даних ЕКГ для ідентифікації та аутентифікації осіб. Використовуючи дані з бази даних CYBHi, автори тестували різні алгоритми та біометричні шаблони, отримавши високу точність, підтверджує потенціал ЕКГ, який виходить за межі медичної діагностики та включає застосування в галузі кібербезпеки, такі як персональна

ідентифікація та контроль доступу до систем, що підкреслює унікальний та особистий характер ЕКГ-патернів як ефективних інструментів для біометричного розпізнавання.

На сучасному ринку існує широкий асортимент побутових систем моніторингу ЕКГ, кожна з яких пропонує унікальний набір функцій, призначених для задоволення різноманітних потреб користувачів. Від простих моделей, які забезпечують базовий моніторинг, до складних пристроїв із розширеними аналітичними можливостями.

Таблиця 1.1. Характеристики доступних популярних побутових систем моніторингу ЕКГ, які пропонує ринок

Показник	Система моніторингу ЕКГ								
	CONTEC PM10	Polar H10	Heal Force Prince 180	AliveCor Kardia Mobile Personal EKG	Facelake FL20	EMAY Portable ECG Monitor	EKGraph Portable ECG Monitor	HEACO ECG-600G	Lepu Medical TH12
Ціна, $	80	80	90	100	100	110	120	950	1300
Кількість відведень	1	1	2	1	3	1	1	12	12
Частота дискретизації, Гц	120	1К	300	300	250	120	-	1К	20К
Безпровідний зв’язок	Ні	Так	Ні	Так	Ні	Так	Так	Ні	Так
Виводить інформацію на смартфон	Ні	Так	Ні	Так	Ні	Так	Так	Ні	Так
Виводить інформацію на комп'ютер	Так	Ні	Так	Ні	Так	Так	Так	Так	Так
Зчитує ЕМГ	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні	Так
Зчитує ЕКГ	Так	Так	Так	Так	Так	Так	Так	Так	Так

У таблиці 1.1 наведено список популярних побутових систем моніторингу ЕКГ, доступних на ринку. Таблиця містить відомості про такі параметри цих систем як ціна, кількість відведень, частота дискретизації, наявність безпровідного зв'язку, можливість виводу інформації на смартфон та комп'ютер, а також здатність зчитувати ЕМГ та ЕКГ.

У найдешевших моделей (див. таблиця 1.1) побутових систем моніторингу ЕКГ, які пропонують від одного до трьох відведень, відкриваються можливості для базового щоденного моніторингу. Це дозволяє виявляти найпоширеніші порушення серцево-судинної системи на ранніх етапах, дозволяючи користувачам вчасно ідентифікувати патології та звернутися за медичними консультаціями.

Згідно таблиці бачимо, що ринок пристроїв для ЕКГ демонструє широкий діапазон цін, моделей та технічних характеристик, які варіюються від базових пристроїв з одним відведенням до більш складних систем з дванадцятьма відведеннями та високою частотою дискретизації. На нижчому кінці цінового діапазону (від $80 до $120) знаходяться пристрої від компаній, як CONTEC, Polar, Heal Force, AliveCor, Facelake, EMAY, та EKGraph, які пропонують базові функції для моніторингу ЕКГ, з якісними електродами та виведенням інформації на смартфон для деяких моделей. Ці пристрої, як правило, мають меншу кількість відведень та нижчу частоту дискретизації, але вони доступні та зручні для користувачів, які шукають базовий функціонал для моніторингу серцевої активності.

На наступній сходинці цінового діапазону ($950 та $1300) розташовані моделі від HEACO та Lepu Medical, які пропонують 12-відведенні системи з високою частотою дискретизації (до двадцяти тисяч герц у випадку Lepu Medical TH12), що робить їх ідеальними для детального діагностичного аналізу та професійного використання. Ці пристрої також підтримують бездротовий зв'язок і здатні виводити інформацію на смартфони та комп'ютери, надаючи користувачам зручні інструменти для аналізу та зберігання даних.

Загалом, вибір пристрою для ЕКГ залежить від індивідуальних потреб користувача, включаючи необхідність у високій точності, портативності, легкості використання та бюджеті. Бездротовий зв'язок та можливість виведення даних на смартфони та комп'ютери стають стандартом для багатьох сучасних пристроїв, хоча наявність додаткових функцій, таких як зчитування ЕМГ може бути вирішальною для деяких користувачів.

Нейроінтерфейс або інтерфейс мозок-комп’ютер (ІМК), представляють собою технологію, що забезпечують пряму взаємодію між людським мозком та електронними пристроями, такими як комп’ютер. Головною задачею такої системи є створення каналу зв’язку для обміну інформацією між нервовою системою людини та комп’ютером.

Таблиця 1.2. Класифікація нейроінтерфейсів за способом підключення до користувача

Характеристики	Типи нейроінтерфейсів	
	Неінвазивні	Інвазивні
Точність	Низька, оскільки електроди безпосередньо не контактують з головним мозком, а реєструють електромагнітне поле на поверхні голови	Висока, оскільки електроди знаходяться на поверхні кори головного мозку або під’єднуються безпосередньо до нейронів
Спосіб з'єднання з мозком	Електроди знаходяться на поверхні шкіри голови; з деякими типами електродів використовують спеціальний провідний гель	Електроди вживлюються безпосередньо в кору головного мозку або знаходяться на її поверхні

Нейроінтерфейси фіксують активність мозку за допомогою методів, які можуть бути як неінвазивними, так і інвазивними (див. таблиця 1.2). Неінвазивні методи включають в себе зчитування сигналів через шкіру голови без втручання в тіло, тоді як інвазивні методи передбачають моніторинг активності

безпосередньо з мозкової тканини або окремих нейронів, вимагаючи хірургічного втручання [2].

Хоча інвазивні нейроінтерфейси мають вищу точністю та швидкістю розпізнавання керуючих сигналів порівняно з неінвазивними, їх застосування має значні обмеження. Це зумовлено потребою у хірургічному втручанні кваліфікованого нейрохірурга для виконання складної операції, яка може викликати ускладнення для здоров'я пацієнта. Варто зазначити, що більшість експериментальних інвазивних нейроінтерфейсів не призначені для довготривалого використання через корозію електродів та їх обростання сполучною тканиною, що веде до зниження якості контакту або його повної втрати, а також ускладнює процедуру їх вилучення з мозку. Тому інвазивні нейронтерфейси рекомендується встановлювати лише на час проведення експерименту. У зв’язку з наведеними вище факторами, найбільшої популярності отримали нейроінтерфейси неінвазивного типу.

Неінвазивні нейроінтерфейси пропонують простішу альтернативу, дозволяючи зчитувати сигнали мозку без необхідності хірургічного втручання. Ці системи, як правило, використовують електроенцефалографію (ЕЕГ) для моніторингу електричної активності мозку через шкіру голови. Незважаючи на те, що неінвазивні методи можуть пропонувати меншу точність порівняно з інвазивними через шуми та артефакти сигналів, спричинені шаром шкіри та черепа, вони значно безпечніші для користувача та простіші в експлуатації. Останнім часом дослідження у цій галузі зосереджуються на покращенні якості сигналів [25], розробці нових методі та моделей [1] і алгоритмів машинного навчання [26], що дозволяють підвищити точність розпізнавання керуючих сигналів користувача на основі ЕЕГ.

Ринок нейроінтерфейсів представлений лише декількома компаніями, проте зі зростанням інтересу та попиту на ці технології, протягом останніх восьми років можна спостерігати значне розширення асортименту пропонованих продуктів [2]. Кожен виробник намагається виділитися на ринку, пропонуючи споживачам унікальні характеристики – це може бути кількість каналів збору

даних, стаціонарність або мобільність пристрою, спеціалізовані функції або програмне забезпечення або, звичайно, ціна (таблиця 1.3).

Таблиця 1.3. Характеристики нейроінтерфейсів, які пропонує ринок

Показник	Виробник								
	NeuroSky	Muse	Emotiv	OpenBCI	Wearable Sensing	ABM B-Alert	mBrainTrain	Brain Products ActiCHamp	BioSemi
Ціна, $	200	380	300/ 850	800	7K/ 20K	2,5K/ 3K	6,5K	28K	-
Кількість активних давачів	1	4	5/14	4/8/16	7/21	9/20	24	32	32/ 256
Програмне забезпечення для розробників	Так	Так	Так	Ні	Так	Так	Так	Так	Так
Безпровідний зв’язок	Так	Так	Так	Так	Так	Так	Так	Ні	Ні
Виводить інформацію на смартфон	Так	Так	Так	Ні	Ні	Ні	Так	Ні	Ні
Виводить інформацію на комп'ютер	Так	Так	Так	Так	Так	Так	Так	Так	Так
Зчитує ЕЕГ	Так	Так	Так	Так	Так	Так	Так	Так	Так
Зчитує ЕМГ	Так	Так	Так	Так	Так	Так	Ні	Ні	Ні
Зчитує ЕКГ	Так	Так	Ні	Так	Так	Так	Ні	Ні	Ні
Наявність акселерометра	Ні	Ні	Так	Так	Так	Так	Так	Ні	Ні
Наявність гіроскопів	Ні	Ні	Так	Ні	Так	Ні	Ні	Ні	Ні
Підтримка SD карти	Ні	Ні	Ні	Ні	Ні	Так	Так	Ні	Ні
Моніторинг рівня втоми	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні
Моніторинг рівня уваги	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні

З розвитком технологій та нарощуванням дослідницької бази, компанії, що працюють у цій сфері, почали активно інвестувати у вдосконалення існуючих моделей та створення нових [8], які б забезпечували більшу точність зчитування сигналів мозку та комфорт користувача. Окрім того, значну увагу приділяють питанням безпеки та етиці використання нейротехнологій, що, без сумніву, сприяє зростанню довіри споживачів. Це, в свою чергу, може стимулювати подальше розширення ринку нейроінтерфейсів, залучення нових інвестицій і, як наслідок, прискорення науково-технічного прогресу в даній області.

Згідно таблиці бачимо, що ринок нейроінтерфейсів представлений широким асортиментом продукції різних компаній, кожна з яких задовольняє різні сегменти споживачів за ціною та функціональністю. Ці пристрої варіюються від доступних моделей для споживачів до висококласних систем для досліджень.

На початковому рівні цінового діапазону, від $100 до $1,000, пристрої від NeuroSky та Muse примітні своїми споживацькими застосуваннями, такими як допомога в медитації та сні, мають обмежену кількість каналів і, відповідно, обмежені можливості для досліджень. Emotiv вирізняється в цьому сегменті завдяки своїм 5-ти та 14-канальним нейроінтерфейсам, які пропонують більший потенціал для ідентифікації керуючих сигналів шляхом опрацювання первинних сигналів мозкової активності. Додатково, Emotiv пропонує більш дорожчі рішення на 32 канали. Слід відзначити, OpenBCI створює інноваційні відкриті інструменти для біосенсингу та нейронауки, прагнучи подолати бар'єри для технологій інтерфейсу мозок-комп'ютер. Їх нова платформа Galea [27] інтегрує електроенцефалографію, електроміографію, електродермальну активність, фотоплетизмографію та слідкування очей у єдиний шолом, поєднуючи біометричні датчики з можливостями доповненої та віртуальної реальності для розширеного дослідження людського розуму та тіла. OpenBCI співпрацює з Varjo для розвитку нейротехнологій у просторових обчисленнях, надаючи потужні інструменти для розробників та дослідників.

Усі нейропристрої середнього діапазону цін ($1,000 - $20,000) є дослідницькими. Деякі компанії (ABM, mBrainTrain, Neuroelectrics та Wearable Sensing) пропонують бездротові рішення в цьому ціновому діапазоні, що дають змогу збирати дані з підвищеною мобільністю (і підвищеним комфортом). Крім того, можливість збору даних ЕЕГ без провідного гелю пропонують компанії ANT Neuro, Neuroelectrics та Wearable Sensing, що уможливлює скорочення часу на збір даних.

На вершині ринку, починаючи від $20,000, представлені пристрої з великою кількістю каналів, починаючи від 32 каналів, як у Brain Product's ActiCHamp, і досягаючи 160 або навіть 256 каналів, як у BioSemi. Така велика кількість каналів забезпечує високу роздільну здатність при розпізнаванні мозкових керуючих сигналів

Отже, аналіз рину нейроінтерфейсів вказує на його сегментованість, де задовольняються потреби споживачів і досліджень у всьому спектрі цінових категорій та функціональності, від базових інструментів моніторингу та самовдосконалення до передових дослідницьких та клінічних застосувань.

1.3. Аналіз відомих математичних моделей циклічних біосигналів

Ключовим аспектом розробки інформаційних систем для аналізу та моделювання циклічних сигналів є створення математичних моделей, які адекватно відтворюють їх просторово-часову структуру з огляду на специфіку дослідження та задач. Адекватність таких моделей впливає на точність і надійність методів опрацювання та опрацювання сигналів у системі. Це в свою чергу впливає на якість діагностичних, аутентифікаційних та прогностичних показників, та зумовлює достовірність ухвалених на їх основі рішень. Якість математичної моделі також визначає структурні вимоги до програмної та апаратної частин системи, що розробляється.

На рисунку 1.2 представлені ключові кроки розробки інформаційних систем для аналізу та моделювання циклічних сигналів. Починаючи з інтуїтивного феноменологічного розуміння характеристик циклічних сигналів,

які є критичними для задач дослідження, розроблювальної математичної моделі. Ця модель разом з методами та алгоритмами аналізу та моделювання формує математичну основу проектованої системи, що згадується у працях [29-41]. Це математичне забезпечення потім реалізується через відповідні програмно-апаратні рішення.

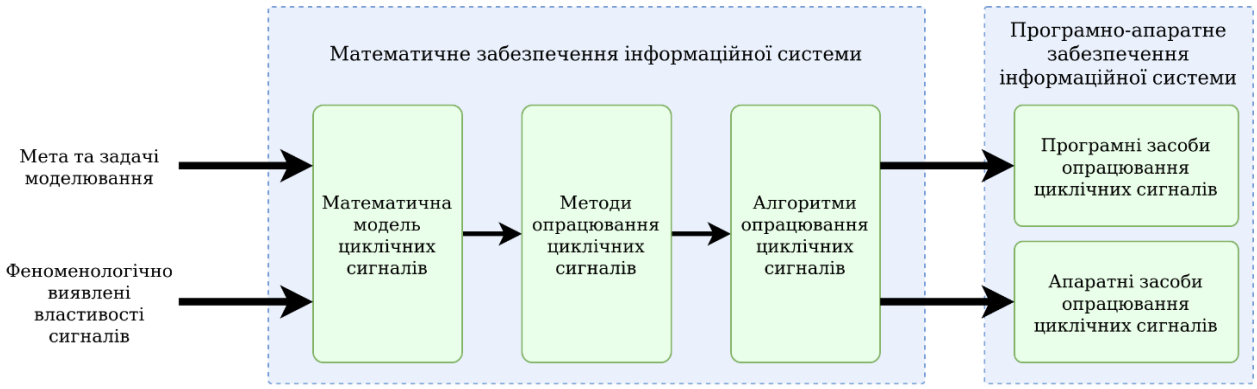


Рисунок. 1.2. Етапи розробки та створення інформаційної системи опрацювання та імітації циклічних сигналів [28]

Початковим та основним кроком проектування інформаційних систем опрацювання та імітації циклічних сигналів є створення їх математичних моделей, які адекватно відображають важливі, з точки зору дослідницьких завдань, аспекти їх просторово-часової структури. Саме математична модель значною мірою задає потенціал та ефективність створюваних інформаційних технологій, визначає структуру програмної та апаратної складових проектованої інформаційної системи. Від якості математичної моделі циклічних сигналів (див. рисунок 1.3) значно залежить точність та достовірність методів їх опрацювання та імітації в інформаційній системі, рівень інформативності та репрезентативності діагностичних, аутентифікаційних, а також достовірність прийнятих рішень.

Технічна складова діагностики за допомогою ЕЕГ та ЕКГ є ключовою, але ще більш важливим є аналіз отриманих сигналів. Розвиток та вдосконалення математичних моделей та методів опрацювання даних відіграють вирішальну

роль у точності інтерпретації результатів, що, в свою чергу, впливає на якість та ефективність діагностики.

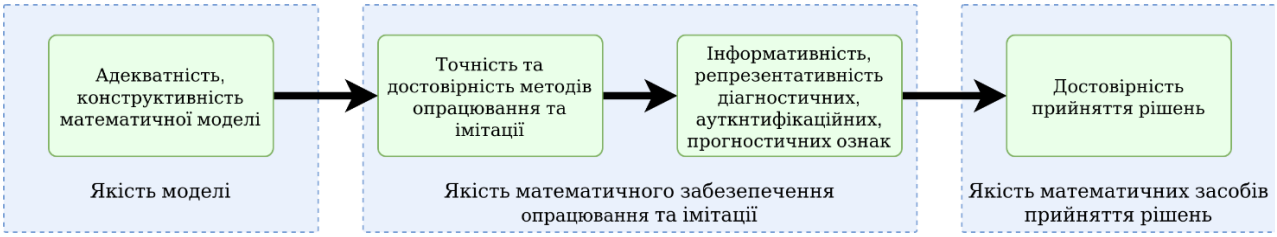


Рисунок. 1.3. Причинно-наслідкові зв’язки стосовно якості інформаційної системи опрацювання та імітації циклічних сигналів [28]

За допомогою сучасних комп'ютерних систем, заснованих на ефективних математичних моделях, медичні фахівці можуть не тільки якісно опрацьовувати та інтерпретувати циклічні біосигнали, але й розробляти персоналізовані підходи до лікування, враховуючи індивідуальні особливості кожного пацієнта.

Адаптація математичних моделей до специфіки циклічних сигналів, як-от ЕЕГ або ЕКГ, сприяє значному підвищенню точності діагностики патологій, ідентифікації та аутентифікації особи, а також розробці ефективних інтерфейсів мозок-комп'ютер. Це не тільки забезпечує високу специфічність та чутливість при виявленні патологічних змін у серцево-судинній системі через аналіз ЕКГ, але й відкриває можливості для розробки безпечних систем аутентифікації особи на основі її унікальних біометричних даних. Окрім цього, використання ЕЕГ у створенні дешевих неінвазивних інтерфейсів мозок-комп'ютер надає нові перспективи для контролю зовнішніх пристроїв безпосередньо за допомогою ментальних сигналів. Такі технології можуть радикально змінити підходи до реабілітації, надаючи людям з обмеженими можливостями засоби для взаємодії з навколишнім світом та покращення якості їхнього життя. Більше того, інтерфейси мозок-комп'ютер відкривають шляхи для їх застосування у сфері розваг та у побуті, включаючи комп'ютерні ігри, де користувачі можуть взаємодіяти з віртуальним світом безпосередньо через ментальну взаємодію. Це створює унікальні можливості для розвитку нових видів інтерактивних програм та додатків, здатних надавати користувачам більш інтуїтивні способи взаємодії з

технологіями. Таким чином, розвиток і удосконалення математичних моделей для аналізу циклічних сигналів відіграє ключову роль у прогресі медичної науки та технологій, відкриваючи широкі можливості для інновацій у діагностиці та лікуванні, сферах носимої електроніки, інтернету речей тощо.

Важливим аспектом в універсальності таких математичних моделей є здатність ефективно працювати з різними типами циклічних сигналів та забезпечувати високу точність виявлення специфічних ознак, що є ключовим для успішного впровадження цих технологій у медицині, побуті та інші сфери застосування.

Вивчення та аналіз математичних моделей, які описують циклічні явища та сигнали, є важливим кроком для розробки надійних систем опрацювання цих сигналів. Детерміновані та стохастичні моделі, з їх здатністю відтворювати ключові характеристики циклічних сигналів, забезпечують необхідну основу для проектування комплексних систем, здатних ефективно виконувати задачі діагностики, ідентифікації та взаємодії з користувачем. При цьому, велике значення має вибір відповідної моделі, заснований на аналізі та порівнянні їх ефективності в різних прикладних областях, що дозволяє оптимізувати процеси опрацювання сигналів та підвищити точність отриманих результатів.

Існує широкий спектр математичних моделей для опису циклічних процесів і сигналів. У цьому контексті, базуючись на результатах праці [28], розглянемо детальніше відомі детерміновані та стохастичні моделі циклічних сигналів, акцентуючи увагу на їхній здатності адекватно відтворювати основні аспекти циклічних сигналів.

Математичним моделям циклічних явищ та сигналів присвячено чимало наукових досліджень. Зазвичай розрізняють два основних підходи до створення математичних моделей сигналів, включаючи циклічні: конструктивний та аксіоматичний. Кожному з цих підходів відповідають різні класи математичних моделей: конструктивні моделі або моделі систем, що генерують досліджувані сигнали, та аксіоматичні моделі, які фокусуються на структурі даних (просторово-часовій структурі сигналів).

У рамках конструктивного підходу основна увага приділяється розробці та аналізу моделей не стільки самого циклічного сигналу, скільки механізму його породження. Це може включати формулювання диференційних або інтегральних рівнянь, які описують роботу коливальної системи, що генерує сигнал. Також до цього підходу відносяться теорія рядів та інтеграл Фур'є, моделі модульованих коливань, де складні коливання формуються з простих (гармонічних, періодичних), а також моделі авторегресії-ковзного середнього з періодичною або майже періодичною зміною коефіцієнтів.

Задачі, які можуть виникати при конструктивному підході, включають визначення вихідного сигналу за допомогою відомої конструкції та вхідного сигналу системи (задача аналізу), створення конструкції або структури системи (задача структурної ідентифікації), або вибір параметрів для вже відомої конструкції системи (задача параметричної ідентифікації) з метою забезпечення формування вихідного сигналу з заданими характеристиками.

Аксіоматичний підхід зосереджений на моделюванні закономірностей просторово-часової структури сигналів, тобто аксіоматичні математичні моделі безпосередньо описують просторово-часове розгортання сигналів.

Окрім конструктивних та аксіоматичних методів, у моделюванні сигналів, зокрема циклічних, виділяють детерміновані, стохастичні, нечіткі та інтервальні підходи, що базуються на різних способах формалізації невизначеностей у поведінці досліджуваних процесів. У контексті детермінованого підходу значні досягнення у математичному моделюванні та опрацюванні циклічних сигналів досягнуто в рамках періодичних, майже періодичних та квазігармонічних функцій, а також шляхом застосування різних методів спектрального (переважно грамонічного) аналізу.

У теоретико-ймовірнісному аспекті моделювання циклічних стохастичних сигналів слід виокремити спектрально-кореляційну теорією стаціонарних випадкових процесів, теорію періодично-корельованих та майже періодично-корельованих випадкових процесів. Особлива увага в задачах моделювання

Таблиця 1.4. Математичні моделі циклічних сигналів [28]

	Властивості циклічних сигналів						
	Циклічність сигналів	Багатовимірна циклічність	Довільність атрибуту циклічності	Невизначеність, неточність, випадковість сигналів	Мінливість ритму	Зміна ритму за довільним законом	Спільність ритму
Гармонічна функція	+	-	-	-	-	-	-
Періодична функція	+	-	-	-	-	-	-
Квазігармонічна функція	+	-	-	-	+	+	-
Майже періодична функція	+/-	-	-	+/-	+/-	-	-
Квазімеандр (модульований меандр)	+	-	-	-	+	+	-
Стаціонарний випадковий процес	+/-	-	-	+	-	-	-
Стохастично періодичний процес (ПКВП, ПРВП, ЛПВП тощо)	+	+	-	+	-	-	-
Періодичний випадковий вектор	+	+	-	+	-	-	+
Майже періодичний випадковий процес	+/-	+/-	-	+	+/-	-	-
Квазігармонічний випадковий процес	+/-	+/-	-	+	+/-	+/-	-
Вектор періодичних у часі з різними періодами випадкових процесів	+	+	-	+	+	-	-
Циклічна числова функція	+	-	-	-	+	+	-
Інтервальна та нечітка циклічні функції	+	-	-	+	+	+	-
Циклічний випадковий процес	+	+	-	+	+	+	-
Вектор циклічних ритмічно пов’язаних випадкових процесів	+	+	-	+	+	+	+
Абстрактна циклічна функція	+	+	+	+	+	+	+

“+” – враховує (відображає), “-” – не враховує (не відображає), “+/-” – враховує (відображає) частково

циклічних сигналів також приділяється дослідженню періодичних марковських випадкових процесів та ланцюгів, лінійних періодичних випадкових процесів, процесів з незалежними періодичними приростами та періодичними білими шумами. Відносно новими математичними моделями циклічних стохастичних сигналів є циклічний випадковий процес, вектор циклічних ритмічно пов'язаних випадкових процесів та умовний циклічний випадковий процес, які розроблено у наукових працях професора Лупенко С.А.

В таблиці 1.4 представлено класифікацію та порівняльний аналіз основних математичних моделей циклічних сигналів.

У дослідженні кардіосигналів широко використовуються моделі у вигляді вектора випадкових величин [42], стохастично періодичні випадкові процеси, такі як періодично-корельований [43, 44], періодично розподілений і лінійний періодичний випадкові процеси [45-47]. У дослідженнях [48-50] було розроблено математичну модель циклічних кардіосигналів, представлену як вектор періодичних випадкових процесів з різними періодами.

Існує значна кількість наукових досліджень, присвячених моделюванню та комп'ютерному аналізу кардіоінтервалограми та ритмокардіограми, які відображені в роботах [51-62]. Багато з цих досліджень концентруються на кардіоінтервалограмах, зареєстрованих у стані спокою. Проте особливо важливим є вивчення кардіоінтервалограм у ситуаціях, коли на серцево-судинну систему накладається фізичне навантаження, адже такі умови можуть виявляти приховані патології, невидимі при звичайному вимірюванні.

Залежно від умов реєстрації та часової структури самої кардіоінтервалограми використовуються різноманітні ймовірнісні математичні моделі. Наприклад, у стані спокою адекватною моделлю може бути стаціонарна або періодично корельована послідовність [62], тоді як під час фізичних навантажень ефективнішими є нестаціонарні послідовності перехідного типу, які є комбінацією детермінованої та стаціонарної випадкової послідовності [63-68].

У дисертаційній роботі обґрунтовуються відносно нові моделі ЕКГ, а саме, циклічний випадковий процес. Ця модель більш точно відображає морфологічну

та ритмічну структуру кардіосигналів, забезпечуючи вищу якість аналізу порівняно з традиційними моделями. Згідно із роботою [28], у таблиці 1.5. наведено порівняльний аналіз відомих математичних моделей кардіосигналів.

Таблиця 1.5. Порівняльна характеристика відомих та нових математичних моделей кардіосигналів [28]

	Відомі математичні моделі кардіосигналів (КС)									
	Детермінована функція, що описує форму одного серцевого циклу	Періодична та майже періодична функції	Вектор випадкових величин як модель реперних точок кардіоциклу	Адитивна, мультиплікативна, адитивно-мультиплікативна моделі	Періодично корельований випадковий процес	Періодично розподілений випадковий процес	Лінійний періодичний випадковий процес	Вектор лінійних періодичних у часі з різними періодаим випадкових	Циклічний випадковий процес	Вектор циклічних ритмічно пов'язаних випадкових процесів
Враховує циклічність КС	-	+	-	+	+	+	+	+	+	+
Враховує випадковий характер КС	-	-	+	+	+	+	+	+	+	+
Враховує стохастичну залежність між кардіоциклами	-	-	-	-	+	+	+	+	+	+
Описує КС в термінах функцій розподілу	-	-	+	+	-	+	+	+	+	+
Враховує мінливість ритму КС	-	-	-	-	-	-	-	+	+	+
Враховує зміну ритму КС за довільним законом	-	-	-	-	-	-	-	-	+	+
Враховує спільність ритму синхронно зареєстрованих КС	-	-	-	-	-	-	-	-	-	+

"+" – враховує (відображає), "-" – не враховує (не відображає)

Коротко розглянемо математичні моделі ЕЕГ сигналів. Стаття [69] присвячена математичним детерміністичним моделям сигналів ЕЕГ, зокрема досліджує моделі нейрофізіологічного генерування сигналів ЕЕГ на основі рівнянь Максвелла. Подібні підходи до моделювання нейрофізіологічних механізмів формування сигналів ЕЕГ розглядаються в роботі [70], яка присвячена

розв'язанню оберненої задачі у аналізі джерел ЕЕГ. Також у роботі [71] представлено детерміновані математичні моделі, що описують нейрофізіологічні механізми генерації сигналів ЕЕГ на мікро-, мезо- та макро-рівнях організації мозку людини. Кілька детерміністичних підходів до моделювання, які є основою різних алгоритмів декодування ЕЕГ та методів класифікації, розглянуто в оглядовій роботі [72].

Окрім детерміністичних математичних моделей сигналів ЕЕГ активно використовуються стохастичні математичні моделі у вигляді випадкових процесів та векторів випадкових процесів. У рамках стохастичного підходу використовуються фрактальні моделі ЕЕГ. Наприклад, у роботі [73] використовуються фрактальний гаусівський шум та фрактальний броунівський рух для розкладу сигналів ЕЕГ на мозкові ритми: дельта, тета, альфа, бета та гама ритми. Серед стохастичних математичних моделей сигналів ЕЕГ часто використовується стаціонарний випадковий процес у широкому розумінні (інформативні характеристики – автокореляційна функція та спектральна функція потужності (спектр потужності) сигналу ЕЕГ).

У роботі [74] марківський процес використовується для моделювання та комп'ютерного симулювання сигналів ЕЕГ для ідентифікації патофізіологічних змін ЕЕГ у клінічній діагностиці. У роботі [75] використовуються модель авторегресії та алгоритм Юла-Вокера для аналізу та прогнозування сигналів ЕЕГ (інформативні характеристики – коефіцієнти моделей авторегресії). Схожий підхід був розглянутий у роботі [76], де для моделювання сигналів ЕЕГ використовувались стохастичні диференціальні рівняння. У статті [77] використовуються як лінійні, так і нелінійні стохастичні математичні моделі сигналів ЕЕГ. Робота [78] присвячена лінійним моделям сигналів ЕЕГ у завданнях лінійного дискримінантного аналізу, аналізу головних компонент та незалежного компонентного аналізу. Стаціонарні лінійні випадкові процеси (інформативні характеристики – параметри ядра лінійного процесу та ймовірнісні характеристики генеруючого процесу з незалежними приростами) [79], а також умовно стаціонарні лінійні випадкові процеси [80]

використовуються для аналізу сигналів ЕЕГ, що дозволяє враховувати нейрофізіологічні процеси генерації сигналів ЕЕГ. У роботі [81] стохастичні часткові диференціальні рівняння використовуються для моделювання мезоскопічної електричної активності людської кори. Схожий підхід до моделювання також був використаний у роботі [82].

Для вивчення кореляцій між сигналами ЕЕГ, отриманими з різних електродів у контексті проблеми торгового агента, використовується модель сигналів ЕЕГ у формі вектора стаціонарних і зв'язаних стаціонарних випадкових процесів [83]. У цій моделі як інформативні характеристики використовуються автокореляційні функції, функції взаємокореляцій та спектральні функції потужності компонентів вектора ЕЕГ. Для врахування нестаціонарності, а саме, кусково стаціонарної природи часової структури ЕЕГ, як модель сигналу ЕЕГ використовується кусково стаціонарний випадковий процес (інформативні характеристики – автокореляційні функції та спектральні функції потужності сигналу ЕЕГ у різні часові інтервали його реєстрації). Зокрема, у [84] кусково стаціонарний випадковий процес використовується як модель сигналу ЕЕГ для його адаптивної сегментації. Також у [85] ця математична модель використовується для вирішення проблеми спектрального аналізу сигналу ЕЕГ на коротких часових інтервалах.

У роботі [86] обґрунтовується використання циклостаціонарного корельованого (періодично корельованого) випадкового процесу як математичної моделі сигналів ЕЕГ для використання в дизайні інтерфейсу мозок-комп'ютер. У рамках цієї математичної моделі як інформативні характеристики використовуються моментні функції першого та другого порядку, біспектральна функція потужності сигналів ЕЕГ. Досить часто для оцінки ймовірнісних характеристик сигналів ЕЕГ використовуються методи усереднення за ансамблем реалізацій для нестаціонарних випадкових процесів з невизначеною ймовірнісною структурою. Наприклад, подання сигналу ЕЕГ через ансамбль його реалізацій здійснюється в [87], де пропонується використовувати результати інтеграції сигналів ЕЕГ під час кожної секунди як інформативні характеристики.

У таблиці 1.6 здійснено порівняння математичних моделей ЕЕГ сигналів, зокрема, розробленої в дисертації математичної моделі векторного ЕЕГ сигналу у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів.

Таблиця 1.6. Порівняльна характеристика відомих і нової математичних моделей ЕЕГ [1]

	Відомі математичні моделі ЕЕГ							**Нова модель**
	Детерміновані моделі	Стаціонарний випадковий процес у широкому сенсі	Стаціонарна послідовність авторегресії та ковзного середнього	Стаціонарний лінійний випадковий процес	Кусково-стаціонарний випадковий процес	Вектор стаціонарних та стаціонарно пов'язаних випадкових процесів	Періодично корельований випадковий процес	Вектор циклічних ритмічно пов’язаних випадкових процесів
Враховує випадкову природу ЕЕГ	-	+	+	+	+	+	+	+
Враховує циклічну структуру ймовірнісних характеристик ЕЕГ	-	-	-	-	-	-	+	+
Враховує сумісні ймовірнісні характеристики різних компонентів у векторній ЕЕГ	-	-	-	-	-	+	-	+
Враховує нерегулярність (змінність) ритму векторної ЕЕГ	-	-	-	-	+	-	-	+
Враховує спільність ритмічної структури для всіх компонентів векторної ЕЕГ	-	-	-	-	-	-	-	+
Враховує наявність зональної структури циклу векторної ЕЕГ	+	-	-	-	+	-	-	+
Враховує функції моментів вищого порядку векторної ЕЕГ	-	-	-	+	-	+	-	+
Враховує стохастичну залежність між різними циклами векторної ЕЕГ	-	-	-	-	-	-	-	+

"+" - враховує (відображає), "-" - не враховує (не відображає)

1.4. Аналіз відомих методів опрацювання циклічних біосигналів

У сучасних системах аналізу сигналів серця методи опрацювання ЕКГ поділяються на методи попереднього опрацювання та методи основного

опрацювання. Попереднє опрацювання ЕКГ включає у себе фільтрацію та процедуру вилучення тренду із кардіосигналів. Фільтрація проводиться з метою усунення завад у зареєстрованих реалізаціях кардіосигналів, які зумовлені дією шумів біологічного та технічного походження, а також дією завад, які привнесені зовні. Процедура вилучення тренду усуває або суттєво послаблює тренди, які викликані різними причинами: дрейфом нуля вимірювальної апаратури, операціями попередньої фільтрації кардіосигналу, факторами біологічного походження. Серед відомих методів фільтрації та вилучення тренду із кардіосигналів можна назвати методи фільтрації за допомогою рекурсивних цифрових фільтрів Баттерворта та Бесселя четвертого порядку, методи вилучення тренду за методом найменших квадратів [88]).

Методи основного опрацювання кардіосигналів поділяють на аналіз серцевого ритму та морфологічний аналіз. Ці методи включають перетворення Фур'є [89-92], wavelet-аналіз [93], фільтрові методи [94], метод фазових просторів [95-98]. Таке різноманіття підходів дозволяє глибше розуміти структуру та динаміку кардіосигналів, особливо під час діагностики та лікування серцевих захворювань.

Аналіз часових рядів у контексті ЕКГ діагностики слугує важливим інструментом для оцінювання серцевої активності пацієнта на протязі часу. Часовий аналіз є фундаментальним методом, який використовується для вивчення та інтерпретації характеристик серцевої діяльності на основі електричних сигналів, генерованих серцем. Цей підхід дозволяє детально аналізувати часові параметри сигналу ЕКГ, такі як тривалість хвиль, інтервалів і періодів, що мають вирішальне значення для діагностики та оцінки серцевого здоров'я. Часовий аналіз ЕКГ зосереджується на вимірюванні та аналізі інтервалів між окремими компонентами сигналу ЕКГ, такими як P-хвиля, QRS-комплекс, і T-хвиля. Ці компоненти відображають різні фази серцевого циклу, включаючи деполяризацію передсердь, деполяризацію шлуночків, та їх реполяризацію. Вимірювання, такі як тривалість QRS-комплексу, інтервал QT, та інтервал PR, надають інформацію про функціональний стан провідної системи

серця та можливі порушення. Слід зазначити, що часовий статистичний аналіз, аналіз головних компонент, методи машинного навчання, алгоритми детекції R-хвиль, та методи оцінювання варіабельності серцевого ритму є основною групою методів методів опрацювання ЕКГ сигналів.

У рамках детерміністичного підходу до моделювання сигналів ЕЕГ (сигнали ЕЕГ інтерпретуються як детерміністичні функції) використовуються методи їх гармонічного аналізу, а також методи спектрального аналізу в негармонічних базах. Наприклад, у статті [99] представлено новий метод просторового гармонічного аналізу даних ЕЕГ з використанням власного простору лапласіана сітчастої поверхні позицій електродів. У статті [100] сигнали ЕЕГ досліджуються за допомогою аналізу вейвлетів з метою видобутку ознак та класифікації за допомогою штучної нейронної мережі та машини опорних векторів. У статті [101] для виявлення часово-локалізованих подій у структурі сигналів ЕЕГ також використовується аналіз вейвлетів. У статті [102] запропоновано ентропію субсмуги вейвлетів та її часову різницю як два кількісні показники для аналізу та сегментації сигналів ЕЕГ.

У статті [103] використовується мультифрактальний аналіз ЕЕГ для завдання розпізнавання емоцій. Наприклад, у роботі [104] як інформативна характеристика сигналів ЕЕГ використовується одновимірний спектр потужності, що відповідає моделі сигналів ЕЕГ у формі стаціонарного випадкового процесу.

Якщо для ЕКГ сигналів як їх математичну модель цілком слушно використовувати відомий циклічний випадковий процес та методи ритмоадаптивного статистичного оцінювання характеристик ЕКГ сигналів, то для моделювання ЕЕГ сигналів в нейроінтерфейсних системах необхідно побудувати нову математичну модель та обґрунтувати нові методи аналізу даних . Це зумовлено тим, що незважаючи на значну кількість математичних моделей сигналів ЕЕГ, ці моделі, за винятком циклостаціонарного корельованого випадкового процесу, не враховують циклічну структуру сигналів ЕЕГ, зареєстрованих у умовах багаторазових повторень ментальних керуючих впливів

оператора ІМК на етапі навчання класифікатора, а циклостаціонарний корельований випадковий процес не дозволяє врахувати моментні функції вищого порядку, не враховує варіативність ритму характеристик досліджуваних сигналів і не має формальних засобів відображення взаємозв'язків між компонентами векторного сигналу ЕЕГ. Використання відомих методів оцінювання для ансамблю записаних циклів сигналів ЕЕГ не дозволяє врахувати взаємозв'язки між характеристиками різних циклів сигналу ЕЕГ у його багатоцикловій реалізації. Також варто відзначити, що більшість методів оцінювання інформативних характеристик сигналів ЕЕГ є методами оцінювання моментних функцій сигналів ЕЕГ в рамках лише спектрально-кореляційної теорії випадкових процесів, а не теорії випадкових процесів у рамках багатовимірних функцій розподілу та моментних функцій вищого порядку.

Зазначені вище недоліки можуть бути можливими причинами низької точності виявлення та класифікації керуючих ментальних впливів оператора ІМК, що вказує на необхідність удосконалення математичних моделей та методів оцінки інформативних характеристик сигналів ЕЕГ в системах ІМК.

У цьому контексті для підвищення показників точності та зниження обчислювальної складності алгоритмів функціонування такого класу систем необхідно провести комплексне дослідження структури та характеристик векторного електроенцефалографічного сигналу з метою селекції інформативних ознак в векторному ЕЕГ-сигналі для конкретних операторів. А також здійснити побудову адекватних критеріїв та методів ефективного опрацювання ЕКГ та ЕЕГ сигналів в системах ІМК, системах біометричної аутентифікації особи та в системах медичної діагностики.

Наведений вище аналіз відомих математичних моделей та методів опарцювання циклічних біосигналів дав змогу сформулювати мету дисертаційного дослідження, якою є розвиток математичних моделей та методів ефективного опрацювання циклічних біомедичних сигналів в сучасних неінвазивних нейроінтерфейсах, системах біометричної аутентифікації особи та системах медичної діагностики.

1.5 Висновки до першого розділу

У даному розділі проведено компаративний аналіз існуючих математичних моделей та методів опрацювання ЕКГ та ЕЕГ сигналів. Здійснено порівняльний огляд наявних технологічних рішень, які застосовуються на ринку для збору та обробки біомедичних сигналів. Визначено основні переваги та недоліки цих методів, що допомогло виявити критичні аспекти, які потребують подальшого вдосконалення.

На базі проведеного аналізу обґрунтовано необхідність розробки нової математичної моделі векторного ЕЕГ сигналу в нейроінтерфейсних системах. Ця модель повина забезпечувати комплексне врахування стохастичності, циклічності, мінливості та спільності ритмів досліджуваних біосигналів. Завдяки цьому, модель повинна демонструвати високу точність та ефективність в задачах опрацювання сигналів, зокрема у неінвазивних нейроінтерфейсних системах.

Обґрунтовано необхідність удосконалення методів ефективного опрацювання ЕКГ та ЕЕГ сигналів в неінвазивних нейроінтерфейсних системах, системах біометричної аутентифікації особи та в системах медичної діагностики.

РОЗДІЛ 2
ЙМОВІРНІСНА МАТЕМАТИЧНА МОДЕЛЬ ТА СТАТИСТИЧНІ МЕТОДИ ОЦІНЮВАННЯ МОМЕНТНИХ ФУНКЦІЙ ВЕКТОРНОЇ ЕЕГ В НЕЙРОІНТЕРФЕЙСНИХ СИСТЕМАХ

У другому розділі дисертаційної роботи проводиться аналіз особливостей часової структури векторного ЕЕГ сигналу як носія відомостей про ментальну активність оператора неінвазивного нейроінтерфейсу, а також проводиться аналіз вимог до математичної моделі ЕЕГ сигналів. Представлено нову математичну модель векторного ЕЕГ сигналу у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів та наведено основі властивості ймовірнісних характеристик цієх моделі. Обгрунтовано методи статистичного аналізу векторного ЕЕГ сигналу.

Основні результати розділу опубліковано в працях [1, 105].

2.1. Критерії та вимоги до математичної моделі сигналів ЕЕГ в нейроінтерфейсних системах

В результаті багатьох неконтрольованих факторів природного та штучного походження, а також враховуючи результати багатьох наукових робіт, присвячених моделюванню та аналізу сигналів ЕЕГ, доцільно використовувати стохастичний підхід до побудови математичної моделі ЕЕГ. Враховуючи, що реєстрація ЕЕГ буде проводитися з L ($L \geq 2$) різних електродів (відведень) на поверхні голови оператора, а саме, в експериментах буде отримано L синхронно зареєстрованих ЕЕГ (таку сукупність L сигналів ЕЕГ ми назвемо векторним ЕЕГ), то є розумним припустити, що загальна математична модель векторного ЕЕГ буде L-вимірним вектором випадкових процесів.

Для навчання системи виявлення (класифікації) контролюючого ментального впливу оператора ІМК (інтерфейсу мозок-комп'ютер) на структуру векторної ЕЕГ необхідно провести серію з M ідентичних експериментів. Під серією тренувальних експериментів розуміється низка ментальних дій оператора

ІМК, які проявляються у змінах тимчасової структури векторної ЕЕГ і які є основою для навчання програмної підсистеми розпізнавати саме цю дію ментального впливу оператора на об'єкт маніпуляції шляхом аналізу векторної ЕЕГ. Крім того, таку багатоциклову векторну ЕЕГ можна використовувати як первинну статистику для вивчення закономірностей навчання та ефективності такого навчання операторів ІМК.

Якщо в результаті ментальної дії оператора ІМК відбувається зміна тимчасової (або спектральної) структури векторної ЕЕГ, яка є значущою для вирішення проблем виявлення (класифікації), то у результаті проведення серії з M таких послідовних за часом експериментів одного типу (ментальні впливи того самого оператора в однакових умовах) є розумним припустити про певну циклічність, повторюваність часової структури векторної ЕЕГ, яка відображає відповідну контрольну інформацію в системі ІМК. При відсутності такої серії послідовних ментальних впливів оператора ІМК, циклічна структура векторної ЕЕГ буде відсутня. Оскільки доцільність стохастичного підходу до математичного моделювання сигналів ЕЕГ була сформульована вище, така циклічна повторюваність повинна бути у ймовірнісній структурі (ймовірнісних характеристиках) досліджуваної векторної ЕЕГ.

Наявність циклічної структури в ймовірнісних характеристиках векторної ЕЕГ, яка відображає електричну активність мозку оператора ІМК в серії послідовних однакових експериментів, є основою для розробки методів і засобів виявлення та класифікації контрольної інформації в системах ІМК. Враховуючи вищевикладене, є розумним вимагати, щоб математична модель сигналів ЕЕГ відображала цю циклічну повторюваність в ймовірнісних характеристиках.

Також, враховуючи, що кожен цикл векторної ЕЕГ можна поділити на зону активності (коли оператор виконує ментальну контролюючу дію) та пасивності (коли ментальна контролююча дія оператора відсутня), математична модель має враховувати цю зонну структуру у гармонії з її циклічною структурою.

Наявність циклічності в певних характеристиках векторної ЕЕГ є основою самої можливості вирішення завдання виявлення контролюючих сигналів

оператора ІМК, оскільки лише в цьому випадку нестаціонарна структура сигналів ЕЕГ підходить для ефективного розрізнення областей, які відповідають зонам активності та пасивності оператора ІМК. Циклічна природа характеристик векторної ЕЕГ є необхідною умовою для успішного вирішення складнішого завдання, ніж виявлення контролюючого сигналу оператора ІМК, а саме завдання розпізнавання типу контролюючого сигналу оператора ІМК, оскільки перед класифікацією контролюючого сигналу його потрібно виявити та відрізнити від потоку даних ЕЕГ.

Тривалості зон активності та пасивності в різних циклах векторної ЕЕГ, а отже, і загальна тривалість її різних циклів є різною, тому математична модель повинна враховувати цю часову мінливість. Враховуючи той факт, що векторна ЕЕГ є упорядкованою множиною сигналів ЕЕГ з різних електродів, які відображають електричну активність різних областей мозку однієї людини (оператора), і часові інтервали, протягом яких оператор здійснює свою ментальну дію для всіх компонентів векторної ЕЕГ будуть однаковими, то є розумним постулювати ритмічний зв'язок всіх компонентів векторної ЕЕГ.

Також важливо, щоб математична модель сигналів ЕЕГ була придатною для забезпечення інтеграції статистичної інформації з ЕЕГ, отриманої з різних сенсорів, зокрема, щоб вона надавала можливість проведення спільного статистичного аналізу сигналів ЕЕГ з різних областей мозку оператора ІМК. Зокрема, крім структурної схожості сигналів ЕЕГ та їх математичної моделі, важливою властивістю такої моделі є її здатність описувати найширший можливий клас характеристик вивчених сигналів, з метою ідентифікації тих, що найбільш чутливі до впливів оператора ІМК.

2.2. Математична модель сигналів ЕЕГ в системах ІМК у формі вектора циклічних ритмічно пов’язаних випадкових процесів із двома зонами на циклах

Враховуючи вищевикладені вимоги до математичної моделі сигналів ЕЕГ, як адекватну математичну модель сигналів ЕЕГ доцільно використовувати вектор

циклічних ритмічно пов'язаних випадкових процесів. Згідно з роботами [1, 28, 105] визначимо вектор циклічних ритмічно пов'язаних випадкових процесів.

Визначення 1. Вектор $\Theta_N(\omega,t)$ випадкових процесів $\{\xi_i(\omega,t), i=\overline{1,N}, \omega \in \Omega, t \in R\}$ називається вектором циклічних ритмічно пов'язаних випадкових процесів (а самі процеси називаються циклічними ритмічно пов'язаними випадковими процесами), якщо існує така функція $T(t,n), t \in R, n \in Z$, яка задовольняє умови (1) - (3) функції ритму і для будь-якого $t_1,\dots,t_k$ з множини роздільності вектора $\Theta_N(\omega,t)$ k-вимірних випадкових векторів $\{\xi_{i_1}(\omega,t_1),\xi_{i_2}(\omega,t_2),\dots,\xi_{i_k}(\omega,t_k)\}$ та $\{\xi_{i_1}(\omega,t_1+T(t_1,n)),\xi_{i_2}(\omega,t_2+T(t_2,n)),\dots,\xi_{i_k}(\omega,t_k+T(t_k,n))\}$ $n \in Z, i_1,\dots,i_k=\overline{1,N}$ є стохастично еквівалентними в широкому розумінні для всіх $n \in Z$ і для всіх $k \in N$.

Функція [62] $T(t,n), t \in R, n \in Z$ називається функцією ритму, якщо вона має такі властивості:

1) $$\begin{cases} T(t,n) > 0 (T(t,1) < \infty), t \in R, if n > 0, \\ T(t,n) = 0, t \in R, if n = 0, \\ T(t,n) < 0, t \in R, if n < 0; \end{cases} \quad (1)$$

2) для будь-яких $t_1 \in R$ і $t_2 \in R$, для яких $t_1 < t_2$, і для функції $T(t,n)$ виконується строга нерівність:

$$T(t_1,n)+t_1 < T(t_2,n)+t_2, \forall n \in Z \quad (2)$$

3) функція $T(t,n)$ є найменшою за модулем $\left(|T(t,n)| \le |T_\gamma(t,n)|\right)$ серед усіх таких функцій $\{T_\gamma(t,n), \gamma \in N\}$, які задовольняють (1) і (2), а саме:

$$|T(t,n)| = min_{\gamma \in N}\{|T_\gamma(t,n)|, \gamma \in N\}, t \in R, n \in Z. \quad (3)$$

Зазначимо, що стохастично еквівалентними у широкому розумінні будуть k-вимірні випадкові вектори, якщо вони мають однакові k-вимірні функції розподілу.

У випадку, коли вектор $\Theta_N(\omega,t)$ містить лише один компонент ($N=1$), тоді такий компонент є циклічним випадковим процесом $\xi(\omega,t), \omega \in \Omega, t \in R$ [106]. У частковому випадку, коли $T(t,n) = n \cdot T, T = const, T > 0$, вектор циклічних ритмічно пов'язаних випадкових процесів є вектором T-періодичних

і T-періодично пов'язаних випадкових процесів (T-періодичний випадковий вектор). Функція ритму $T(t,n), t \in R, n \in Z$ визначає закон зміни часових інтервалів між однофазними значеннями вектора циклічних ритмічно пов'язаних стохастичних процесів. Ритм циклічного сигналу в кваліативних термінах може бути регулярним (стабільним, незмінним) або нерегулярним (змінним, нестабільним). З точки зору концепції функції ритму, вектор періодичних і періодично пов'язаних випадкових процесів є вектором циклічних ритмічно пов'язаних випадкових процесів з регулярним (стабільним) ритмом, а точніше з функцією ритму $T(t,n) = n \cdot T, T = const > 0$. Вектор циклічних сигналів з нерегулярним ритмом (не ритмічні циклічні сигнали, сигнали зі змінним ритмом) є сигнали, модель яких є вектором циклічних ритмічно пов'язаних випадкових процесів з функцією ритму $T(t,n) \neq n \cdot T$ $(T(t,1) \neq const)$. Такий вектор циклічних ритмічно пов'язаних випадкових процесів називається вектором циклічних ритмічно пов'язаних випадкових процесів з нерегулярним (змінним) ритмом. Вектор циклічних ритмічно пов'язаних випадкових процесів та циклічний випадковий процес широко використовуються як математичні моделі циклічних сигналів у медицині та техніці [107-109]. Для цих математичних моделей та методів обробки сигналів була розроблена аксіоматично-дедуктивна стратегія організації знань та інтелектуалізовані інформаційні технології їх онтологічного представлення в орієнтованих на онто-експертні системи підтримки прийняття модельних рішень [110-114].

Оскільки тривалість виконання оператором відповідного завдання в кожному експерименті є різною, відповідні тривалості циклів вектора циклічних ритмічно пов'язаних випадкових процесів також різні. Цей факт вказує на правильність припущення, що функція ритму вектора циклічних ритмічно пов'язаних випадкових процесів не буде задовольняти умові $T(t,n) = n \cdot T$. Таким чином, враховуючи вищезазначені міркування, є розумним припустити, що адекватною математичною моделлю векторної ЕЕГ буде вектор циклічних ритмічно пов'язаних випадкових процесів з нерегулярним ритмом.

Розглянемо властивості деяких ймовірнісних характеристик вектора $\Theta_N(\omega,t)$ циклічних ритмічно пов'язаних випадкових процесів. Так, для його сумісної k-вимірної функції розподілу $F_{k_{\xi_{i_1}\dots\xi_{i_k}}}(x_1,\dots,x_k;t_1,\dots,t_k)$ існує рівність:

$$F_{k_{\xi_{i_1}\dots\xi_{i_k}}}(x_1,\dots,x_k;t_1,\dots,t_k) =$$
$$= F_{k_{\xi_{i_1}\dots\xi_{i_k}}}\big(x_1,\dots,x_k;t_1+T(t_1,n),\dots,t_k+T(t_k,n)\big), \quad (4)$$
$$i_1,\dots,i_k=\overline{1,N},t_1,\dots,t_k\in R,n\in Z,k\in N.$$

Якщо існує сумісна функція розподілу густини вектора $\Theta_N(\omega,t)$, то для неї існує рівність, аналогічна рівності (4).

Сумісна k-вимірна характеристична функція $f_{k_{\xi_{i_1}\dots\xi_{i_k}}}(u_1,\dots,u_k;t_1,\dots,t_k)$ вектора $\Theta_N(\omega,t)$ задовольняє рівність:

$$f_{k_{\xi_{i_1}\dots\xi_{i_k}}}(u_1,\dots,u_k;t_1,\dots,t_k)=f_{k_{\xi_{i_1}\dots\xi_{i_k}}}\big(u_1,\dots,u_k;y(t_1,n),\dots,y(t_k,n)\big)=$$
$$f_{k_{\xi_{i_1}\dots\xi_{i_k}}}\big(u_1,\dots,u_k;t_1+T(t_1,n),\dots,t_k+T(t_k,n)\big), \quad (5)$$
$$i_1,\dots,i_k=\overline{1,N},t_1,\dots,t_k\in R,n\in Z,k\in N.$$

Функція змішаного початкового моменту порядку $p=\sum_{j=1}^{k}r_j$ вектора $\Theta_N(\omega,t)$ задовольняє рівність:

$$C_{p_{\xi_{i_1}\dots\xi_{i_k}}}(t_1,\dots,t_k)=E\left\{\xi_{i_1}^{r_1}(\omega,t_1)\cdot\dots\cdot\xi_{i_k}^{r_k}(\omega,t_k)\right\}=$$
$$C_{p_{\xi_{i_1}\dots\xi_{i_k}}}\big(t_1+T(t_1,n),\dots,t_k+T(t_k,n)\big), \quad (6)$$
$$t_1,\dots,t_k\in R,i_1,\dots,i_k=\overline{1,N},n\in Z,k\in N.$$

Функція змішаного центрального моменту порядку $p=\sum_{j=1}^{k}r_j$ вектора $\Theta_N(\omega,t)$ задовольняє рівність:

$$R_{p_{\xi_{i_1}\dots\xi_{i_k}}}(t_1,\dots,t_k)=$$
$$E\left\{\left(\xi_{i_1}(\omega,t_1)-m_{\xi_{i_1}}(t_1)\right)^{r_1}\cdot\dots\cdot\left(\xi_{i_k}(\omega,t_k)-m_{\xi_{i_k}}(t_k)\right)^{r_k}\right\}= \quad (7)$$
$$R_{p_{\xi_{i_1}\dots\xi_{i_k}}}\big(t_1+T(t_1,n),\dots,t_k+T(t_k,n)\big),$$
$$t_1,\dots,t_k\in R,i_1,\dots,i_k=\overline{1,N},n\in Z,k\in N.$$

Вищезазначені ймовірнісні характеристики вектора циклічних ритмічно пов'язаних випадкових процесів є інформативними, проте занадто обчислювально складними. Тому, для потреб статистичної обробки векторної ЕЕГ, доцільно використовувати менш інформативні, але набагато обчислювально простіші ймовірнісні характеристики, такі як вектор математичних сподівань та матриця кореляційних функцій вектора $\Theta_N(\omega,t)$.

Враховуючи, що кожен цикл векторної ЕЕГ можна поділити на зону активності (коли оператор виконує ментальну контрольну дію) та зону пасивності (коли ментальна контрольна дія оператора відсутня), кожен компонент $\xi_i(\omega,t)$ вектора $\Theta_N(\omega,t)$ можна представити наступним чином:

$$\xi_i(\omega,t)=\sum_{m\in Z}\sum_{k=1}^{2}\tilde{\xi}_{i,m,k}(\omega,t)=\sum_{m\in Z}\left(\tilde{\xi}_{i,m,1}(\omega,t)+\tilde{\xi}_{i,m,2}(\omega,t)\right), \quad \omega\in\Omega, t\in R, i=\overline{1,N}, \tag{8}$$

де випадковий процес $\tilde{\xi}_{i,m,1}(\omega,t)$ збігається (є ідентичним) з випадковим процесом $\xi_i(\omega,t)$ на ділянці $W_{m,1}$, що відповідає стану пасивності оператора ІМК у m-му циклі циклічного випадкового процесу $\xi_i(\omega,t)$ і які є рівними:

$$\tilde{\xi}_{i,m,1}(\omega,t)=\xi_i(\omega,t)\cdot I_{W_{m,1}}(t), \omega\in\Omega, t\in R, i=\overline{1,N}, m\in Z. \tag{9}$$

Випадковий процес $\tilde{\xi}_{i,m,2}(\omega,t)$ збігається (є ідентичним) з випадковим процесом $\xi_i(\omega,t)$ на ділянці $W_{m,2}$, що відповідає стану активності оператора ІМК у m-му циклі циклічного випадкового процесу $\xi_i(\omega,t)$ і які є рівними:

$$\tilde{\xi}_{i,m,2}(\omega,t)=\xi_i(\omega,t)\cdot I_{W_{m,1}}(t), \omega\in\Omega, t\in R, i=\overline{1,N}, m\in Z. \tag{10}$$

Функції $I_{W_{m,1}}(t)$ та $I_{W_{m,1}}(t)$ є індикаторними функціями:

$$I_{W_{m,1}}(t)=\begin{cases}1, t\in W_{m,1},\\ 0, t\notin W_{m,1},\end{cases} \tag{11}$$

$$I_{W_{m,2}}(t)=\begin{cases}1, t\in W_{m,2},\\ 0, t\notin W_{m,2}.\end{cases} \tag{12}$$

Об'єднання множин $W_{m,1}$ та $W_{m,2}$ є областю визначення m-го циклу циклічного випадкового процесу $\xi_i(\omega,t)$, а саме

$$W_{c_m}=W_{m,1}\cup W_{m,2}, m\in Z \tag{13}$$

Кожен m-й цикл у часовому діапазоні W_{c_m} відповідає m-му тренувальному експерименту ($m = \overline{1, M}$).

Застосування вектора циклічних ритмічно пов’язаних випадкових процесів як моделі взаємопов'язаних електроенцефалографічних сигналів з однаковою ритмічною структурою має низку важливих переваг (див. Таблиця 1.6), а саме модель враховує:

1) стохастичність векторної ЕЕГ;

2) циклічну структуру ймовірнісних характеристик векторної ЕЕГ;

3) сумісні ймовірнісні характеристики різних компонентів у векторній ЕЕГ;

4) нерегулярність (змінність) ритму осциляторного процесу;

5) спільність ритмічної структури для всіх компонентів векторної ЕЕГ.

6) стохастичну залежність між різними циклами векторної ЕЕГ, що виникають в різних експериментах з тренування оператора.

7) наявність двох зон (сегментів) у часовій структурі кожного сигналу ЕЕГ, які відповідають станам активності та пасивності оператора ІМК.

Як ми можемо бачити, нова модель має ряд переваг порівняно з відомими, оскільки одночасно враховує стохастичність, циклічність, змінність та спільність ритму компонентів векторної ЕЕГ. Модель має ефективні статистичні засоби для дослідження широкого класу її характеристик та дозволяє враховувати та інтегрувати дані, отримані з різних областей поверхні голови оператора ІМК. Тому вона була насправді взята за основу математичного моделювання векторної ЕЕГ, оскільки ця модель більш повно враховує особливості просторово-часової структури ЕЕГ у порівнянні з відомими математичними моделями.

Врахування у математичній моделі спільності ритмічної структури компонентів векторної ЕЕГ, крім ясного формального відображення факту такої спільності, дозволяє підвищити точність та надійність виявлення контрольної інформації (контрольних сигналів), генерованих мозком оператора під час його свідомих ментальних зусиль, оскільки можливе врахування усередненої

контрольної інформації, отриманої з різних компонентів векторної ЕЕГ. Завдяки розгляду стохастичної залежності між різними циклами векторної ЕЕГ та через урахування інформації про закономірності змін ритму осциляторного процесу у векторній ЕЕГ оператора ІМК, можливо отримати більш повну, надійну, точну інформацію не тільки про контрольні сигнали оператора ІМК, але й про закономірності тренування, зокрема, ефективність навчання оператора ІМК. Також серед переваг розробленої математичної моделі векторної ЕЕГ є існування відомих методів комп'ютерного моделювання вектора випадкових сигналів, що дозволяє проводити комп'ютерні моделювальні експерименти з метою тестування, оптимізації методів та апаратного і програмного забезпечення для обробки векторної ЕЕГ у системах нейроінтерфейсу.

2.3. Ритмоадаптивні методи опрацювання векторної ЕЕГ в системах ІМК

Фактична розробка математичної моделі векторної ЕЕГ у формі вектора циклічних ритмічно пов’язаних випадкових процесів із двома зонами на циклах дозволяє розробити необхідний математичний апарат для опрацювання векторної ЕЕГ. А саме, методи оцінювання характеристик векторної ЕЕГ полягають у застосуванні методів статистичного оцінювання ймовірнісних характеристик вектора циклічних ритмічно пов’язаних випадкових процесів, так і в застосуванні методів гармонійного аналізу отриманих статистичних оцінок.

Згідно з роботою [62], для застосування статистичних методів оцінювання ймовірнісних характеристик вектора циклічних ритмічно пов’язаних випадкових процесів необхідна попередня оцінка його функції ритму. Розглянемо найпростіший метод оцінювання - функцію ритму векторної ЕЕГ, а саме метод кусково-лінійної інтерполяції, який розроблено в роботах [28, 62].

Розглянемо найпростіший тип інтерполяції – кусково-лінійну інтерполяцію. У цьому випадку функція інтерполяції $\hat{T}(t,1)$ буде виглядати так:

$$\hat{T}(t,1) = \sum_{m\in Z}\sum_{k=1}^{2}\hat{T}_{mk}(t), t \in R, \quad (14)$$

де $\{\hat{T}_{mk}(t)\}$ - набір функцій, що дорівнює:

$$\hat{T}_{mk}(t) = \begin{cases} g_{mk} \cdot t + b_{mk}, t \in W_{mk}, \\ 0, t \notin W_{mk}, m \in Z, k = \overline{1,2}. \end{cases} \quad (15)$$

Область $W_{mk} = [t_{mk}, t_{m,k+1}]$ відповідає *k*-тій зоні у *m*-му циклі. Якщо *k=2*, то $t_{m,3} = t_{m+1,1}$. Звісно, на практиці m приймає свої значення зі скінченної підмножини цілих чисел.

Отже, при кусково-лінійній інтерполяції функції ритму необхідно знайти набори коефіцієнтів $\{g_{mk}, m \in Z, k = \overline{1,2}\}$ та $\{b_{mk}, m \in Z, k = \overline{1,2}\}$, які повністю визначатимуть функцію інтерполяції $\hat{T}(t, 1)$. Знайдемо вирази для обчислення необхідних коефіцієнтів функції інтерполяції. Для цього запишемо рівняння відрізка, що з'єднує точки з координатами $\left(t_{m,k}, T\left(t_{m,k}, 1\right)\right)$ та $\left(t_{m,k+1}, T\left(t_{m,k+1}, 1\right)\right)$, воно поєднує показники дискретної функції ритму $T\left(t_{m,k}, 1\right)$ у моменти $t_{m,k}$ та $t_{m,k+1}$:

$$\frac{\hat{T}_{mk}(t) - T\left(t_{m,k+1}, 1\right)}{T\left(t_{m,k+1}, 1\right) - T\left(t_{m,k}, 1\right)} = \frac{t - t_{m,k+1}}{t_{m,k+1} - t_{m,k}}, t \in W_{mk}, m \in Z, k = \overline{1,2}. \quad (16)$$

Тоді можна легко показати, що ці рівняння можна звести до наступних рівнянь:

$$\hat{T}_{mk}(t) = \frac{T(t_{m,k+1},1) - T(t_{m,k},1)}{t_{m,k+1} - t_{m,k}} \cdot t + T\left(t_{m,k+1}, 1\right) - \frac{T(t_{m,k+1},1) - T(t_{m,k},1)}{t_{m,k+1} - t_{m,k}} \cdot t_{m,k+1} \quad (17)$$
$$t \in W_{mk}, m \in Z, k = \overline{1,2},$$

або аналогічно:

$$\hat{T}_{mk}(t) = \frac{T\left(t_{m,k+1}, 1\right) - T\left(t_{m,k}, 1\right)}{t_{m,k+1} - t_{m,k}} \cdot \left(t - t_{m,k+1}\right) + T\left(t_{m,k+1}, 1\right), \quad (18)$$
$$t \in W_{mk}, m \in Z, k = \overline{1,2}$$

Отже, згідно з рівнянням (17), коефіцієнти $\{g_{mk}, m \in Z, k = \overline{1,2}\}$ та $\{b_{mk}, m \in Z, k = \overline{1,2}\}$ будуть визначені співвідношеннями:

$$g_{mk} = \frac{T\left(t_{m,k+1}, 1\right) - T\left(t_{m,k}, 1\right)}{t_{m,k+1} - t_{m,k}}, m \in Z, k = \overline{1,2} \quad (19)$$

$$b_{mk} = T\left(t_{m,k+1}, 1\right) - \frac{T\left(t_{m,k+1}, 1\right) - T\left(t_{m,k}, 1\right)}{t_{m,k+1} - t_{m,k}} \cdot t_{m,k+1}, m \in Z, k = \overline{1,2.} \quad (20)$$

Згідно з умовами, накладеними на функцію інтерполяції її похідна $\hat{T}(t, 1)$, яка існує у всіх точках множини $\boldsymbol{R}$, за винятком точок множини $D_z = \left\{t_{m,k}, m \in Z, k = \overline{1,2}\right\}$ повинна бути більшою за мінус один, і це можливо лише тоді, коли функції похідних $\left\{\hat{T}_{mk}(t)\right\}$ будуть більші за -1. Функції похідних $\left\{\hat{T}_{mk}(t)\right\}$ дорівнюють коефіцієнтам, $\{g_{mk}\}$, які обчислюються згідно з формулою (19) і завжди більші за -1, оскільки $T\left(t_{m,k+1}, 1\right) - T\left(t_{m,k}, 1\right) > t_{m,k} - t_{m,k+1}$, що випливає з умов функції ритму, яка, звісно, задовольняється дискретною функцією ритму, $T\left(t_{m,k}, 1\right)$ а саме:

$$g_{mk} = \frac{T\left(t_{m,k+1}, 1\right) - T\left(t_{m,k}, 1\right)}{t_{m,k+1} - t_{m,k}} > -1, m \in Z, k = \overline{1,2} \quad (21)$$

Запишемо формули для обчислення реалізацій відповідних статистичних оцінок ймовірнісних характеристик векторної ЕЕГ. У випадку її тривалої реалізації $\Theta_{N_\omega}(t) = \left\{\xi_{i_\omega}(t), i = \overline{1,N}, t \in W \subset R\right\}$, яка складається з M циклів. У цьому випадку $W = U_{m=1}^{M} W_{c_m}$.

Реалізація статистичної оцінки математичного сподівання $m_{\xi_i}(t)$ кожного компонента $\xi_i(\omega, t)$ вектора $\Theta_N(\omega, t)$:

$$\hat{m}_{\xi_i}(t) = \frac{1}{M} \sum_{n=0}^{M-1} \xi_{i_\omega}\left(t + T(t, n)\right), t \in W_{c_1} = [t_1, t_2), i = \overline{1,N}. \quad (22)$$

Реалізація статистичної оцінки дисперсії $d_{\xi_i}(t)$ кожного компонента $\xi_i(\omega, t)$ вектора $\Theta_N(\omega, t)$:

$$\hat{d}_{\xi_i}(t) = \frac{1}{M} \cdot \sum_{n=0}^{M-1} \left[\xi_{i_\omega}\left(t + T(t, n)\right) - \hat{m}_{\xi_i}\left(t + T(t, n)\right)\right]^2, \quad t \in W_{c_1} = [t_1, t_2), i = \overline{1,N}. \quad (23)$$

Реалізація статистичної оцінки функції початкового моменту ${m_{\xi_i}}^k(t)$ k-го порядку кожного компонента $\xi_i(\omega, t)$ вектора $\Theta_N(\omega, t)$:

$$\hat{m}_{\xi_i}{}^k(t) = \frac{1}{M} \cdot \sum_{n=0}^{M-1} \xi_{i\omega}^k\big(t + T(t,n)\big), t \in W_{c_1} = [t_1, t_2), i = \overline{1,N}. \quad (24)$$

Реалізація статистичної оцінки функції центрального моменту $d_{\xi_i}{}^k(t)$ k-го порядку кожного компонента $\xi_i(\omega,t)$ вектора $\Theta_N(\omega,t)$:

$$\hat{d}_{\xi_i}{}^k(t) = \frac{1}{M-1} \cdot \sum_{n=0}^{M-1} \big[\xi_{i_\omega}\big(t + T(t,n)\big) - \hat{m}_{\xi_i}\big(t + T(t,n)\big)\big]^k, \quad (25)$$
$$t \in W_{c_1} = [t_1, t_2), i = \overline{1,N}.$$

Реалізація статистичної оцінки змішаної функції початкового моменту $C_{p_{\xi_i}}{}^k(t_1, \dots, t_k)$ порядку $p = \sum_{i=1}^k r_i$ кожного компонента $\xi_i(\omega,t)$ вектора $\Theta_N(\omega,t)$:

$$\hat{C}_{p_{\xi_i}}{}^k(t_1, \dots, t_k) = \frac{1}{M - M_1 + 1} \cdot$$
$$\cdot \sum_{n=0}^{M-M_1} \big[\xi_{i_\omega}{}^{r_1}\big(t_1 + T(t_1,n)\big) - \xi_{i_\omega}{}^{r_k}\big(t_k + T(t_k,n)\big)\big], \quad (26)$$
$$t_1 \in W_{c_1}, t_2, \dots, t_k \in m = 1M_1W_{c_m}, i = \overline{1,N}.$$

Де $M_1 (M_1 \ll M)$ - кількість циклів, у яких аргументи набувають значення $t_2, \dots, t_k$.

Реалізація статистичної оцінки змішаної функції центрального моменту $R_{p_{\xi_i}}(t_1, \dots, t_k)$ порядку $p = \sum_{i=1}^k r_i$ кожного компонента $\xi_i(\omega,t)$ вектора $\Theta_N(\omega,t)$:

$$\hat{R}_{p_{\xi_i}}(t_1, \dots, t_k) = \frac{1}{M - M_1} \cdot$$
$$\cdot \sum_{n=0}^{M-M_1} \Big[\Big(\xi_{i_\omega}\big(t_1 + T(t_1,n)\big) - \hat{m}_{\xi_i}\big(t_1 + T(t_1,n)\big)\Big)^{r_1} \cdot \dots \quad (27)$$
$$\cdot \Big(\xi_{i_\omega}\big(t_k + T(t_k,n)\big) - \hat{m}_{\xi_i}\big(t_k + T(t_k,n)\big)\Big)^{r_k}\Big],$$
$$t_1 \in W_{c_1}, t_2, \dots, t_k \in m = 1M_1W_{c_m}, i = \overline{1,N}.$$

Наведені вище статистичні оцінки дають змогу оцінити ймовірнісні характеристики кожної окремої компоненти вектроного ЕЕГ сигналу. З метою

проведення сумісного статистичного опрацювання векторного ЕЕГ сигналу необхідно застосовувати наведені нижче статистичні оцінки.

Реалізація статистичної оцінки змішаної функції початкового моменту $C_{p_{\xi_{i_1},\dots,\xi_{i_k}}}(t_1,\dots,t_k)$ порядку $p=\sum_{j=1}^{k} r_j$ вектора $\Theta_N(\omega,t)$:

$$\hat{C}_{p_{\xi_{i_1},\dots,\xi_{i_k}}}(t_1,\dots,t_k)=\frac{1}{M-M_1+1}\cdot$$
$$\cdot\sum_{n=0}^{M-M_1}\left[\xi_{i_{1\omega}}^{R_1}\left(t_1+T(t_1,n)\right)\cdot\ldots\cdot\xi_{i_{k\omega}}^{R_k}\left(t_k+T(t_k,n)\right)\right], \tag{28}$$
$$t_1\in W_{c_1},t_2,\dots,t_k\in U_{m=1}^{M_1}W_{c_m},i_i,\dots,i_k=\overline{1,N}.$$

Реалізація статистичної оцінки змішаної функції центрального моменту $R_{p_{\xi_{i_1},\dots,\xi_{i_k}}}(t_1,\dots,t_k)$ порядку $p=\sum_{j=1}^{k} r_j$ вектора $\Theta_N(\omega,t)$:

$$\hat{R}_{p_{\xi_{i_1},\dots,\xi_{i_k}}}(t_1,\dots,t_k)=\frac{1}{M-M_1}\cdot$$
$$\cdot\sum_{n=0}^{M-M_1}\left(\xi_{i_1\omega}\left(t_1+T(t_1,n)\right)-\hat{m}_{\xi_{i_1}}\left(t_1+T(t_1,n)\right)\right)^{r_1}\cdot\ldots \tag{29}$$
$$\cdot\left(\xi_{i_k\omega}\left(t_k+T(t_k,n)\right)-\hat{m}_{\xi_{i_k}}\left(t_k+T(t_k,n)\right)\right)^{r_k},$$
$$t_1\in W_{c_1},t_2,\dots,t_k\in U_{m=1}^{M_1}W_{c_m},i_i,\dots,i_k=\overline{1,N}.$$

Наведені вище методи статистичного оцінювання ймовірнісних характеристик векторного ЕЕГ сигналу дають змогу проводити його статистичний аналіз у рамках початкових, центральних та змішаних моментних функцій.

2.4 Висновки до другого розділу

У другому розділі розроблено нову математичну модель векторної ЕЕГ у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів, що є математичною основою для обґрунтування методів аналізу ЕЕГ сигналів у нейроінтерфейсних системах. Модель ефективно враховує стохастичність,

циклічність, мінливість та спільність ритму ЕЕГ сигналів, що дає змогу досліджувати статистичні характеристики ментальних керуючих впливів операторів нейроінтерфейсних систем.

Обґрунтовано методи статистичного оцінювання ймовірнісних характеристик векторної ЕЕГ на базі розробленої математичної моделі. Зокрема, обґрунтовано методи статистичного оцінювання сумісних моментних функцій векторної ЕЕГ. Отримані результати розширюють можливості підвищення точності аналізу та класифікації нейроінтерфейсних ЕЕГ сигналів, та є основою для підвищення інформативності неінвазивних нейроінтерфейсних систем до ментальних керуючих впливів їх операторів.

РОЗДІЛ 3
МЕТОДИ ТА ПРОГРАМНО-АПАРАТНІ ЗАСОБИ ОПРАЦЮВАННЯ ЕЕГ СИГНАЛІВ У НЕЙРОІНТЕРФЕЙСНИХ СИСТЕМАХ

Третій розділ дисертації присвячено розробці, обґрунтуванню та експериментальній верифікації методів та програмно-апаратних засобів прототипу неінвазивної нейроінтерфейсної системи, яка втілює розроблені у другому розділі математичну модель та методи статистичного опрацювання векторного ЕЕГ сигналу. Розглянуто основні етапи опрацювання ЕЕГ сигналів в нейроінтерфейсних системах, від реєстрації за допомогою платформи OpenBCI до класифікації та використання даних в реальному часі. Експериментально обґрунтовано інформативні та чутливі до ментального впливу оператора ІМК ймовірнісні характеристики та вектор ознак в системах ІМК. Розроблено програмне забезпечення прототипу неівазивної нейроінтерфейсної системи на базі Python з інтеграцією таких бібліотек як Pandas, Numpy, Scipy, та інших, що уможливило ефективне виявлення та аналіз ментальних керуючих впливів.

Основні результати розділу опубліковано в працях [1, 9, 115, 116].

3.1. Основні етапи опрацювання ЕЕГ сигналів у нейроінтерфейсній системі

З метою експериментальної верифікації розробленої математичної моделі та методів статистичного оцінювання ймовірнісних характеристик векторного ЕЕГ сигналу необхідно було розробити методи та програмно-апаратні засоби опрацювання ЕЕГ сигналів. Тому було розроблено прототип програмно-апаратного неінвазивного нейроінтерфейсу та проведено низку вимірювальних та обчислювальних експериментів для різних операторів ІМК.

Процедуру опрацювання ЕЕГ сигналів у нейроінтерфейсній системі можна розділити на п’ять етапів (Рисунок 3.1): реєстрація ЕЕГ сигналів, їх попереднє опрацювання та оцінювання ймовірнісних характеристик, класифікація (детектування, розпізнавання) сигналів та комп’ютерна взаємодія [1, 2, 9].

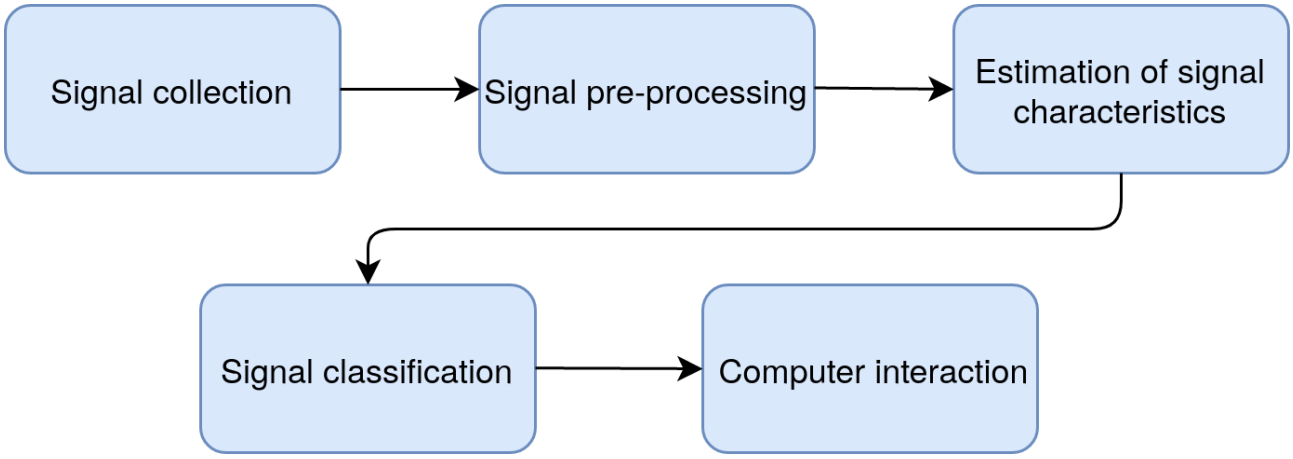


Рисунок. 3.1. Основні етапи опрацювання ЕЕГ сигналів в нейроінтерфейній системі

У наступних підрозділах коротко розглянемо методи та програмно-апаратні засоби для реалізації усіх цих етапів опрацювання ЕЕГ сигналів.

3.2. Платформа OpenBCI як основа апаратного забезпечення неінвазивного нейроінтерфейсу

Для реєстрації сигналів ЕЕГ використовувалася платформа OpenBCI (Open Source Brain-Computer Interface). Платформа OpenBCI представляє собою відкрите та доступне рішення для реєстрації, візуалізації та аналізу біоелектричних сигналів мозку (ЕЕГ), а також інших видів біометричних даних. Вона забезпечує розробників та користувачів інструментами для інноваційних досліджень у галузі нейронаук і біомедичної інженерії. Особливістю OpenBCI є її відкритість та модульність, що дозволяє користувачам адаптувати систему під конкретні дослідницькі потреби. OpenBCI вирізняється своєю модульною структурою, яка дозволяє додавати або змінювати компоненти системи залежно від специфіки дослідження. Це може включати зміну типу електродів, розширення кількості каналів або інтеграцію з іншими біометричними сенсорами. Така гнучкість робить платформу ідеальною для широкого спектру досліджень в області нейронаук та розробки медичних пристроїв. OpenBCI

підтримує активну та відкриту спільноту користувачів і розробників, яка сприяє обміну знаннями, досвідом та ідеями.

Наявність великої кількості відкритої документації, навчальних матеріалів, та форумів для обговорення робить платформу доступною для людей з різним рівнем знань та досвіду. Завдяки цьому, у домашніх умовах було самостійно виготовлено апаратну платформу для досліджень. Цей процес не тільки демонструє доступність та відкритість технології, але й відкриває нові можливості для індивідуальних та незалежних наукових проектів. Самостійне складання платформи дало змогу детально вивчити її принципи роботи та адаптувати систему під специфічні дослідницькі потреби, надаючи унікальну можливість для інновацій та експериментів у сфері ІМК.

Платформа використовує 8-канальний, 24-бітний, низькошумовий аналого-цифровий перетворювач (ADC) ADS1299ADC для запису сигналів ЕЕГ (Рисунок 3.2). Частота дискретизації у 250 Гц для кожного каналу є достатньою для захоплення широкого спектра нейронних сигналів. Для фіксації електродів (Cz, C3, C4) використовувався гарнітура Ultracortex Mark IV (Рисунок 3.3). Електроди, закріплені в гарнітурі, є сухого типу з покриттям Ag-AgCl. Кожен електрод має 12 тупих зубців довжиною 5 мм, що забезпечує комфорт та хороший контакт із поверхнею шкіри голови оператора ІМК.

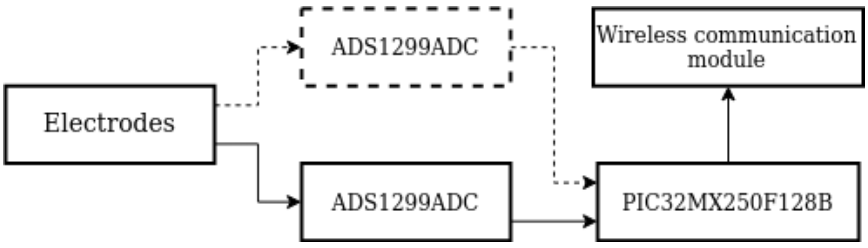


Рисунок. 3.2. Структурна схема платформи OpenBCI

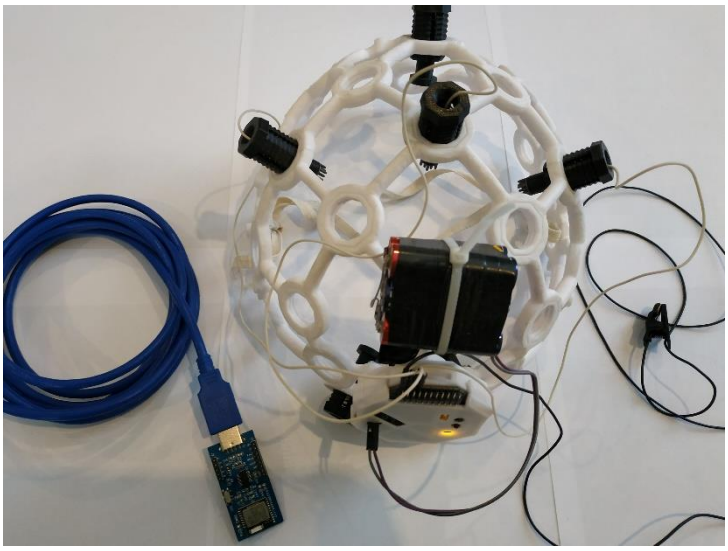


Рисунок. 3.3. Платформа OpenBCI, яку було виготовлено власноруч

3.3. Програмне забезпечення неівазивної нейроінтерфейсної системи

У відкритому доступі існує широка множина наборів інструментів та програмного забезпечення для відображення та опрацювання сигналів ЕЕГ, включаючи EEGLAB [117], CARTOOL [118], FieldTrip [119], Brainstorm [120], утиліта візуалізації GraphVar [121], EEGNET [122] тощо. Однак всі ці програмні системи не мають змогти опрацювувати векторний ЕЕГ сигнал на основі розробленої у дисертації нової математичної моделі у вигляді вектора циклічних ритмічно повязаних випадкових процесів. Такий стан справ вимагав розробки нового програмного забезпечення найроінтерфейсної системи та адаптації існуючих програмних систем.

Для управління процесом реєстрації сигналу ЕЕГ використовується утиліта OpenBCI GUI (Рисунок 3.3). Результати вимірювань записуються на карту microSD. Для цифрового опрацювання ЕЕГ сигналів було використано власні скрипти, написаних мовою Python.

Python є потужною та гнучкою мовою програмування, яка широко використовується в наукових дослідженнях, зокрема у сфері аналізу даних та машинного навчання. Його популярність серед дослідників зумовлена простотою синтаксису, великою кількістю бібліотек та високою продуктивністю. В контексті

опрацювання сигналів ЕЕГ та інших біомедичних даних Python і його бібліотеки пропонують потужні інструменти для аналізу, візуалізації та інтерпретації даних.

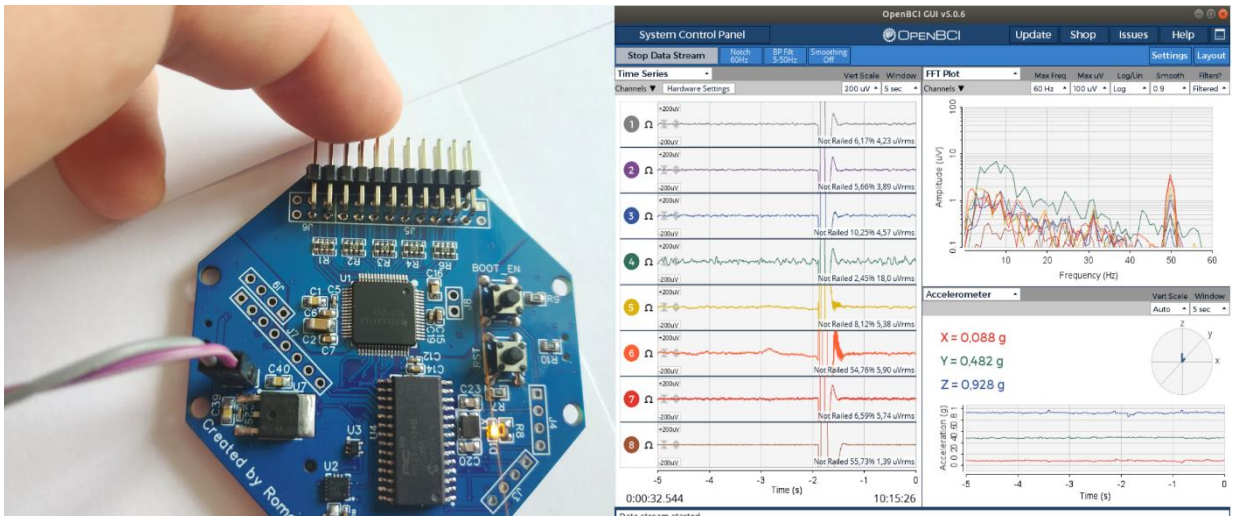


Рисунок. 3.4. Утиліта OpenBCI GUI

Розробка прототипу програмного забезпечення проводилася на базі операційної системи Linux, яка забезпечувала надійну та універсальну платформу, придатну для обчислень високого рівня та аналізу даних. Програмне забезпечення було створено з використанням Python 3.6.9. У процесі розробки були використані кілька спеціалізованих бібліотек, кожна з яких зробила значний внесок у функціональність програмного забезпечення:

— Pandas: Ця бібліотека була використана для перетворень даних та їх аналізу. Вона надає структури даних та операції для маніпулювання числовими таблицями та часовими рядами, роблячи її незамінною для опрацювання набору даних ЕЕГ.

— Numpy: є фундаментальною бібліотекою для наукових обчислень у Python. Вона надає підтримку для великих, багатовимірних масивів і матриць, разом із великою колекцією математичних функцій для операцій з цими структурами даних. У сфері ЕЕГ, numpy використовується для ефективного зберігання та маніпулювання даними, обчислення статистичних показників, фільтрації сигналів та інших основscipy

— Scipy: розширює функціонал numpy, додаючи корисні наукові та інженерні інструменти для Python. Вона охоплює різні області, такі як оптимізація, інтеграція, інтерполяція, власні вектори, статистика, спеціальні функції та інше. Для аналізу ЕЕГ scipy використовується для фільтрації сигналів, спектрального аналізу, та інших обчислень, що вимагаються при опрацюванні та аналізі біомедичних сигналів.

— Sklearn (Scikit-learn): є однією з провідних бібліотек Python для машинного навчання. Вона містить широкий спектр алгоритмів для класифікації, регресії, кластеризації, а також засоби для вибору моделі, обробки даних та оцінки їх якості. В контексті аналізу ЕЕГ, Scikit-learn може бути використана для розпізнавання патернів у даних, класифікації мозкових станів або прогнозування певних когнітивних або фізіологічних подій.

— Matplotlib є основною бібліотекою для візуалізації даних у Python. Вона дозволяє створювати якісні графіки, гістограми, спектральні діаграми, тощо, з великою гнучкістю та контролем над виглядом. У дослідженнях ЕЕГ matplotlib використовується для візуалізації сигналів, порівняння спектральних характеристик, представлення результатів класифікації, що робить її незамінною для аналізу та презентації результатів.

Використання цих бібліотек у комбінації із Python надає потужний інструментарій для обробки та аналізу даних ЕЕГ. Спільно ці інструменти дозволяють виконувати комплексні аналізи, від простих статистичних обчислень до складних моделей машинного навчання.

Для забезпечення ізольованого та контрольованого середовища розробки та запуску програмного забезпечення, було створено спеціалізоване середовище Python, відоме як Python віртуальне середовище (Python virtual environment). Віртуальне середовище є ізольованим технічним простором, який дозволяє встановлювати бібліотеки та пакети, необхідні для конкретного проекту, без впливу на інші проекти або основне системне середовище. Це ключовий компонент управління залежностями та версіями в проектах на Python, оскільки

це дозволяє створювати та використовувати точні конфігурації середовищ для кожного проекту.

Створення віртуального середовища було виконано за допомогою інструменту venv, який є стандартною частиною Python з версії 3.3. Після активації віртуального середовища, всі пакети та бібліотеки встановлюються та використовуються ізольовано від основної системи. Це значно спрощує управління проектом, оскільки забезпечує, що всі залежності точно відповідають потребам проекту, а також уникнення конфліктів між різними версіями бібліотек, які можуть бути використані в різних проектах.

Використання віртуального середовища також полегшує процес розгортання програмного забезпечення, оскільки дозволяє легко відтворити середовище розробки в продуктивному або іншому тестовому середовищі. Це досягається шляхом створення файлу requirements.txt, який містить список усіх залежностей проекту. Потім цей файл може бути використаний для встановлення точних версій усіх необхідних пакетів у новому віртуальному середовищі, забезпечуючи консистентність між розробкою та розгортанням.

3.4. Експериментальне дослідження ментальних керуючих впливів операторів ІМК

Суть експерименту полягала в безперервній реєстрації векторної ЕЕГ оператора ІМК, який мав візуалізувати процес згинання та розгинання пальців долоні правої руки за голосовими командами керівника експерименту. Всього було проведено серію з 11 експериментів (для 11 різних операторів ІМК), кожен з яких містив 50 циклів впливу оператора ІМК.

Оператори ІМК були розділені на дві групи: навчені та ненавчені (Група 1 та Група 2, відповідно). Навчені оператори мали попередній досвід роботи з ІМК, відзначалися високою здатністю до концентрації на задачі та ефективно використовували техніки візуалізації. Вони демонстрували здатність швидко адаптуватися до вимог експерименту та ефективно управляти сигналами ЕЕГ для досягнення заданих цілей. Натомість, ненавчені оператори не мали попереднього

досвіду роботи з ІМК та відзначалися нижчим рівнем концентрації та візуалізації, що впливало на їхню здатність до ефективного виконання експериментальних завдань. Це розділення дозволило детально оцінити вплив досвіду та тренування на ефективність використання нейроінтерфейсів та визначити ключові фактори, які сприяють успішній взаємодії людини з ІМК.

3.5. Попереднє опрацювання ЕЕГ сигналів

При записі сигналу ЕЕГ існує кілька технічних проблем, пов'язаних, в основному, з малою амплітудою сигналу. При проходженні сигналу ЕЕГ через тверду оболонку мозку, цереброспінальну рідину та череп до скальпу, його повна амплітуда становить всього близько 1 – 100 мікровольт, а частотний діапазон 0.5 – 100 Гц. Крім того, на запис також впливають матеріал електроду і стиснення контактів. Під час запису сигнал ЕЕГ може знаходитись під впливом зовнішніх артефактів, які за своєю природою бувають фізіологічного та технічного походження. До фізіологічних артефактів відносяться накладення кардіограми, рух очей, скорочення м'язів, рухи голови, тощо. До технічних артефактів відносяться мережне наведення частотою 50 Гц (60 Гц), що виникає внаслідок наявності електромагнітних полів, які генеруються електричною мережею в приміщенні, а також артефакти, пов'язані із ненадійним закріпленням електродів. Всі технічні артефакти зазвичай легко усунути.

Для того щоб мінімізувати кількість артефактів запису ЕЕГ необхідно, щоб суб'єкт дослідження під час експерименту перебував у розслабленому положенні, сидячи в спеціалізованому зручному кріслі. Повинна бути мінімізована кількість зовнішніх світлових і звукових подразників. Дуже важливим фактором є правильне накладання електродів і дотримання невеликого опору "електрод-шкіра" (не більше 5 кОм).

Згідно з схемою (див. рисунок 3.1), перед застосуванням методів оцінювання характеристик векторної ЕЕГ та методів виявлення (класифікації) контролюючих ментальних впливів, необхідно провести попереднє опрацювання досліджуваних сигналів ЕЕГ.

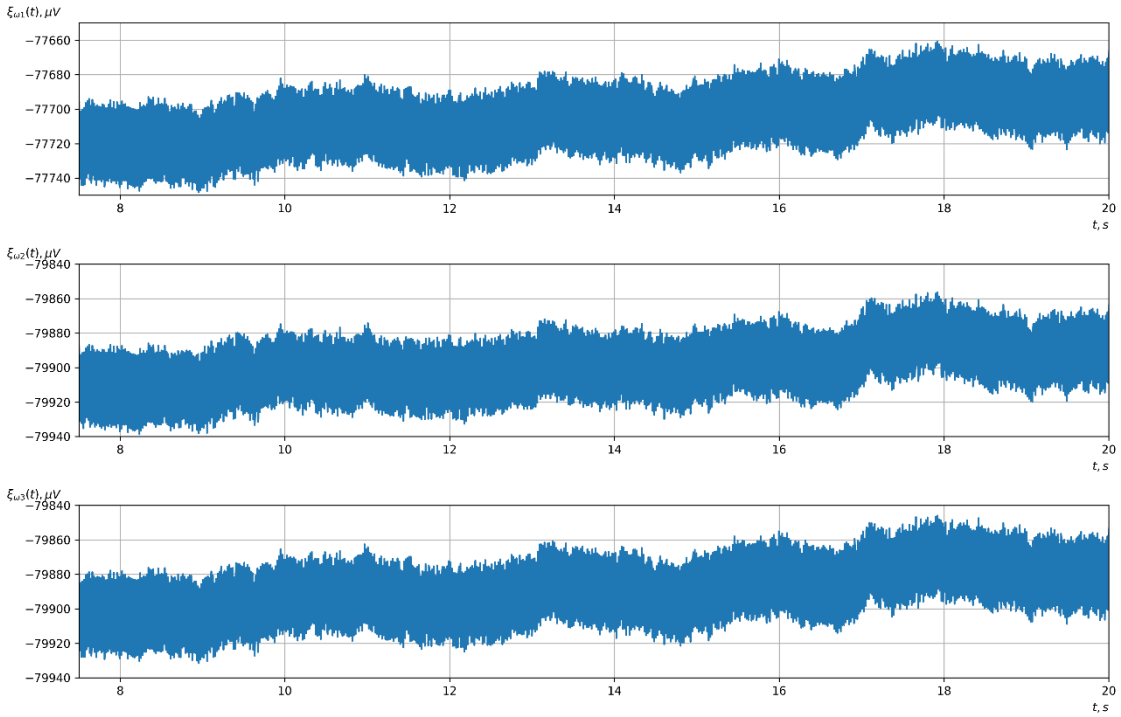


Рисунок. 3.5. Сигнали ЕЕГ після аналого-цифрового перетворювача, записані платформою OpenBCI

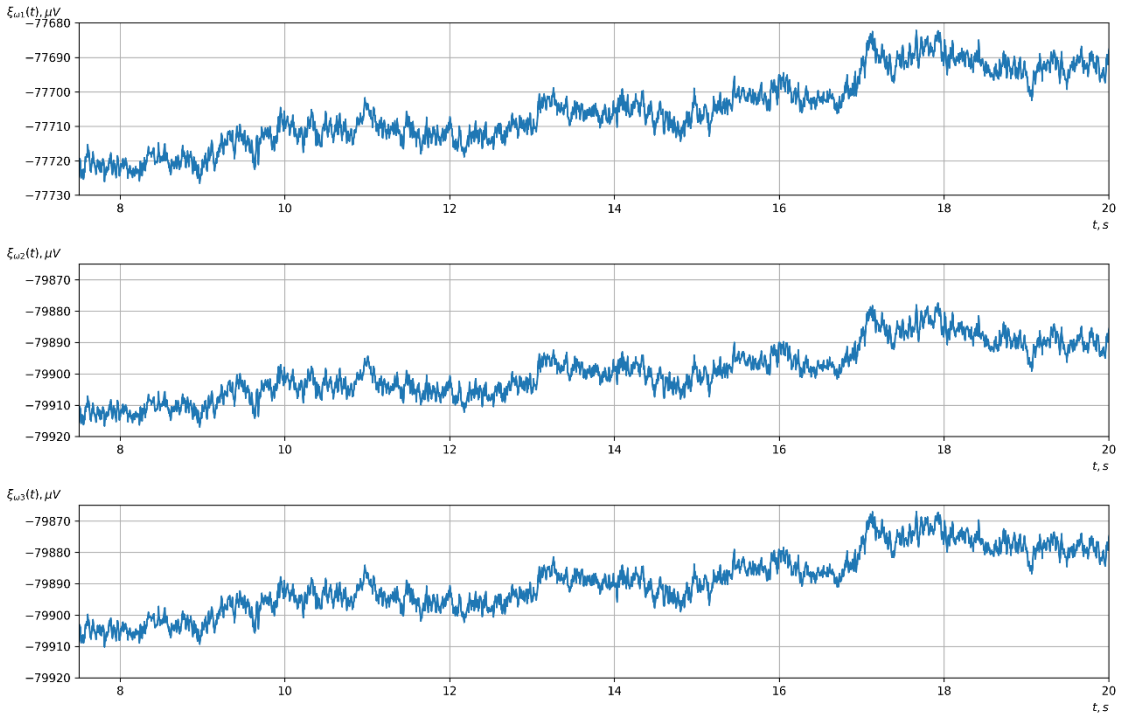


Рисунок. 3.6. Сигнали ЕЕГ після першого етапу фільтрації режекторним фільтром 50 Гц

Суть такого опрацювання полягає в використанні фільтрів Баттерворта. Для першого етапу використовується режекторним фільтр 3-го порядку. Його завданням є фільтрація шумів електромережі з частотою 50 Гц (60 Гц). Сигнали ЕЕГ до та після першого етапу фільтрації показані на рисунках 3.5 та 3.6.

Для другого етапу використовується смуговий фільтр 5-го порядку. У цьому експерименті смуга пропускання фільтра становить 1-17 Гц. Це дозволяє видалити всі низькочастотні та високочастотні шуми. Відфільтровані сигнали, які готові до наступних етапів опрацювання, показані на рисунку 3.7.

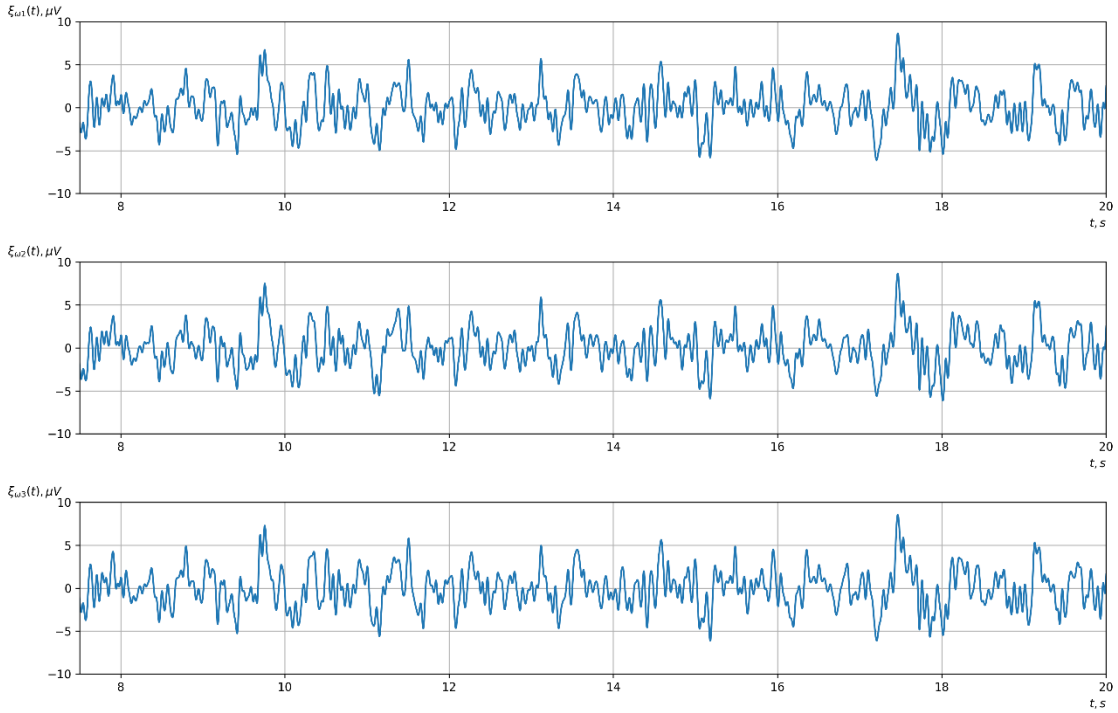


Рисунок. 3.7. Сигнали після другого етапу фільтрації смуговим фільтром

3.6. Оцінювання характеристик сигналів

На основі вищеописаних методів статистичного оцінювання ймовірнісних характеристик вектора циклічних ритмічно пов'язаних випадкових процесів (див. формули 22-25) було проведено статистичне оцінювання математичного сподівання, дисперсії, центральних і початкових моментних функцій вищих порядків ЕЕГ сигналів. Для всіх отриманих статистичних оцінок було проведено перетворення Фур'є.

На рисунку 3.8 показано графік оцінюваної функції ритму векторної ЕЕГ, який було отримано на основі методу кусково-лінійної інтерполяції, що описано у другому розділі дисертації. Аналогічно було проведено оцінювання функції ритму, та інших початкових і центральних моментних функцій для всіх одинадцяти операторів.

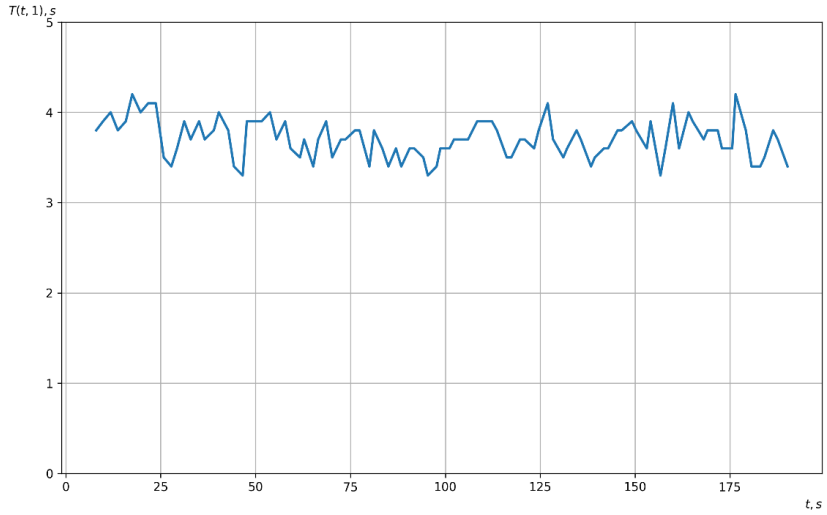
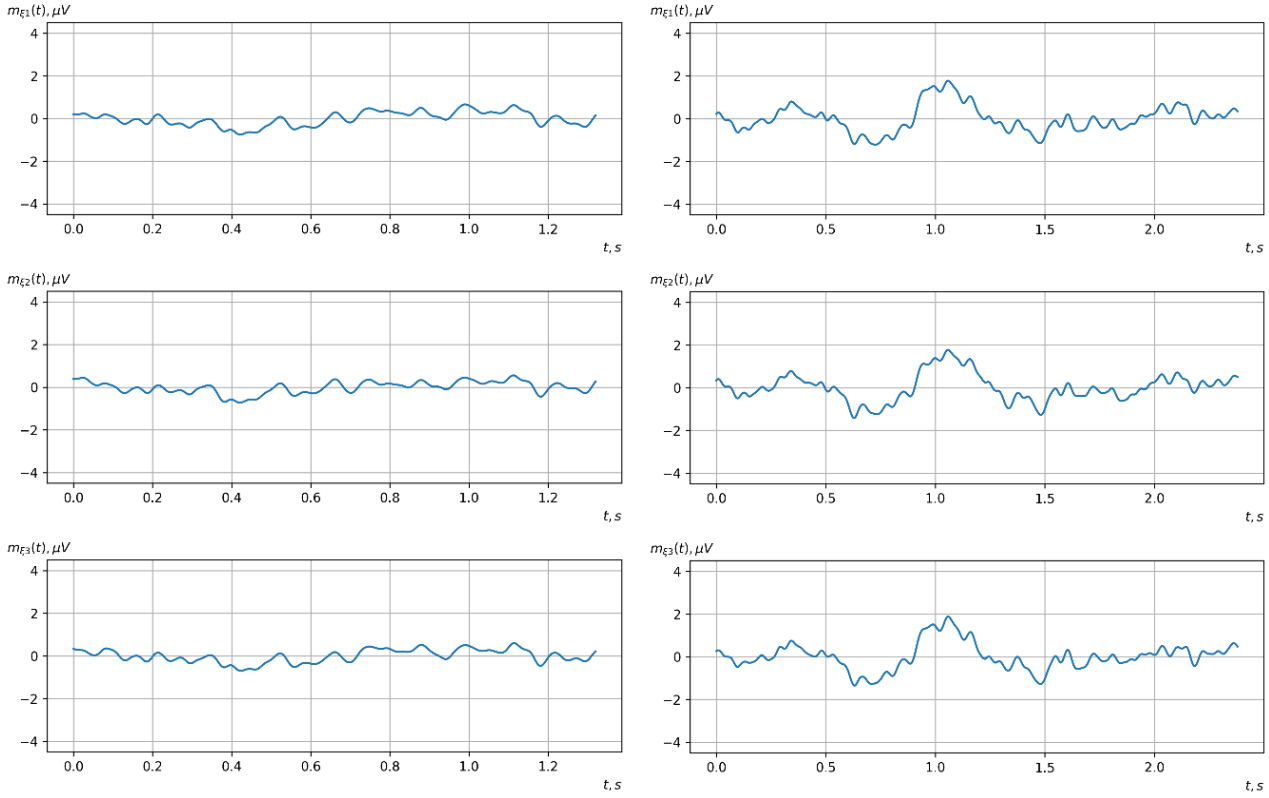


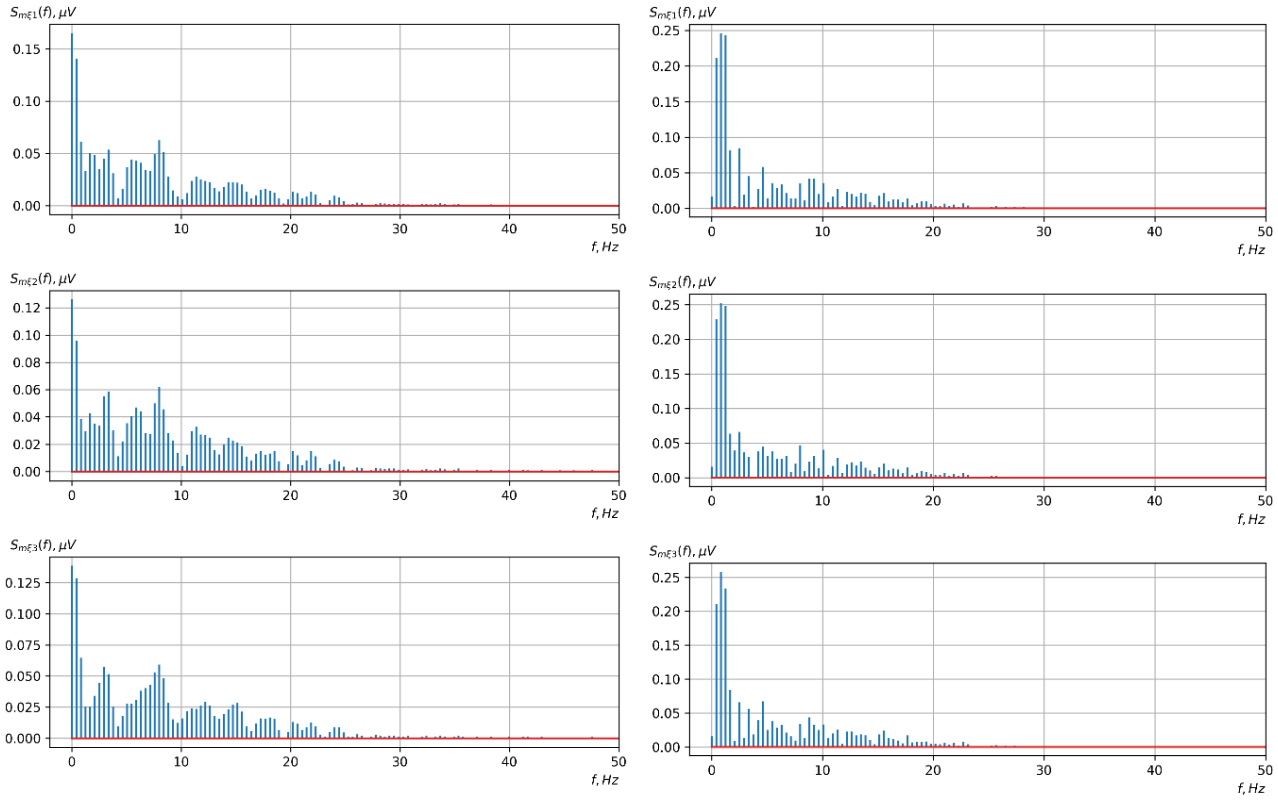
Рисунок. 3.8. Графік оцінки функції ритму векторної ЕЕГ для одинадцятого оператора

На рисунках 3.9 – 3.20 показані графіки реалізацій статистичних оцінок ймовірнісних характеристик компоненти векторної ЕЕГ для зони пасивності та зони активності оператора ІМК, які отримані за допомогою методів статистичного оцінювання ймовірнісних характеристик вектора циклічних ритмічно пов'язаних випадкових процесів.



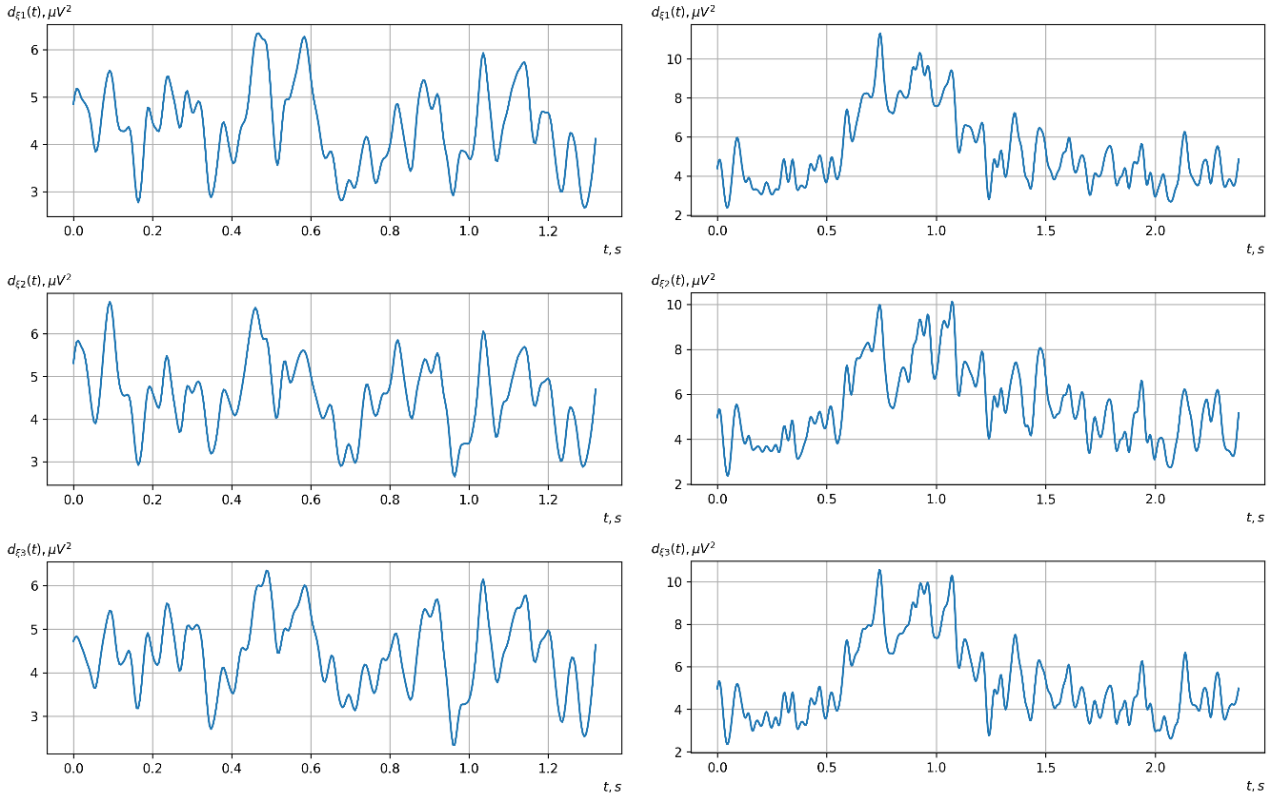
а) б)

Рисунок. 3.9. Графіки реалізацій статистичних оцінок математичних сподівання компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



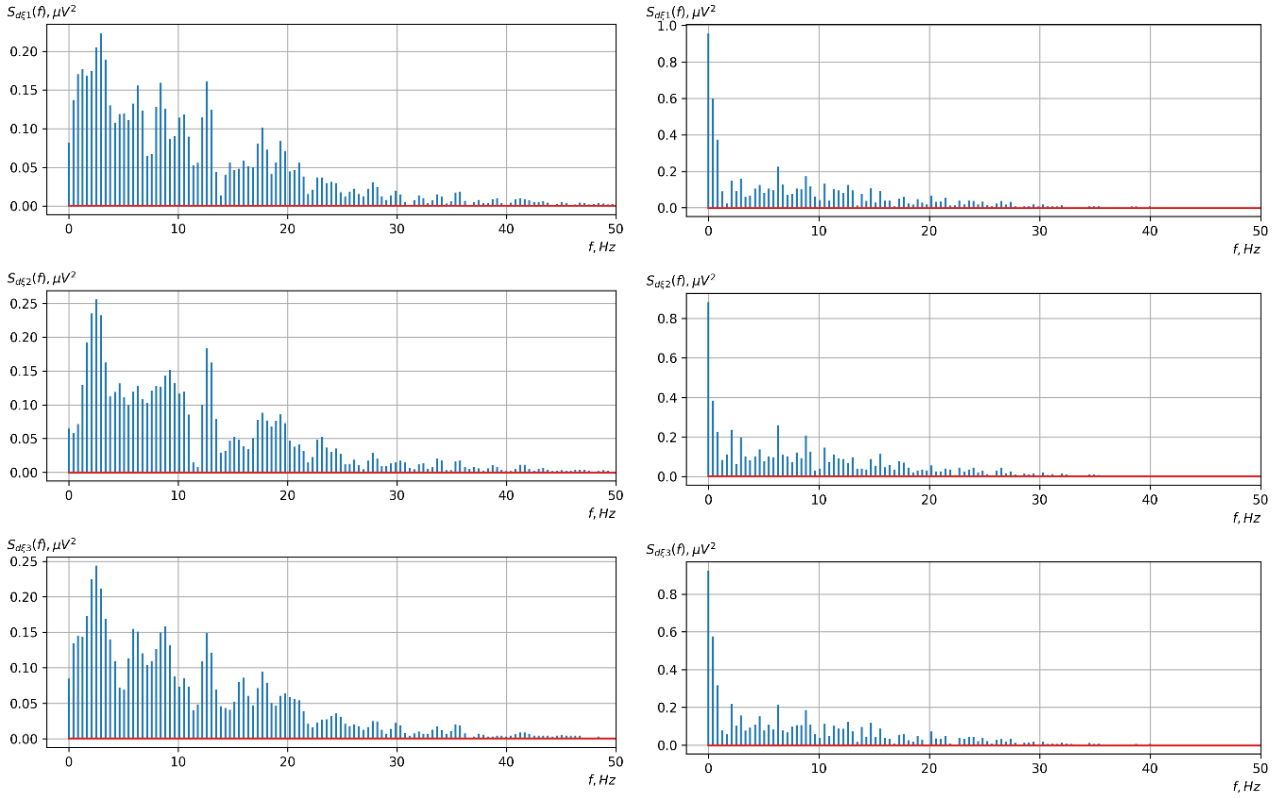
а) б)

Рисунок. 3.10. Графіки перетворень Фур'є реалізацій статистичних оцінок математичних сподівань компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



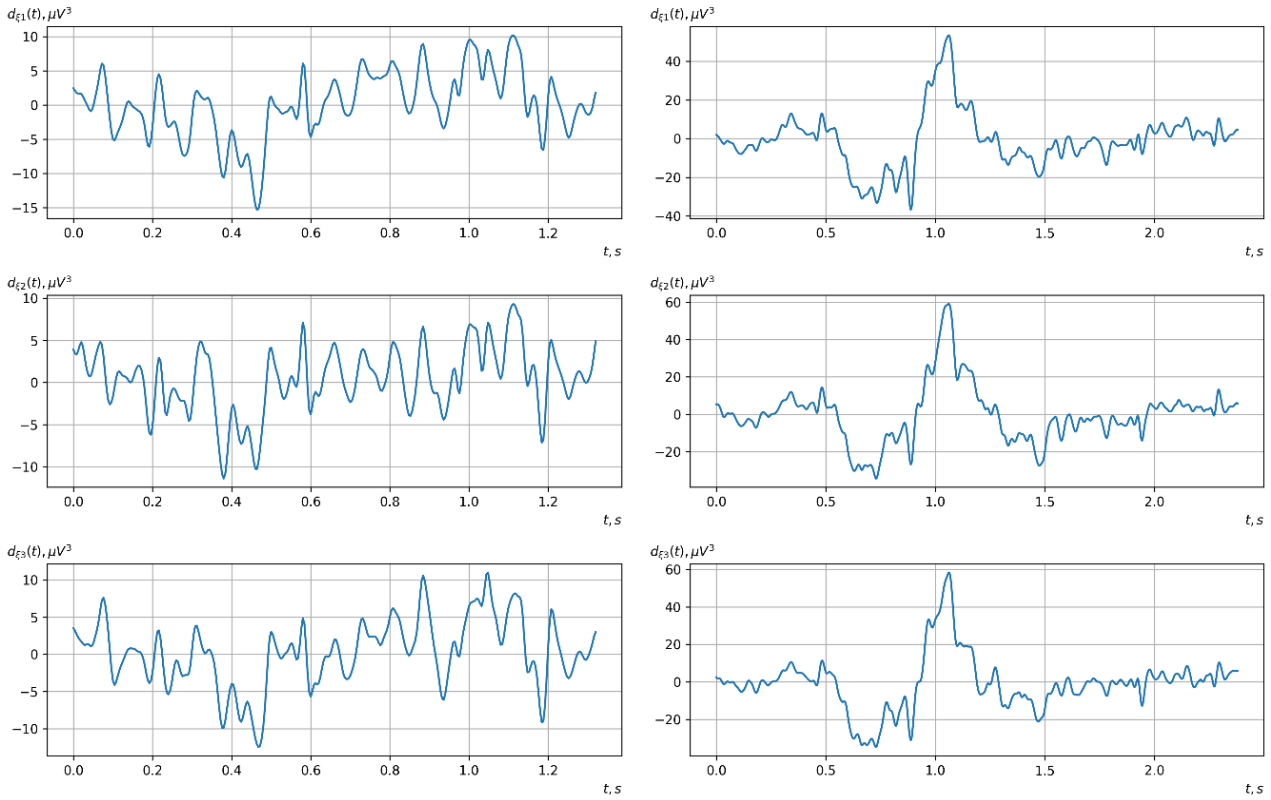
а) б)

Рисунок. 3.11. Графіки реалізацій статистичних оцінок початкових моментних функцій другого порядку компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



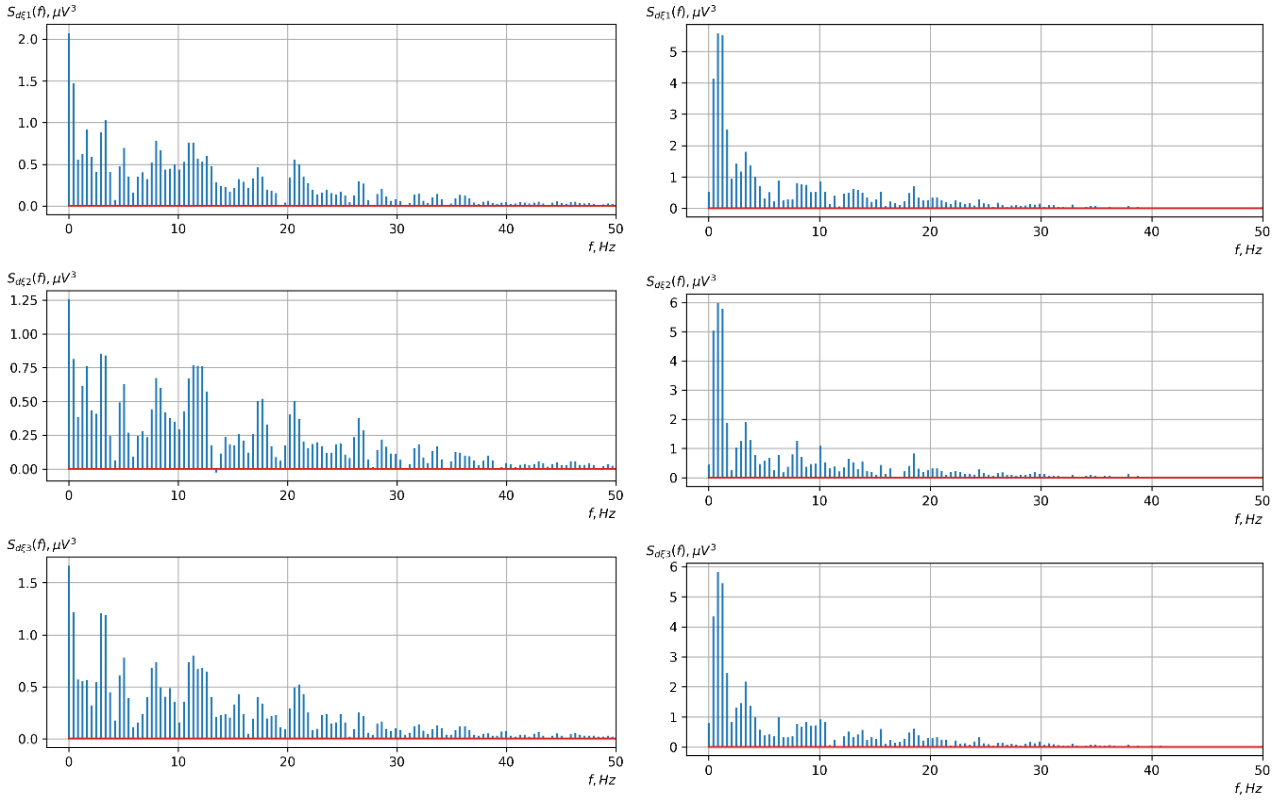
а) б)

Рисунок. 3.12. Графіки перетворень Фур'є реалізацій статистичних оцінок початкових моментних функцій другого порядку компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



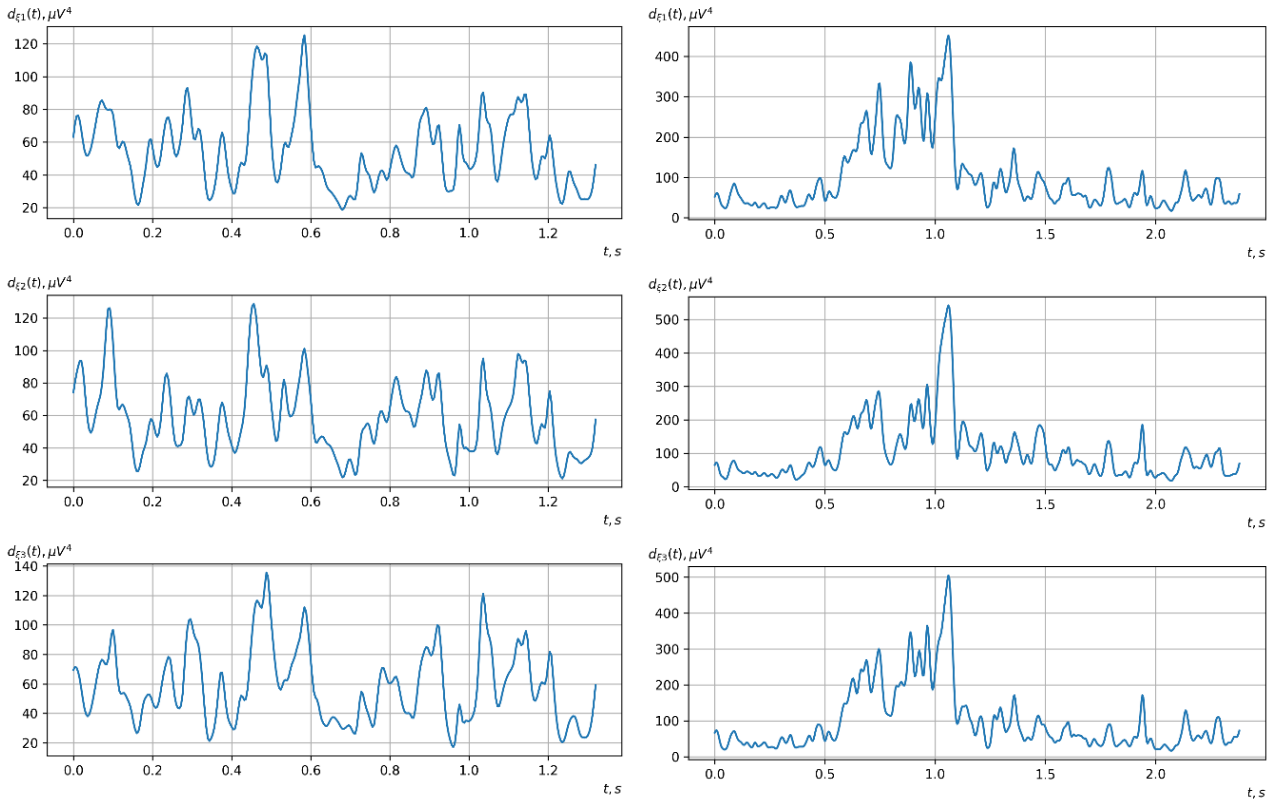
а) б)

Рисунок. 3.13. Графіки реалізацій статистичних оцінок початкових моментних функцій третього порядку компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



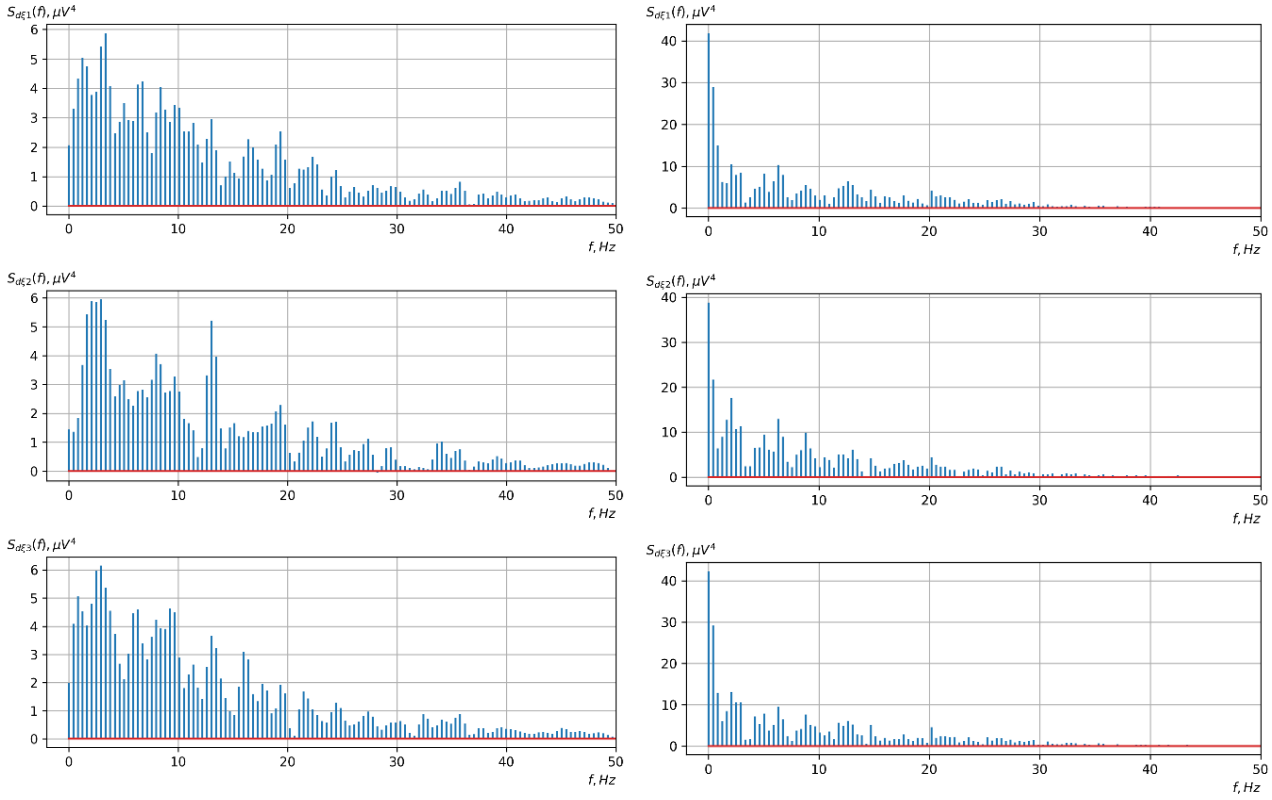
а) б)

Рисунок. 3.14. Графіки перетворень Фур'є реалізацій статистичних оцінок початкових моментних функцій третього порядку компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



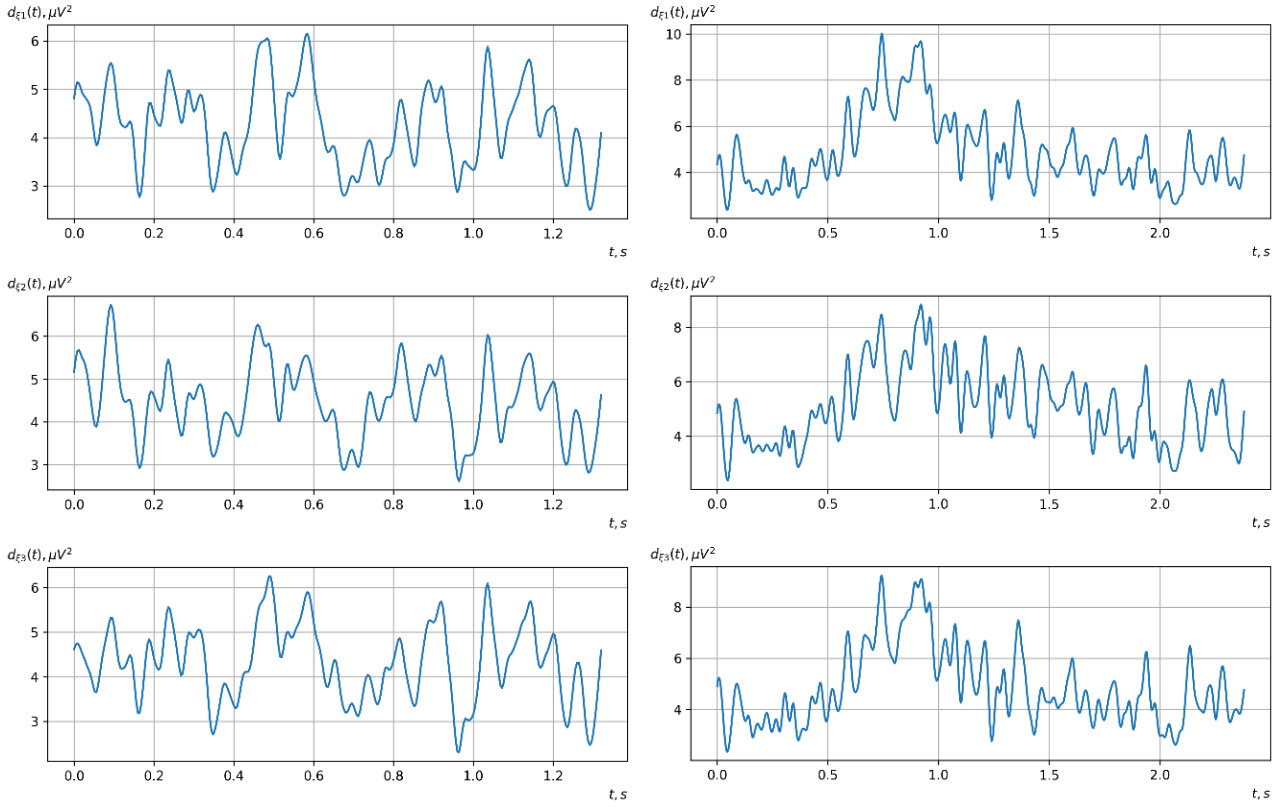
а) б)

Рисунок. 3.15. Графіки реалізацій статистичних оцінок початкових моментних функцій четвертого порядку компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



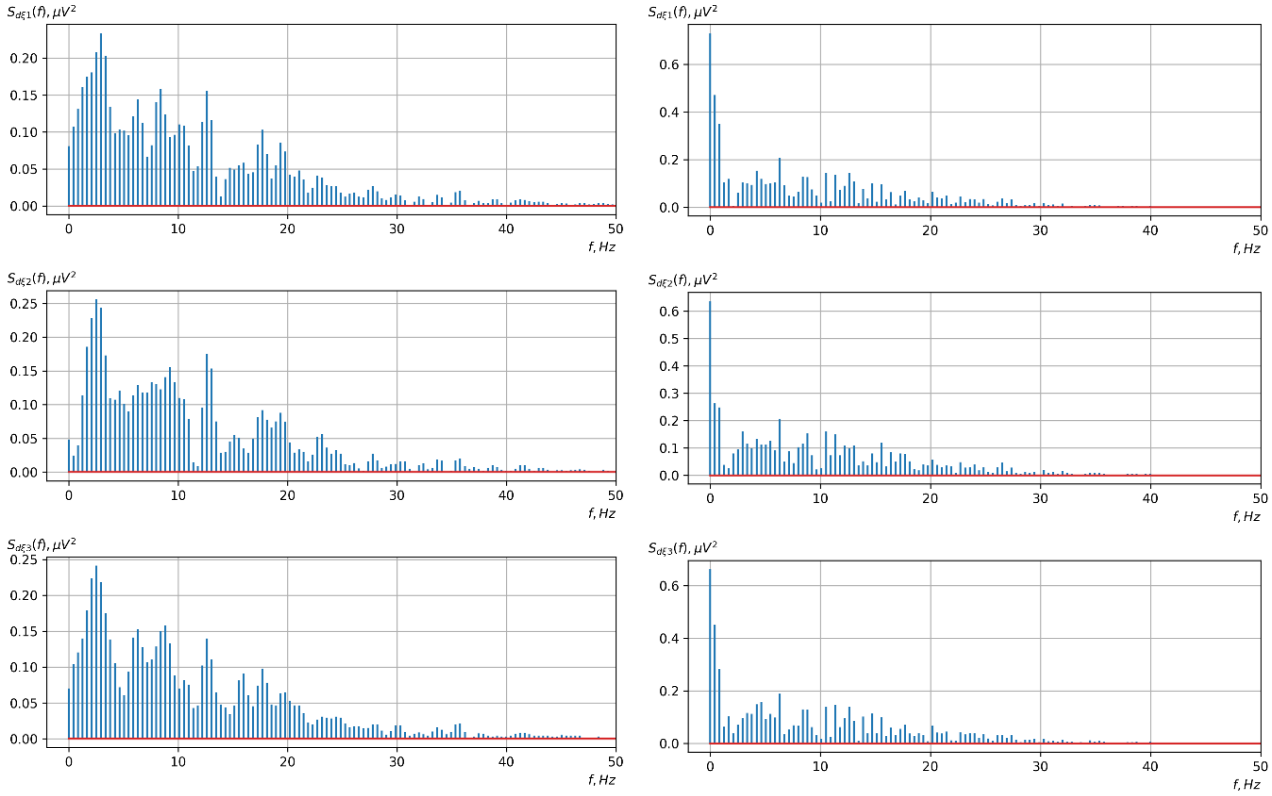
а) б)

Рисунок. 3.16. Графіки перетворень Фур'є реалізацій статистичних оцінок початкових моментних функцій четвертого порядку компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



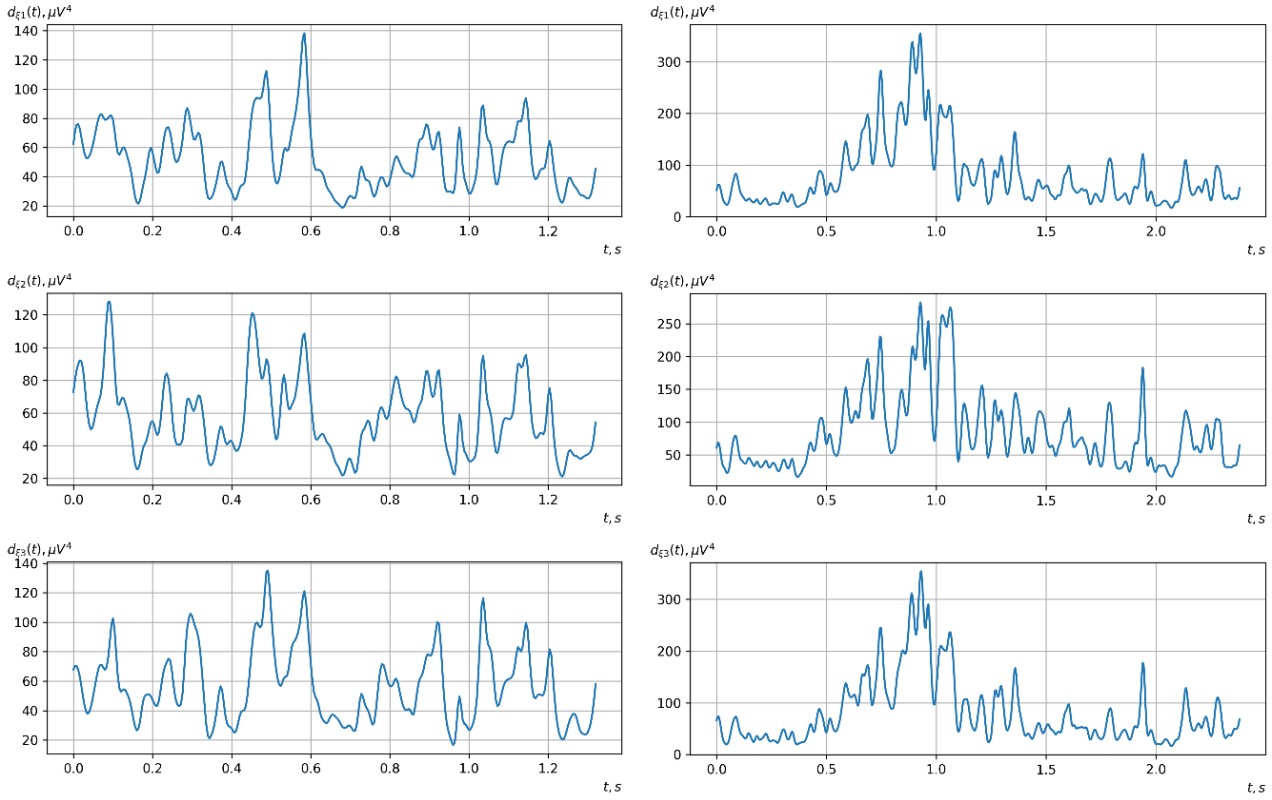
а) б)

Рисунок. 3.17. Графіки реалізацій статистичних оцінок дисперсій компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



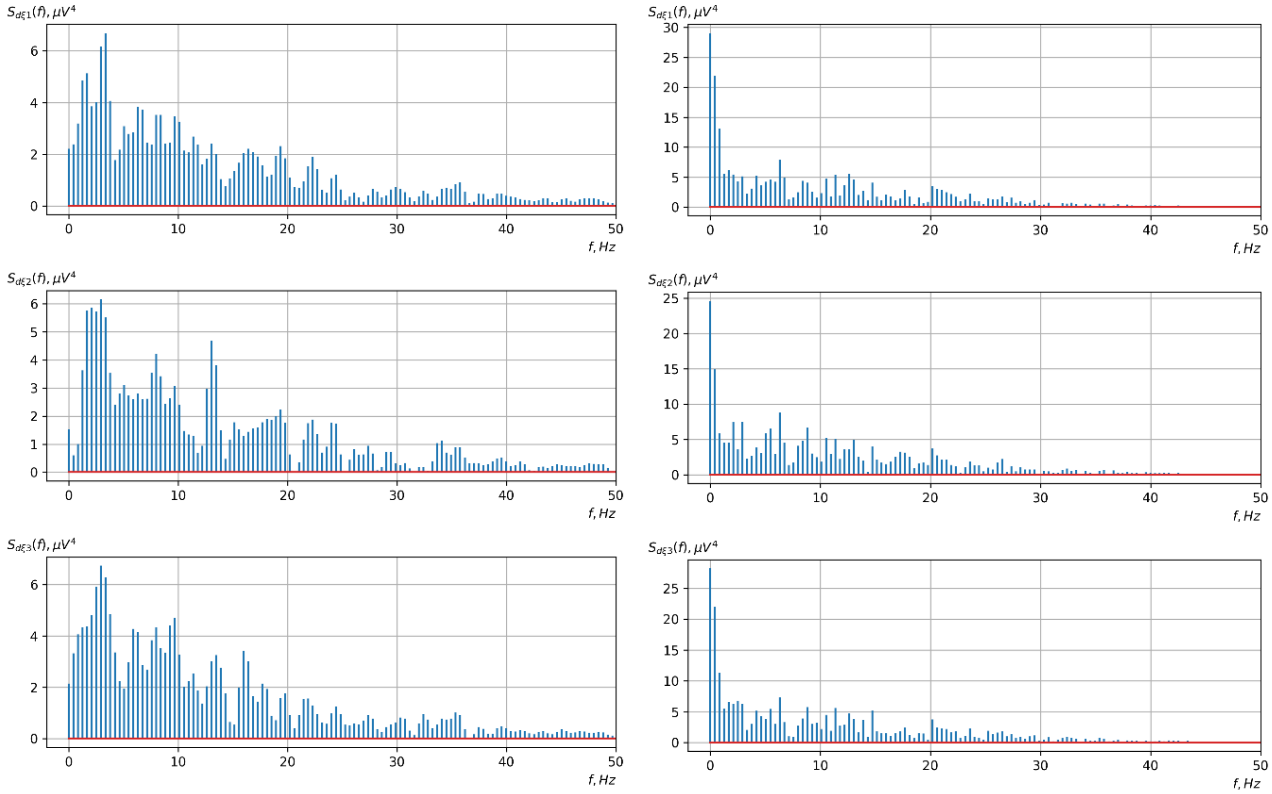
а) б)

Рисунок. 3.18. Графіки перетворень Фур'є реалізацій статистичних оцінок дисперсій компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



а) б)

Рисунок. 3.19. Графіки реалізацій статистичних оцінок центральних моментних функцій четвертого порядку компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК



а) б)

Рисунок. 3.20. Графіки перетворень Фур'є реалізацій статистичних оцінок центральних моментних функцій четвертого порядку компонент векторної ЕЕГ: а) зона пасивності (відсутність дії) оператора ІМК; б) зона активності оператора ІМК

Як видно з графіків статистичних оцінок ймовірнісних характеристик векторної ЕЕГ у часовому та спектральному діапазонах, навіть візуально можна помітити значні відмінності між зонами активності та пасивності, що вказує на чутливість цих ймовірнісних характеристик до дії ментального керуючого впливу оператора ІМК. Різниці в структурі характеристик між зонами активності та пасивності особливо виражені для моментних функцій вищого порядку компонент вектора ЕЕГ.

Також можна побачити значну схожість у часовій структурі статистичних оцінок ймовірнісних характеристик усіх компонент векторної ЕЕГ для зони активності оператора ІМК, що вказує на спільність закономірностей у цих структурах для різних областей функціонування мозку, що дозволяє проведенню

ефективної спільної статистичної оцінки ймовірнісних характеристик компонент векторної ЕЕГ і можливість координованої інтеграції інформації з різних давачів з метою підвищення точності виявлення дії ментального керуючого впливу оператора ІМК.

Загалом, вищенаведені результати вказують на адекватність та здатність розробленої ймовірнісної математичної моделі та ритмоадаптивних статистичних методів опрацювання вектора ЕЕГ сигналів, відображати дію ментального керуючого впливу оператора ІМК та можливість проведення високоточної процедури для її виявлення (детекції). Однак, оскільки існує багато таких характеристик і є значна схожість у їх часових та спектральних структурах, необхідно вибрати найбільш інформативні характеристики вектора ЕЕГ.

3.7 Процедури класифікації ЕЕГ сигналів в нейроінтерфейсних системах

Вибір відповідних класифікаторів є важливим етапом у процесі розробки ефективних нейроінтерфейсних систем, особливо коли мова йде про детекцію та розпізнавання сигналів ЕЕГ. Якість та точність класифікації безпосередньо впливають на здатність системи коректно інтерпретувати нейронні сигнали, що, в свою чергу, визначає ефективність взаємодії між користувачем та ІМК.

Для дослідження було вибрано наступні класифікатори:

— k-Nearest Neighbors (k-NN): цей алгоритм використовується для класифікації об'єктів на основі голосування найближчих сусідів. Він простий у використанні та часто ефективний для невеликих наборів даних [123].

— Linear SVM (Support Vector Machine): цей класифікатор максимізує відстань між різними класами, створюючи "гіперплощину" для розділення. Це забезпечує високу точність класифікації для лінійно розділимих датасетів [124].

— Decision Tree: дерева рішень класифікують об'єкти, рухаючись від кореня дерева до листків, які представляють класифікаційні мітки. Ці класифікатори легко інтерпретуються, але можуть бути схильні до перенавчання [125].

— Random Forest: як ансамбль дерев рішень, цей метод зменшує ризик перенавчання та забезпечує вищу стабільність та точність за рахунок використання великої кількості дерев і усереднення їх прогнозів [123, 125].

— Multilayer Perceptron (MLP): це основна форма нейронної мережі, що включає множину шарів з нейронами, з'єднаними вагами. MLP ефективний для виявлення складних нелінійних зв'язків в даних [126].

— Adaptive Boosting (AdaBoost): метод, що поєднує багато слабких класифікаторів для створення сильного класифікатора, використовуючи процес поступового "навчання" на помилках попередніх класифікаторів.

— Naive Bayes: цей класифікатор застосовує теорему Байєса для прогнозування ймовірності класифікації на основі попередніх знань та властивостей об'єктів. Він простий та часто ефективний в умовах великої розмірності даних [57].

— SIC: цей класифікатор, відомий як Статистичний Інтервальний Класифікатор (Statistical Interval Classifier). Методологія та деталі реалізації цього класифікатора ретельно описані в роботі [127], де наголошено на використанні довірчих інтервалів для нормованих циклів. Основна ідея методу полягає у виборі між двома статистичними гіпотезами, H_0 and H_1:

$$H_0: \xi_i(\omega, t) \notin \left(\hat{m}_{\xi_i}(t) - 3\hat{\sigma}_{\xi_i}, \hat{m}_{\xi_i}(t) + 3\hat{\sigma}_{\xi_i}\right), \tag{30}$$

$$H_1: \xi_i(\omega, t) \in \left(\hat{m}_{\xi_i}(t) - 3\hat{\sigma}_{\xi_i}, \hat{m}_{\xi_i}(t) + 3\hat{\sigma}_{\xi_i}\right). \tag{31}$$

Правило ухвалення рішення ґрунтується на визначенні параметру N_i - це частка (кількість) випадків, коли сигнал $\xi_i(\omega, t)$ потрапляє в заданий інтервал для i-го суб'єкта. Правило ухвалення рішення для SIC в контексті розпізнавання ментальних керуючих впливів за допомогою ЕЕГ виглядає наступним чином:

1) Відхилити гіпотезу H_0 і прийняти гіпотезу H_1, якщо N_i більше 95%,
2) Відхилити гіпотезу H_1 і прийняти гіпотезу H_0, якщо N_i менше чи рівне 95%.

Кожен з цих класифікаторів має свої переваги та обмеження, тому вибір конкретного методу залежить від характеристик датасету, цілей дослідження та

специфіки завдань, які необхідно вирішити. Використання різноманітних класифікаторів дозволяє провести порівняльний аналіз їх ефективності у контексті задач нейроінтерфейсу, що є ключовим для вибору оптимальної моделі для конкретного застосування.

Для об'єктивного порівняння та оцінювання ефективності різних класифікаторів використано матриці невідповідностей (Confusion Matrix). Матриця невідповідностей - це специфічна таблиця, що дозволяє візуалізувати ефективність алгоритму класифікації, представляючи кількість вірних та помилкових прогнозів, розділених за категоріями. Вона складається із чотирьох частин: істинно позитивних (TP), істинно негативних (TN), помилково позитивних (FP) та помилково негативних (FN) прогнозів.

Істинно позитивні (True Positives, TP) відповідності відбуваються, коли модель правильно прогнозує позитивний клас. Істинно негативні (True Negatives, TN) результати відбуваються, коли модель правильно ідентифікує негативний клас. Помилково позитивні (False Positives, FP) прогнози відбуваються, коли модель неправильно прогнозує позитивний клас. Помилково негативні (False Negatives, FN) результати відбуваються, коли модель неправильно прогнозує негативний клас.

Використання матриці невідповідностей дозволяє не тільки оцінити загальну точність моделі, але й детально проаналізувати її поведінку по відношенню до кожного класу. Наприклад, високе значення FP може вказувати на те, що модель занадто часто визначає відсутність характеристики як присутню, що може бути критичним у медичних застосуваннях. Водночас, високе значення FN вказує на проблему пропуску існуючих характеристик, що також може мати серйозні наслідки.

Метрики, такі як істинно позитивний рівень (TPR), істинно негативний рівень (TNR), помилково негативний рівень (FNR) та помилково позитивний рівень (FPR), є ключовими показниками, що випливають із матриці невідповідностей та дають змогу глибше зрозуміти ефективність класифікаційної моделі.

Істинно позитивний рівень, також відомий як чутливість, вимірює здатність моделі правильно ідентифікувати позитивні випадки серед усіх дійсно позитивних випадків.

$$TPR = \frac{TP}{TP + FN}. \quad (30)$$

Істинно негативний рівень, також відомий як специфічність, показує здатність моделі правильно ідентифікувати негативні випадки серед усіх дійсно негативних випадків.

$$TNR = \frac{TN}{TN + FP}. \quad (31)$$

Помилково негативний рівень вимірює частку дійсно позитивних випадків, які були помилково класифіковані як негативні.

$$FNR = \frac{FN}{TP + FN} = 1 - TPR. \quad (32)$$

Помилково позитивний рівень показує частку дійсно негативних випадків, які були помилково класифіковані як позитивні.

$$FPR = \frac{FP}{TN + FP} = 1 - TNR. \quad (33)$$

Ці метрики допомагають не лише в оцінюванні загальної ефективності класифікатора, але й у розумінні його поведінки у різних сценаріях. Наприклад, високий рівень TPR вказує на те, що модель добре виявляє позитивні випадки, тоді як високий рівень TNR свідчить про ефективність моделі в ідентифікації негативних випадків. Водночас, FNR та FPR дозволяють ідентифікувати слабкі сторони моделі, вказуючи на ситуації, коли модель неправильно класифікує позитивні чи негативні випадки.

Точність (Accuracy), міра F1 (F1 Score або Harmonic Mean of Precision and Sensitivity), та збалансована точність (Balanced Accuracy) є додатковими ключовими метриками для оцінювання ефективності класифікаційних моделей. Кожна з цих метрик надає унікальний вимір якості класифікації, дозволяючи глибше аналізувати та зрозуміти поведінку моделі.

Точність - це найбільш інтуїтивно зрозуміла метрика, яка вимірює частку вірно класифікованих випадків (як позитивних, так і негативних) серед усіх випадків у наборі даних.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (34)$$

Хоча точність є корисною для загальної оцінки, вона може бути оманливою в незбалансованих наборах даних, де кількість представників одного класу значно переважає над іншим.

Міра F1 - це гармонійне середнє між точністю і чутливістю, яке забезпечує кращий баланс між ними, особливо в умовах незбалансованих класів.

$$F1\ score = 2 * \frac{PPV * TPR}{PPV + TPR} = \frac{2\,TP}{2\,TP + FP + FN}. \quad (35)$$

Збалансована точність - це середнє значення між чутливістю (TPR) та специфічністю (TNR), яке забезпечує більш точну оцінку для незбалансованих наборів даних.

$$Balanced\ Accuracy = \frac{TPR + TNR}{2}. \quad (36)$$

Збалансована точність допомагає вирішити проблему оцінки моделей, де домінування одного класу може призвести до високих значень звичайної точності, незважаючи на погану класифікаційну здатність по відношенню до менш представленого класу.

Використання цих трьох метрик разом дає змогу глибше зрозуміти сильні та слабкі сторони моделі класифікації, а також забезпечує комплексний підхід до оцінювання її загальної ефективності та придатності для конкретних.

У працях [9, 115] було проведено оптимальний вибір ймовірнісних характеристик ЕЕГ сигналів в нейроінтерфейсних системах. Цей процес включав аналіз та порівняння різних характеристик з метою ідентифікації тих, які найкраще реагують на ментальний керуючий вплив оператора ІМК. Завдяки цьому було визначено набір характеристик, які мають найвищий рівень чутливості та інформативності, що значно підвищує точність виявлення ментальних впливів.

Найкращі результати класифікації спостерігаються для математичного сподівання, як в часовій області так і в спектральній області для 40 коефіцієнтів при розкладі в ряд Фур'є. Це підкреслює важливість спектрального аналізу даних ЕЕГ та його перевагу в ідентифікації ментальних команд, порівняно з часовим аналізом.

На рисунку 3.21 подано графік залежностей, які ілюструють різниці між спектральною та часовими областями, а також середньоквадратичні відхилення характеристик точності. Цей графік демонструє, що спектральні характеристики мають менші відхилення та більш виражену диференціацію між станами активності та пасивності, ніж аналогічні характеристики в часовій області.

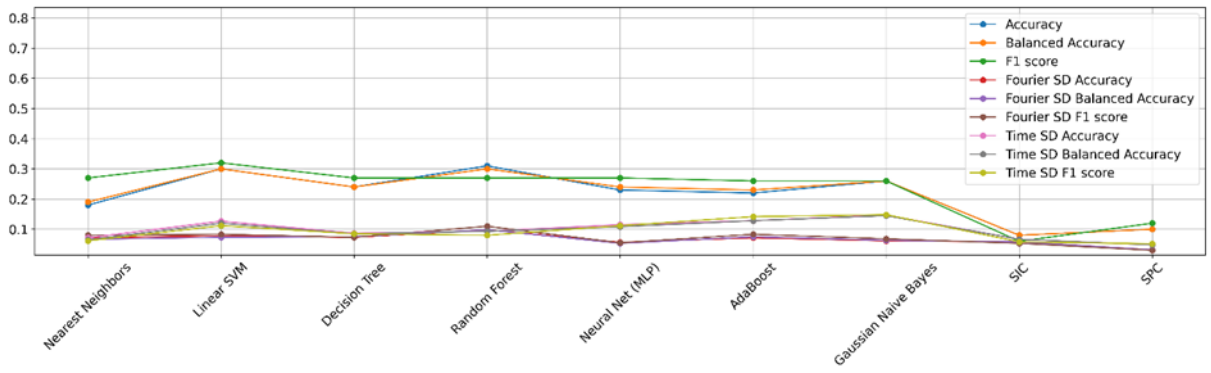


Рисунок. 3.21. Графік залежностей, що ілюструє різниці між спектральною та часовими областями, та середньоквадратичні відхилення цих характеристик

Варто зазначити, що класифікатор, заснований на методі SIC, продемонстрував однаково високу ефективність як в часовій, так і в спектральній областях. Це підкреслює універсальність даного класифікатора та його здатність адаптуватися до різних типів даних без втрати точності класифікації. Така особливість є важливою для розробки більш гнучких та ефективних ІМК систем, що можуть оперативно реагувати на змінні умови ментальної активності оператора.

Однак, слід відмітити, що використання розкладу сигналів ЕЕГ у ряд Фур'є значно підвищує точність ідентифікації ментальних керуючих впливів, водночас

збільшуючи обчислювальну складність. Цей аспект проілюстровано на рисунку 3.22. Таке збільшення обчислювальної складності є критичним чинником при проектуванні ІМК систем які працюють в реальному часі, де вимоги до швидкодії та ресурсоємності є дуже строгими.

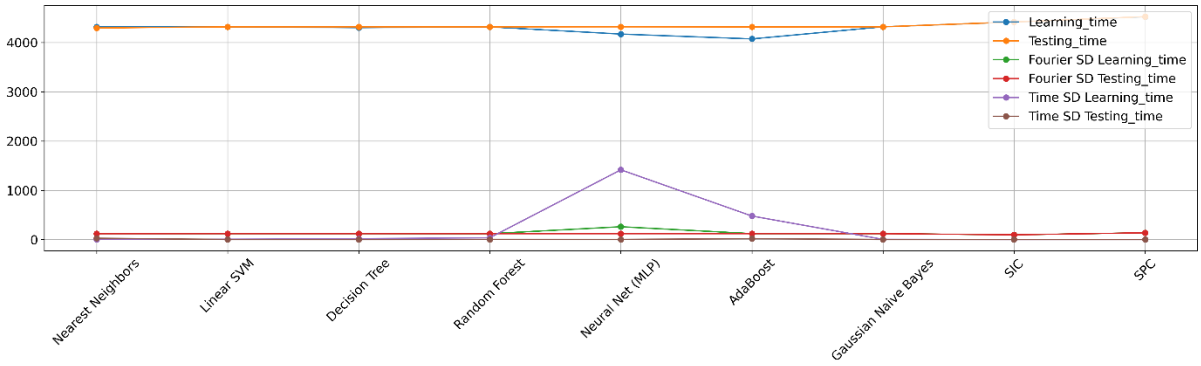


Рисунок. 3.22. Графік залежностей, що ілюструє різниці обчислювальної складності між спектральною та часовими областями, та середньоквадратичні відхилення цих характеристик

У контексті аналізу ефективності спектральних коефіцієнтів ряду Фур'є для класифікації ментальних керуючих впливів, важливим аспектом є визначення оптимальної кількості коефіцієнтів, яка забезпечує найвищу точність розпізнавання. В ході дослідження було проведено аналіз залежності ключових метрик ефективності класифікації — точності, міри F1 та збалансованої точності — від кількості спектральних коефіцієнтів ряду Фур'є.

Аналіз проведено на основі даних тренованих операторів (Група 1), результати яких представлені на рисунку 3.23 (точність), рисунку 3.24 (міра F1) та рисунку 3.25 (збалансована точність). Ці графіки ілюструють залежності вказаних метрик від кількості спектральних коефіцієнтів, використаних у розкладі.

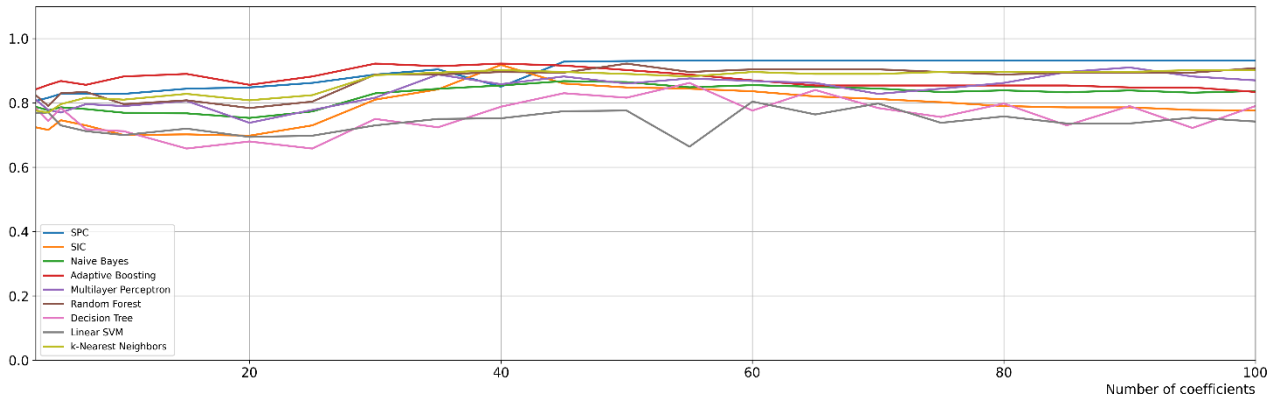


Рисунок. 3.23. Точність класифікації

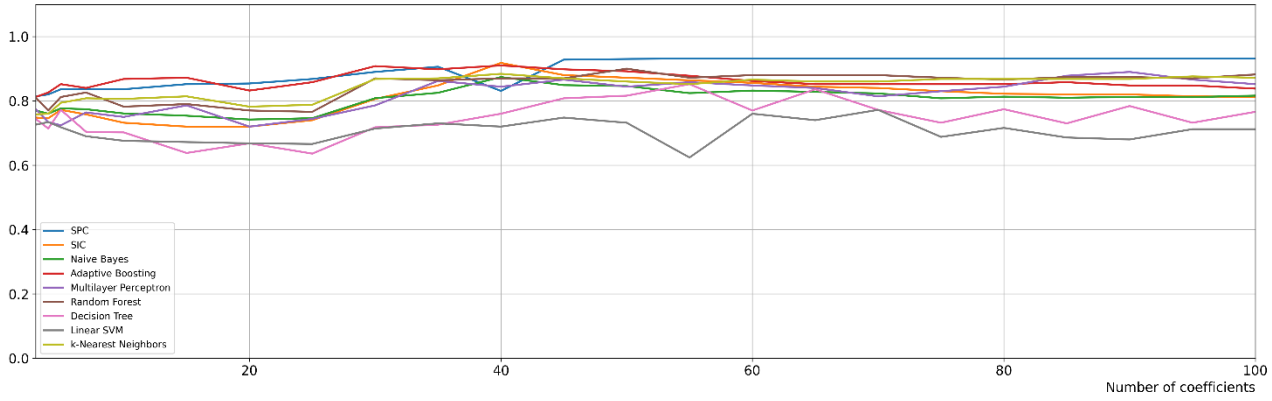


Рисунок. 3.24. Міра F1 класифікації

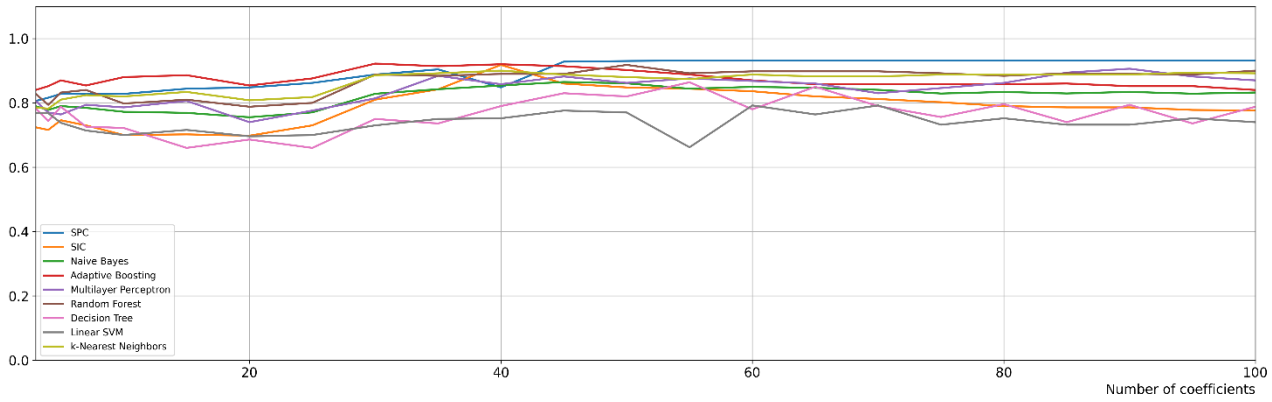


Рисунок. 3.25. Збалансована точність класифікації

Згідно з отриманими результатами, найоптимальніша кількість спектральних коефіцієнтів, яка забезпечує максимальну точність і баланс між враховуваною інформативністю спектра та обчислювальною складністю, становить 40. Використання саме цієї кількості коефіцієнтів дозволяє досягти значного поліпшення у точності класифікації порівняно з меншою кількістю коефіцієнтів, що підтверджується аналізом даних.

З іншого боку, варто зазначити, що зі збільшенням кількості коефіцієнтів після оптимальних 40, обчислювальна складність системи значно зростає. Це зумовлено необхідністю проведення більш складних обчислень для врахування додаткових коефіцієнтів при аналізі спектра. Однак, важливо відзначити те, що після досягнення певної значення — у нашому випадку, після включення 40 спектральних коефіцієнтів — спостерігається зниження точності класифікації. Це зниження може бути пов'язане з тим, що додаткові коефіцієнти, які враховуються після цієї точки, мають переважно шумовий характер (див. рисунок 3.10), що не сприяє покращенню ідентифікації ментальних керуючих впливів, а навпаки, призводить до перенавчання класифікаторів.

Це перенавчання свідчить про те, що класифікатор починає навчатися шумом або несуттєвою особливістю даних, що в кінцевому результаті погіршує його здатність до узагальнення на нових даних. Таким чином, вибір оптимальної кількості спектральних коефіцієнтів є критично важливою для покращення якості розпізнавання ментальних команд у системах інтерфейсу мозок-комп'ютер. Застосування 40 спектральних коефіцієнтів у ряду Фур'є виявилося найефективнішим варіантом, що водночас забезпечує високу точність класифікації та оптимальне співвідношення між обчислювальною складністю та ефективністю опрацювання ЕЕГ сигналів.

Тренованість оператора є ще одним ключовим фактором, що впливає на ефективність систем інтерфейс мозок-комп'ютер. Аналізуючи дані для нетренованих операторів, можна спостерігати, що при будь-якій кількості спектральних коефіцієнтів ряду Фур'є не відбувається значущого підвищення точності класифікації. Ці результати ілюстровані на окремих графіках для нетренованих операторів, представлених на рисунку 3.26 (точність), рисунку 3.27 (міра F1) та рисунку 3.28 (збалансована точність), де ментальні керуючі впливи класифікуються однаково низько за будь-яких умов.

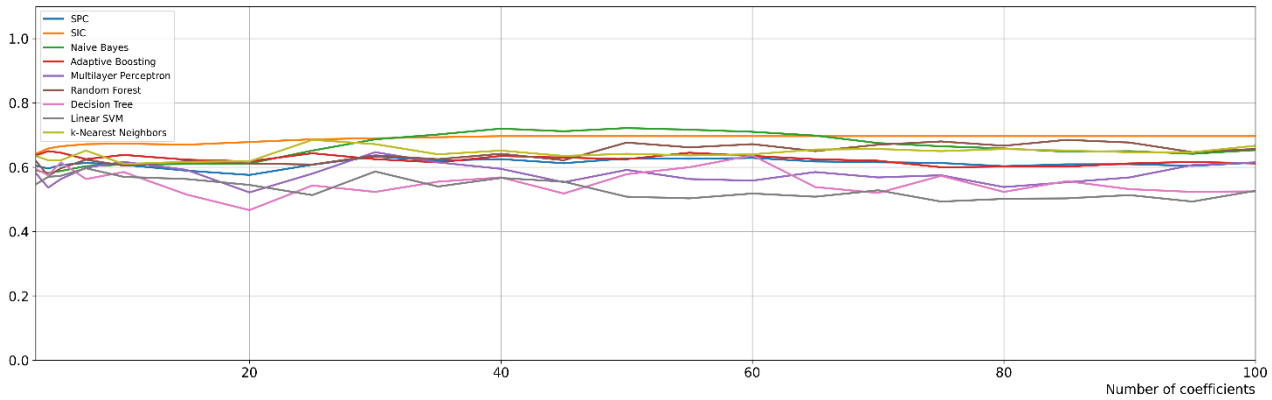


Рисунок. 3.26. Точність класифікації

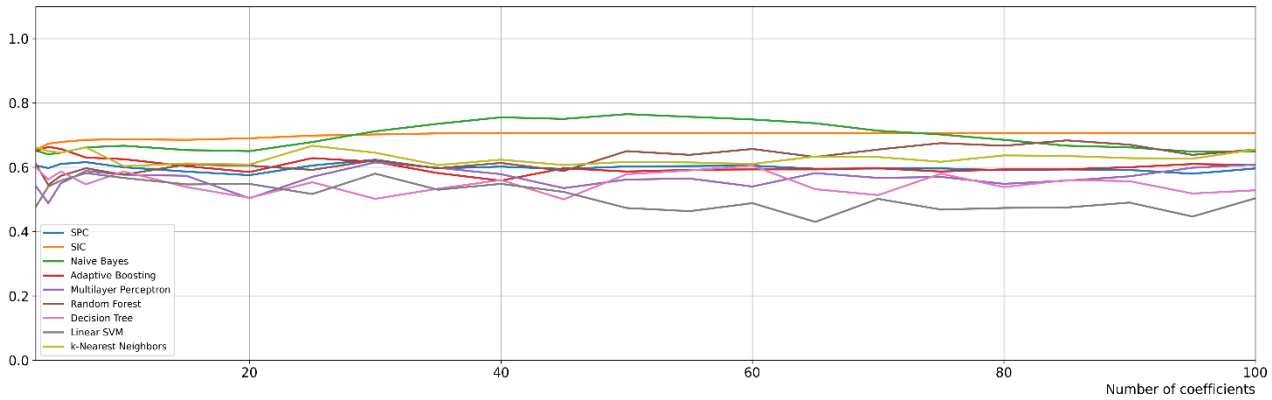


Рисунок. 3.27. Міра F1 класифікації

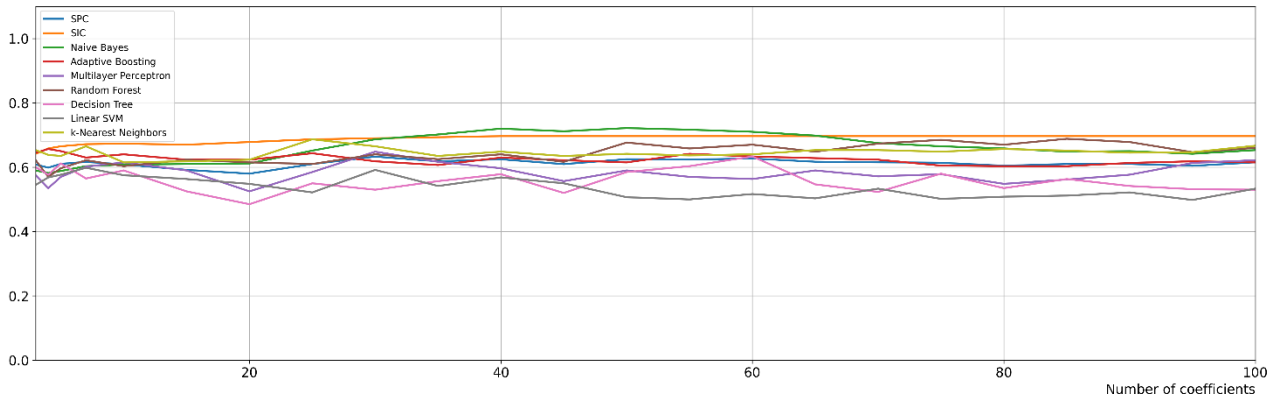


Рисунок. 3.28. Збалансована точність класифікації

Це підкреслює важливість попереднього тренування операторів ІМК систем, оскільки взаємодія між ментальними процесами людини та технологією вимагає певного рівня адаптації та навичок. Для нетренованих операторів ефективність виявлення ментальних команд залишається низькою. Це свідчить про те, що без відповідної підготовки здатність оператора генерувати чіткі та консистентні ментальні команди, які можуть бути ефективно розпізнані ІМК

системою, є обмеженою. Отже, тренування операторів стає невід'ємною частиною процесу впровадження та експлуатації ІМК систем, яке дозволяє значно підвищити точність та ефективність.

Як вже зазначалося вище, у рамках дисертаційної роботи було проведено порівняльний аналіз ефективності різних класифікаторів для ідентифікації ментальних команд на основі спектральних коефіцієнтів ряду Фур'є, зокрема для оптимальної кількості в 40 коефіцієнтів. Аналіз включав оцінку точності, збалансованої точності, міри F1, а також часу тренування та тестування різних типів класифікаторів, як для тренованих, так і для нетренованих операторів.

Для тренованих операторів, результати, представлені в таблиці 3.1, вказують на високу ефективність класифікаторів, зокрема Naive Bayes, який продемонстрував найвищу точність (0.932), збалансовану точність (0.934) та міру F1 (0.921). Це підтверджує, що деякі алгоритми машинного навчання можуть ефективно використовуватися для обробки даних ЕЕГ в контексті ІМК систем, особливо коли оператори мають досвід роботи з такими системами.

Таблиця 3.1. Оцінка ефективності різних класифікаторів для тренованих операторів ІМК при 40 спектральних коефіцієнтах

	Classifier type								
	SPC	SIC	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	0.916	0.888	0.791	0.908	0.802	0.781	0.908	0.848	0.932
Balanced Accuracy	0.916	0.888	0.782	0.904	0.802	0.781	0.904	0.851	0.934
F1 score	0.836	0.902	0.734	0.891	0.778	0.754	0.886	0.826	0.921
Training time (s)	4.502	4.543	4.233	4.239	4.239	4.279	6.272	4.623	4.234
Testing time (s)	4.516	4.558	4.237	4.229	4.229	4.231	4.232	4.255	4.229

Водночас, аналіз ефективності класифікаторів для нетренованих операторів, який представлено в таблиці 3.2, виявив значно нижчі показники по всіх критеріях. Наприклад, найвища точність серед нетренованих операторів була зафіксована для SIC класифікатора і становила 0.748, що є значно нижчим порівняно з тренованими операторами. Така різниця в ефективності класифікації підкреслює важливість тренування операторів для досягнення оптимальної роботи ІМК систем.

Таблиця 3.2. Оцінка ефективності різних класифікаторів для не тренованих операторів ІМК при 40 спектральних коефіцієнтах

	Classifier type								
	SPC	SIC	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	0.697	0.748	0.552	0.625	0.553	0.601	0.627	0.592	0.621
Balanced Accuracy	0.697	0.748	0.552	0.625	0.551	0.603	0.627	0.593	0.612
F1 score	0.707	0.781	0.471	0.597	0.528	0.585	0.602	0.571	0.598
Training time (s)	4.838	4.957	4.593	4.597	4.601	4.639	6.601	4.994	4.594
Testing time (s)	4.852	4.974	4.595	4.589	4.589	4.591	4.592	4.615	4.589

Ці дані свідчать про критичну роль тренування операторів у процесі взаємодії з ІМК системами, підтверджуючи, що навіть з використанням оптимізованої кількості спектральних коефіцієнтів, ефективність ментального керуючого впливу значно залежить від досвіду та адаптації користувача до системи.

3.8 Висновки до третього розділу

У третьому розділі розроблено методи та програмно-апаратні засоби прототипу неінвазивної нейроінтерфейсної системи, яка втілює розроблені у другому розділі дисертації математичну модель та методи статистичного опрацювання векторного ЕЕГ сигналу.

На основі серії експериментальних досліджень, що проведені із тренованими та нетренованими операторами ІМК, вперше обґрунтовано нові інформативні характеристики у вигляді моментних функцій ЕЕГ сигналів та нові інформативні ознаки в нейроінтерфейсних систем у вигляді спектральних коефіцієнтів (перші 40 спектральних коефіцієнтів) розкладів Фур’є статистичних оцінок моментних функцій ЕЕГ сигналів. Обґрунтування цих ознак здійснено шляхом зорієнтованості на забезпечення максимуму характеристик точності та мінімуму часової обчислювальної складності алгоритмів опрацювання ЕЕГ сигналів. Експериментально встановлено важливість попереднього тренування операторів для оптимізації взаємодії з ІМК системами

Обгрунтовано вибір оптимальних класифікаторів для задачі детекції ментального керуючого впливу опратора ІМК, що дозволило досягти високої точності детекції (розпізнавання) ментального керуючого впливу оператора, яка в середньому становить приблизно 93% для тренованих операторів ІМК.

РОЗДІЛ 4
МЕТОДИ ТА ПРОГРАМНО-АПАРАТНІ ЗАСОБИ ОПРАЦЮВАННЯ ЕКГ ТА СКГ СИГНАЛІВ В СИСЕМАХ БІОМЕТРИЧНОЇ АУТЕНТИФІКАЦІЇ ОСОБИ ТА МЕДИЧНОЇ ДІАГНОСТИКИ АРИТМІЙ

У четвертому розділі розглянуто методи опрацювання (сегментація, нормалізація, статистичне оцінювання та класифікація) ЕКГ та СКГ сигналів в системах біометричної аутентифікації особи та в системах медичної діагностики аритмій. Застосовано методи ритмоадаптивного статистичного оцінювання моментних функцій кардіосигналів як інформативних біометричних та кардіодіагностичних характеристик. Досліджено ефективність різних класифікаторів в задача аутентифікації та медичної діагностики.

Основні результати розділу опубліковано в працях [105, 127].

4.1. Основні етапи опрацювання кардіосигналів у системах біометричної аутентифікації особи

Біометрична аутентифікація особи за її кардіосигналами (ЕКГ, СКГ та ін.), поступово набуває популярності [128] завдяки своїм унікальним характеристикам. Зокрема, однією із ключових переваг ЕКГ як біометричного маркера є її високий рівень індивідуальної репрезентабельності, а саме, часова структура ЕКГ сигналів є унікальною для кожної людини, що дозволяє точно аутентифікувати особу. Крім того, відтворення або підробка таких біометричних сигналів є значною мірою ускладненою, що забезпечує додатковий рівень захисту в системах інформаційної безпеки. Ця унікальність та складність підробки роблять ЕКГ особливо цінним для застосувань, де критично важливі надійність і точність аутентифікації. Використання ЕКГ для біометричної ідентифікації не тільки підвищує безпеку доступу до приватних даних або обмежених фізичних просторів, але й відкриває нові можливості для персоналізації медичного обслуговування, фінансових послуг, та систем контролю доступу.

На рисунку 4.1 зображено ключові етапи обробки сигналу ЕКГ для біометричної аутентифікації, які включають: реєстр бази даних ЕКГ, сегментацію сигналу ЕКГ, нормалізацію сигналу ЕКГ, генерацію навчального та тестового наборів даних, а також класифікацію для задачі аутентифікації.

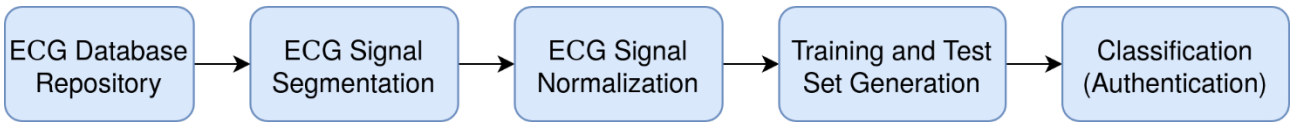


Рисунок. 4.1. Узагальнена структурна схема опрацювання сигналу ЕКГ у системах біометричної аутентифікації особи

Подібна ситуація має місце і для біометричної аутентифікації особи за кардіосигналами іншого типу, зокрема, за сейсмокардіограмами (СКГ).

Для проведення досліджень було використано базу даних Combined Measurement of ECG, Breathing, and Seismocardiograms (CEBS) [129, 130, 131], яка містить дані 20 здорових осіб. Для експерименту з аутентифікації акцент робився на сигнал ЕКГ з другого відведення та на СКГ сигнал. Обрані для цього дослідження пацієнти були 001-003, 005-017, 019-020. Для тренування та тестування класифікаторів використовувались 30-хвилинні записи ЕКГ та СКГ кожного пацієнта. Наприклад, перші цикли ЕКГ (відведення II) та СКГ з бази даних CEBS для особи 10, представлені на рисунку 4.2 та на рисунку 4.3.

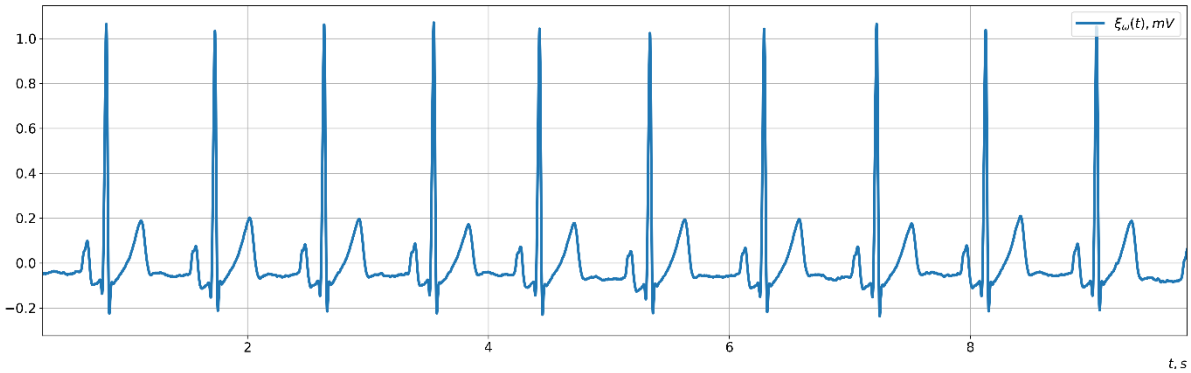


Рисунок. 4.2. Графік перших циклів ЕКГ для умовно здорового пацієнта

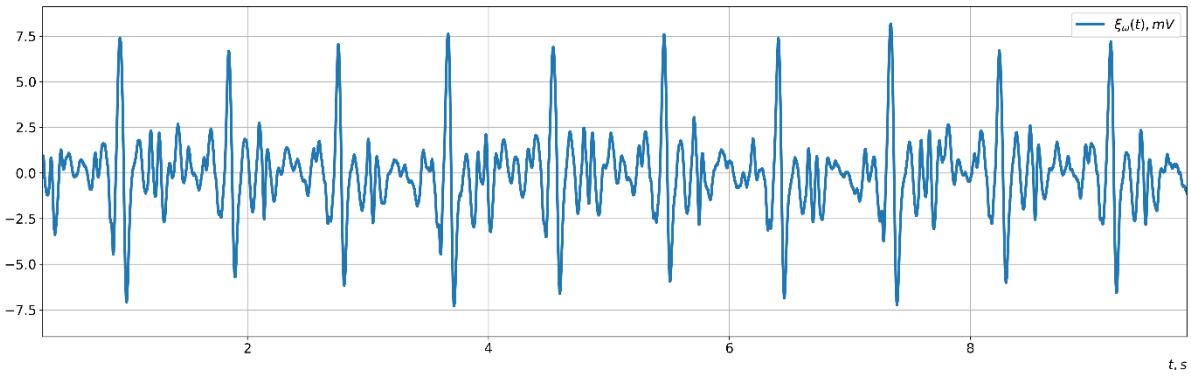


Рисунок. 4.3. Графік перших циклів СКГ для умовно здорового пацієнта

4.2. Математична модель та методи статистичного опрацювання ЕКГ та СКГ сигналів в системах біометричної аутентифікації

Адекватною математичною моделлю кардіосигналів різної фізичної природи є модель у вигляді циклічного випадкового процесу, яку обґрунтовано в працях [28, 29, 46-62]. Оскільки циклічний випадковий процес можна трактувати як компоненту вектора циклічних ритмічно пов’язаних випадкових процесів, то властивості циклічного випадкового процесу повністю витікають із властивостей вектора циклічних ритмічно пов’язаних випадкових процесів, які описано в другому розділі дисретації.

Ритмоадаптивні методи статистичного оцінювання характеристик ЕКГ та СКГ як циклічних випадкових процесів є частинним випадком ритмоадаптивних методів статистичного оцінювання ймовірнісних характеристик (моментних функцій) вектора циклічних ритмічно пов’язаних випадкових процесів, які також було описано у другому розділі дисертації. А саме, статистичне оцінювання початкових, центральних, змішаних моментних функцій ЕКГ та СКГ здійснюється на основі формул (23) – (27).

Оцінювання функції ритму цих циклічних кардіосигналів здійснюється на основі описаної в другому розділі дисертації кусково-лінійної інтерполяції дискретної функції ритму, яка у свою чергу, отримується на основі методів сегментування кардіосигналів на цикли та зони.

На рисунку 4.4 показано графік оціненої функції ритму ЕКГ для пацієнта 10, який було отримано на основі методу кусково-лінійної інтерполяції.

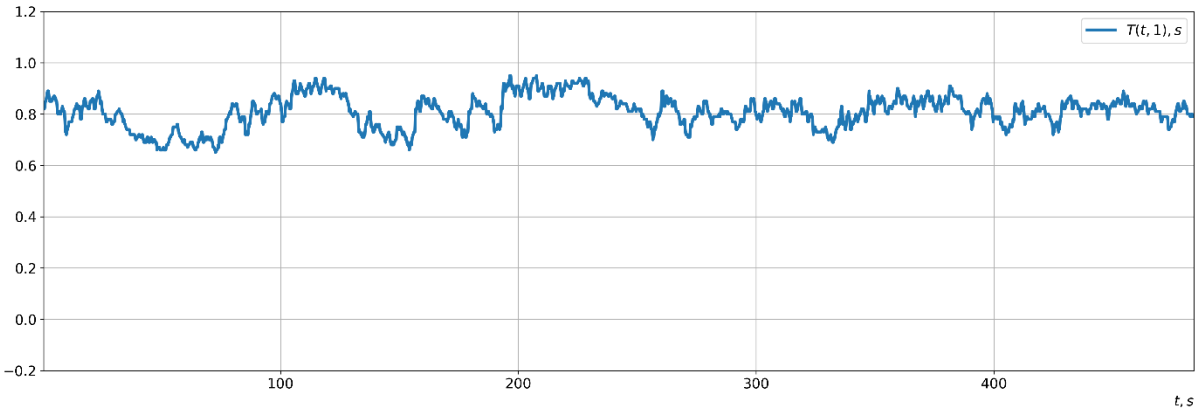


Рисунок. 4.4. Графік оцінки функції ритму ЕКГ для десятого пацієнта

На рисунках 4.5 та 4.6 наведено приклад сегментації ЕКГ та СКГ здорового пацієнта на цикли.

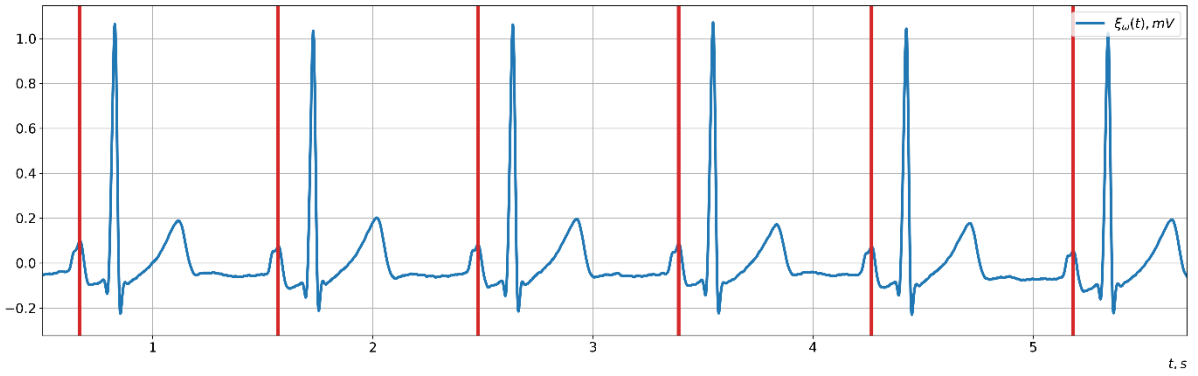


Рисунок. 4.5. ЕКГ і результат її сегментації на цикли

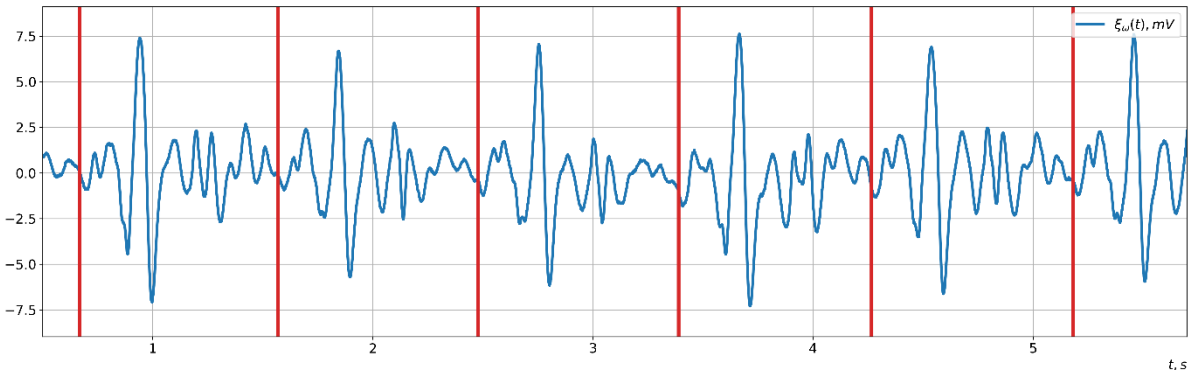


Рисунок. 4.6. СКГ і результат її сегментації на цикли

Перші цикли статистичних оцінок деяких початкових та центральних моментних функцій сигналів ЕКГ та СКГ представлені на рисунках 4.7-4.14. Реалізація статистичної оцінки $\hat{m}_{k_\xi}(t)$ початкової моментної функції k-го порядку $m_{k_\xi}(t)$ сигналів ЕКГ та СКГ, розраховується за формулою (24).

Формула (22) є обчислювальною формулою для реалізації статистичної оцінки $\hat{m}_\xi(t) = \hat{m}_{1_\xi}(t)$ математичного сподівання $m_\xi(t)$ ЕКГ та СКГ (див. рисунок 4.7 та рисунок 4.8).

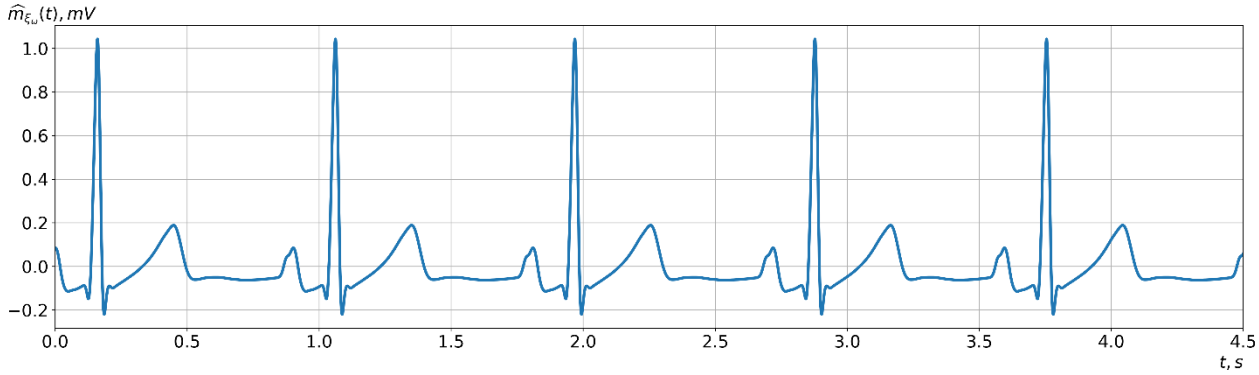


Рисунок. 4.7. Графік реалізації статистичної оцінки математичного сподівання ЕКГ при її опрацюванні на основі циклічного випадкового процесу

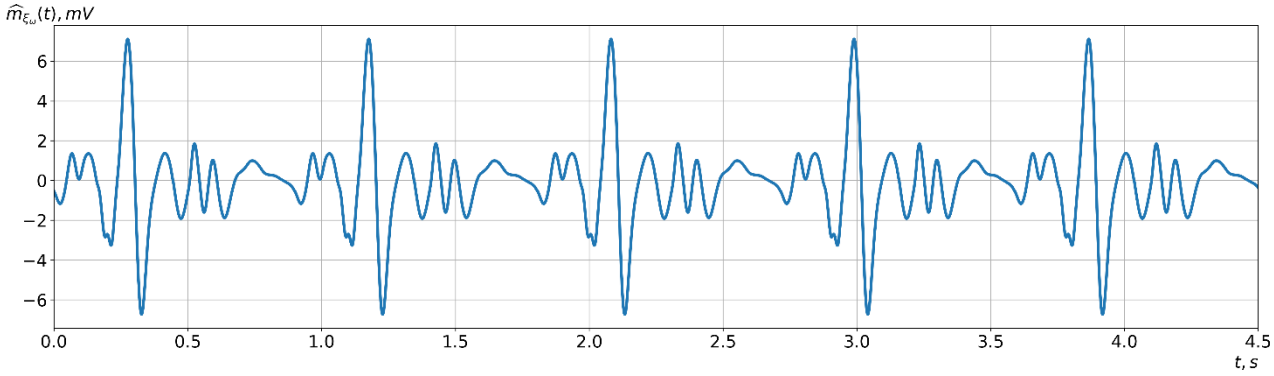


Рисунок. 4.8. Графік реалізації статистичної оцінки математичного сподівання СКГ при її опрацюванні на основі циклічного випадкового процесу

За умови, що *k=2*, формула (24) є обчислювальною формулою для реалізації статистичної оцінки $\hat{m}_{2_\xi}(t)$ початкової моментної функції 2-го порядку $m_{2_\xi}(t)$ ЕКГ та СКГ (див. рисунок 4.9 та рисунок 4.10).

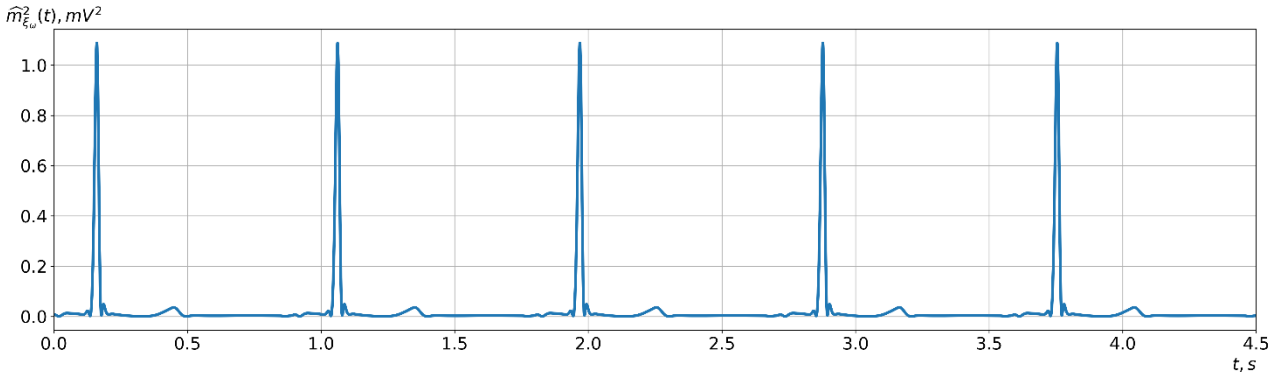


Рисунок. 4.9. Графік реалізації статистичної оцінки початкової моментної функції 2-го порядку ЕКГ при її опрацюванні на основі циклічного випадкового процесу

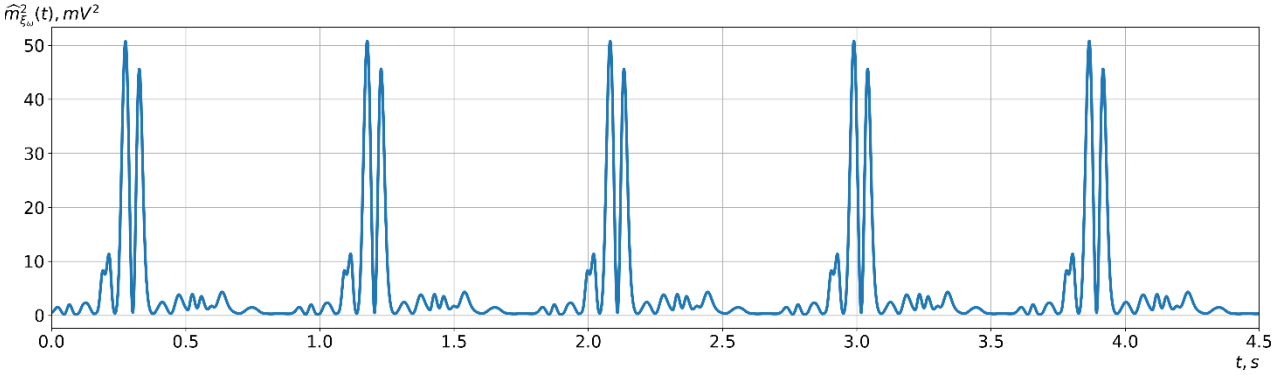


Рисунок. 4.10. Графік реалізації статистичної оцінки початкової моментної функції 2-го порядку СКГ при її опрацюванні на основі циклічного випадкового процесу

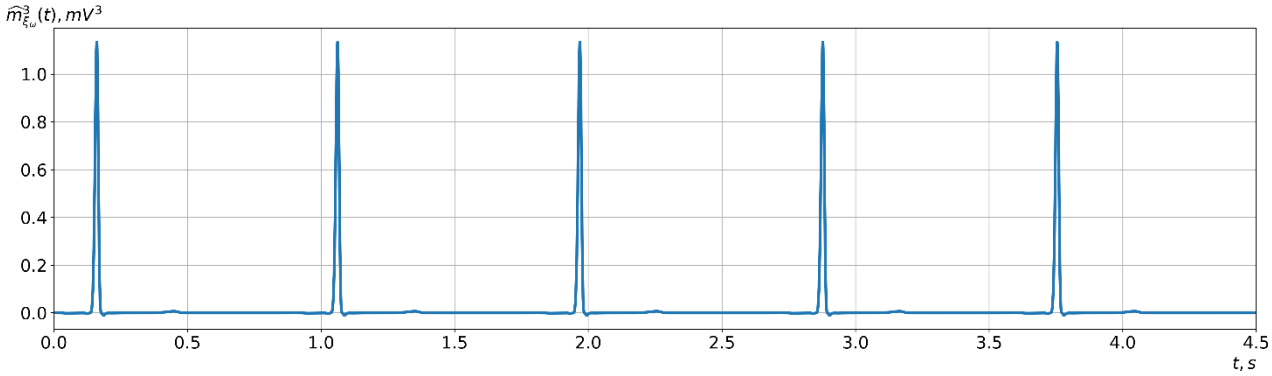


Рисунок. 4.11. Графік реалізації статистичної оцінки початкової моментної функції 3-го порядку ЕКГ при її опрацюванні на основі циклічного випадкового процесу

За умови, що $k=3$, формула (24) є обчислювальною формулою для реалізації статистичної оцінки $\widehat{m}_{3_\xi}(t)$ початкової моментної функції 3-го порядку $m_{3_\xi}(t)$ ЕКГ та СКГ (див. рисунок 4.11 та рисунок 4.12).

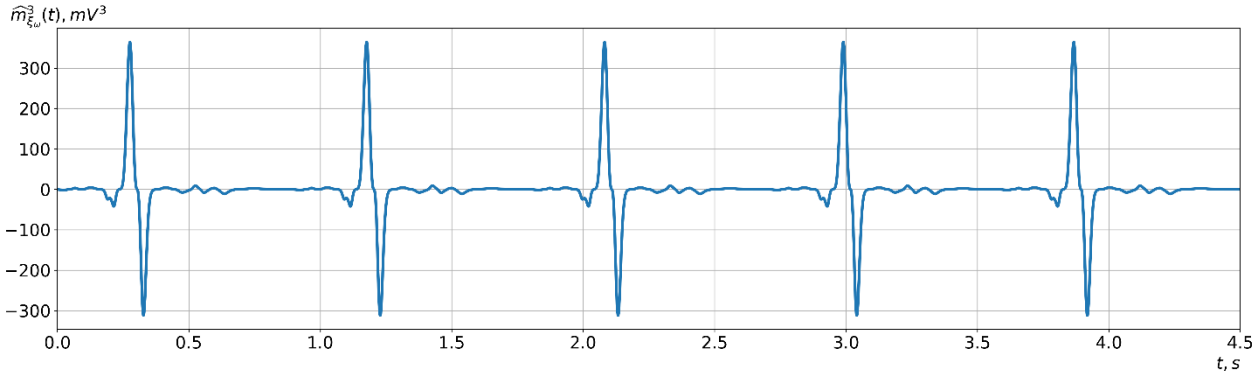


Рисунок. 4.12. Графік реалізації статистичної оцінки початкової моментної функції 3-го порядку СКГ при її опрацюванні на основі циклічного випадкового процесу

Реалізація статистичної оцінки $\hat{d}_{k_\xi}(t)$ центральної моментної функції *k-го* порядку $d_{k_\xi}(t)$ сигналів ЕКГ та СКГ, розраховується за формулою (25). За умови, що $k=2$, формула (25) є обчислювальною формулою для реалізації статистичної оцінки $\hat{d}_{2_\xi}(t)$ центральної моментної функції 2-го порядку (дисперсії) $d_{2_\xi}(t)$ ЕКГ та СКГ (див. рисунок 4.13 та рисунок 4.14).

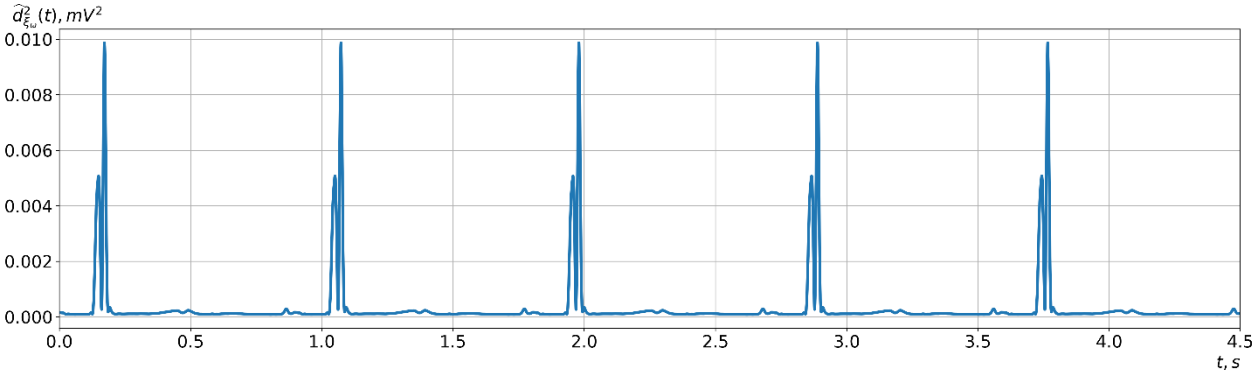


Рисунок. 4.13. Графік реалізації статистичної оцінки центральної моментної функції 2-го порядку (дисперсії) ЕКГ при її опрацюванні на основі циклічного випадкового процесу

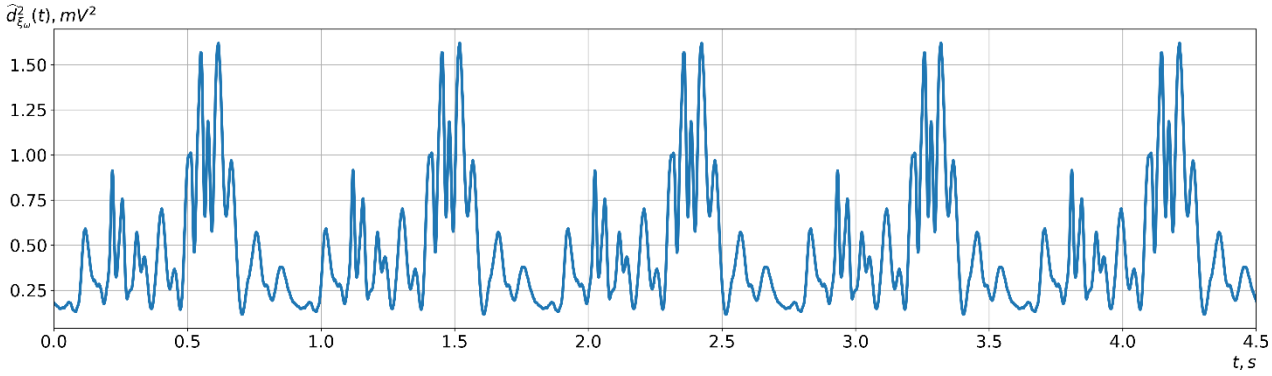
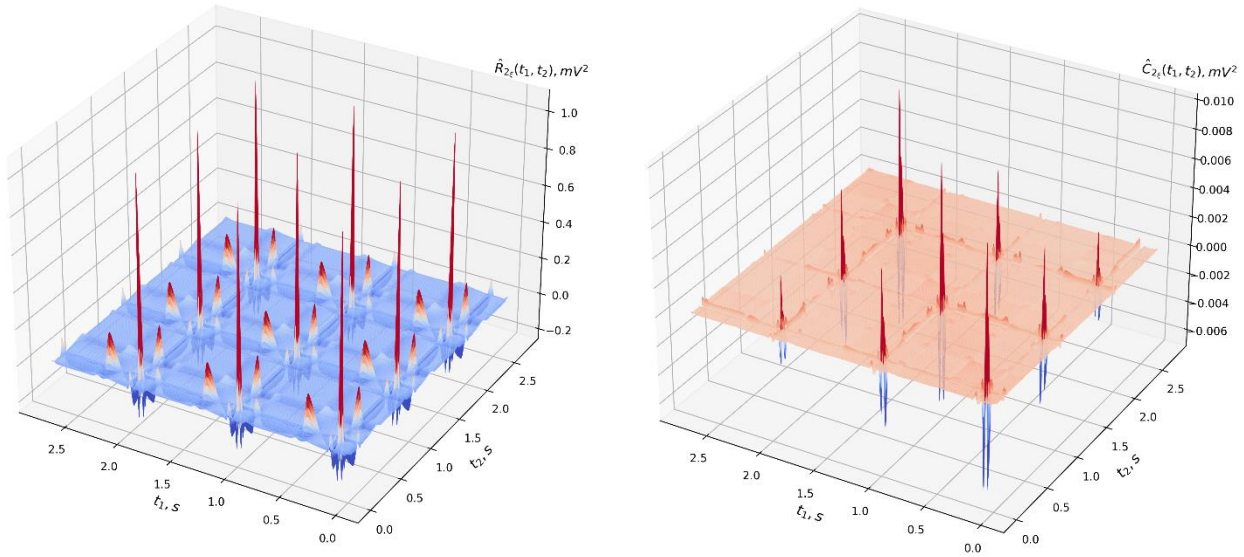


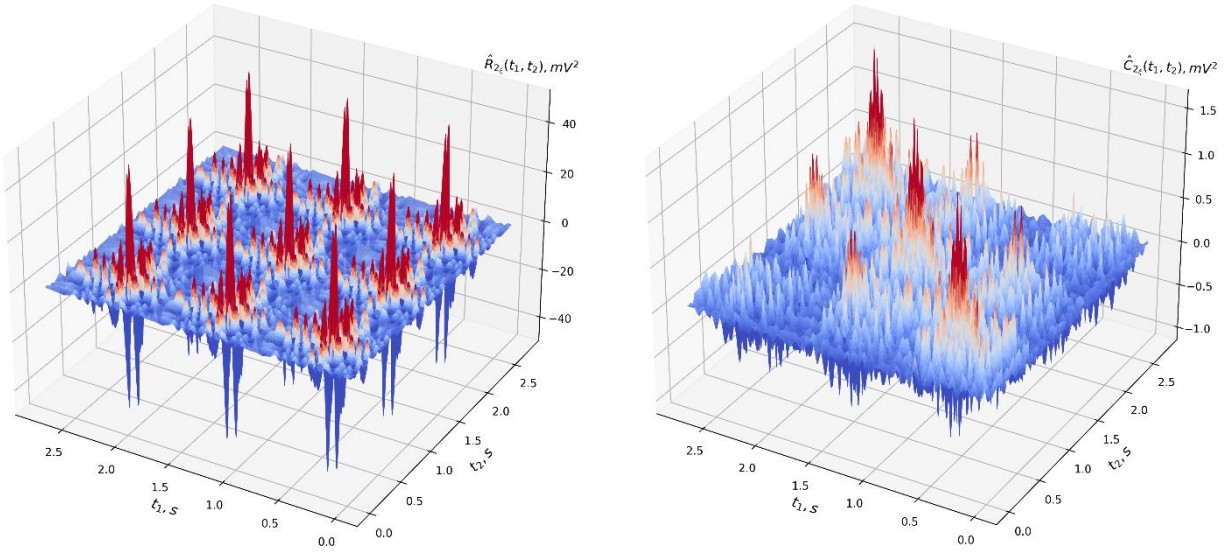
Рисунок. 4.14. Графік реалізації статистичної оцінки центральної моментної функції 2-го порядку (дисперсії) СКГ при її опрацюванні на основі циклічного випадкового процесу

Реалізація статистичної оцінки $\hat{R}_{2_\xi}(t_1, t_2)$ автокореляційної функції $R_{2_\xi}(t_1, t_2)$ сигналів ЕКГ та СКГ, розраховується за формулою (29). Реалізація статистичної оцінки $\hat{C}_{2_\xi}(t_1, t_2)$ автоковаріаційної функції $C_{2_\xi}(t_1, t_2)$ сигналів ЕКГ та СКГ, розраховується за формулою (28). Графіки реалізацій статистичних оцінок автокореляційної та автоковаріаційної функцій ЕКГ представлені на рисунку 4.15 та рисунку 4.16.



а) б)

Рисунок. 4.15. Графіки реалізації статистичних оцінок автокореляційної функції (а) та автоковаріаційної функції (б) ЕКГ при її опрацюванні на основі циклічного випадкового процесу



а) б)

Рисунок. 4.16. Графіки реалізації статистичних оцінок автокореляційної функції (а) та автоковаріаційної функції (б) СКГ при її опрацюванні на основі циклічного випадкового процесу

Наведені вище статистичні оцінки моментних функцій ЕКГ та СКГ є інформативними характеристиками в системах біометричної аутентифікації особи. На основі цих характеристик в системах біометричної аутентифікації можуть бути реалізовані процедури їх навчання та тестування.

4.3 Дослідження класифікаторів у системах біометричної аутентифікації за одним циклом ЕКГ та СКГ

З метою побудови експрес-методів біометричної аутентифікації особи розглянемо результати навчання та тестування різних класифікаторів на основі відомостей лише про один карідіоцикл особи. Коротко розглянемо цю процедуру.

Щоб формально відобразити циклічну структуру ЕКГ та СКГ, $\xi_i(\omega, t)$ для i-ої особи може бути представлено наступним чином:

$$\xi_i(\omega,t) = \sum_{m=1}^{M} \xi_{i,m}(\omega,t)\,, \omega \in \boldsymbol{\Omega},\ t \in \boldsymbol{W}, \tag{37}$$

де $\boldsymbol{W}$ — часова область визначення ЕКГ/СКГ, M — кількість зареєстрованих циклів в ЕКГ/СКГ.

Випадковий процес $\xi_{i,m}(\omega,t)$ збігається (є ідентичним) з випадковим процесом $\xi_i(\omega,t)$ на області $\boldsymbol{W}_{i,m}=[t_{m-1},t_m)$, яка відповідає m-му циклу ЕКГ/СКГ та яка є рівною:

$$\xi_{i,m}(\omega,t)=\xi_i(\omega,t)\cdot I_{\boldsymbol{W}_{i,m}}(t),\ \omega\in\boldsymbol{\Omega},\ t\in\boldsymbol{W},m=\overline{1,M}. \quad (38)$$

Функція $I_{\boldsymbol{W}_{i,m}}(t)$ — це індикаторна функція:

$$I_{\boldsymbol{W}_{i,m}}(t)=\begin{cases}1,\ t\in\boldsymbol{W}_{i,m},\\0,\ t\notin\boldsymbol{W}_{i,m}.\end{cases} \quad (39)$$

Часова область $\boldsymbol{W}$ може бути представлена як $\boldsymbol{W}=U_{m=1}^{M}\boldsymbol{W}_{i,m}$.

Результатом сегментації ЕКГ/СКГ є отримання набору сегментів-циклів $\{\xi_{i,m}(\omega,t),m=\overline{1,M}\}$, які розташовані на таких часових областях $\{\boldsymbol{W}_{i,m},m=\overline{1,M}\}$.

На рисунках 4.17, 4.18 показано приклад сегментації нормалізованих ЕКГ та СКГ здорового пацієнта на цикли.

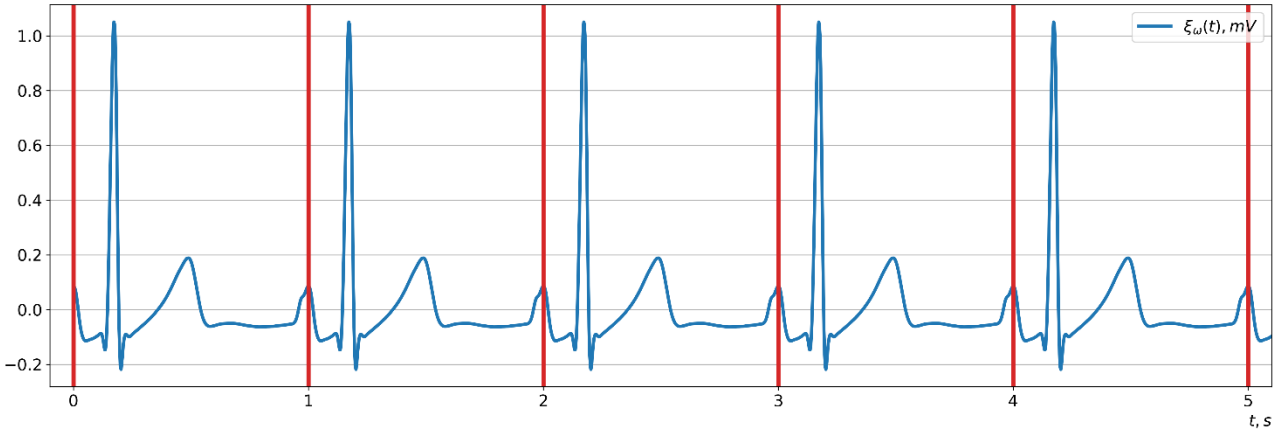


Рисунок. 4.17. ЕКГ і результат її сегментації на цикли

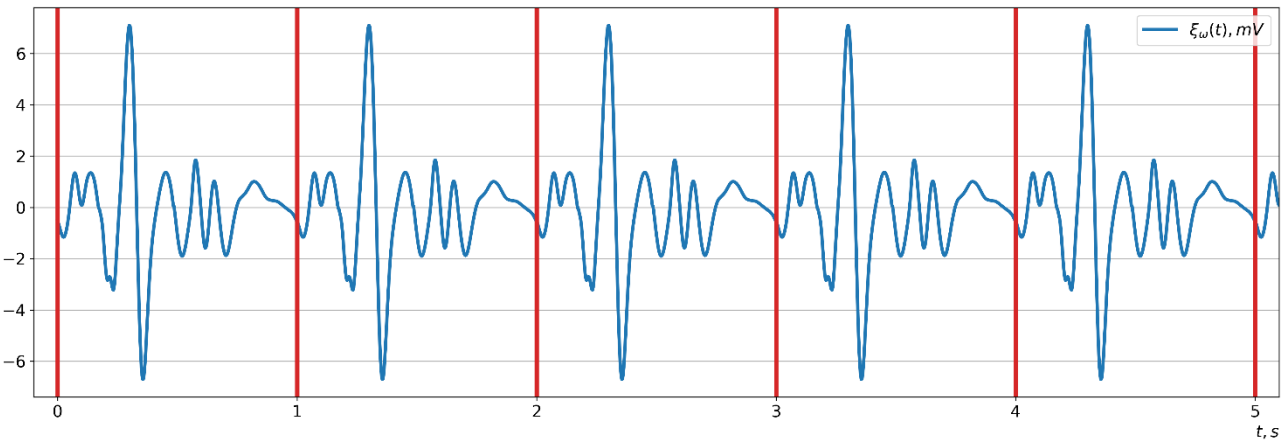


Рисунок. 4.18. СКГ і результат її сегментації на цикли

Для формування однорідних навчальних та тестових вибірок з багатоциклової ЕКГ/СКГ для кожного циклу $\xi_{i,m}(\omega,t)$ застосовують оператор статичного зсуву $\boldsymbol{G}_{s_{i,m}}[\cdot]$ та оператор статичного масштабування $\boldsymbol{G}_{\alpha_{i,m}}[\cdot]$, які забезпечують перенесення цього циклу з області $\boldsymbol{W}_{i,m}$ в одиничний інтервал [0,1], а саме:

$$\tilde{\xi}_{i,m}(\omega,t)=\boldsymbol{G}_{s_{i,m}}\left[\boldsymbol{G}_{\alpha_{i,m}}\left[\xi_{i,m}(\omega,t)\right]\right]=\xi_{i,m}\left(\omega,\alpha_{i,m}\cdot t+s_{i,m}\right),$$
$$t\in\boldsymbol{W}_{i,m}=\left[t_{i,m-1},t_{i,m}\right),m=\overline{1,M}, \tag{40}$$

де коефіцієнти статичного зсуву $s_{i,m}$ визначають за виразом:

$$s_{i,m}=-t_{i,m-1}, \tag{41}$$

та коефіцієнти статичного масштабування $\alpha_{i,m}$ визначають за виразом:

$$\alpha_{i,m}=\frac{1}{t_{i,m}-t_{i,m-1}}. \tag{42}$$

В результаті таких трансформацій зсуву та масштабування циклів ЕКГ/СКГ формується послідовність нормалізованих циклів $\left\{\tilde{\xi}_{i,m}(\omega,t),m=\overline{1,M}\right\}$.

Таблиця 4.1. Основні характеристики ефективності біометричної аутентифікації за одним циклом ЕКГ та часова обчислювальна складність алгоритмів для особи 10

	Classifier type							
	SIC	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	0.916	0.999	1.0	1.0	0.996	0.985	0.999	0.999
Balanced Accuracy	0.917	0.999	1.0	1.0	0.996	0.985	0.999	0.999
F1 score	0.917	0.999	1.0	1.0	0.996	0.986	0.999	0.999
Training time (ms)	3.47	18.13	48.27	57886.1	425.25	42.91	3558.57	4936.56
Testing time (ms)	12.17	138.83	4.77	508.55	3.34	5.67	8.072	38.91

Всі вісім класифікаторів, які були описані в третьому розділі дисертації, були навчені та протестовані на десяти умовно здорових особах, чиї ЕКГ містяться у базі даних CEBS. Основні характеристики ефективності біометричної аутентифікації та часова обчислювальна складність алгоритмів для навчання та тестування класифікаторів представлені у таблиці 4.1 для особи 10.

Таблиця 4.2 показує середні значення для 18 осіб — середні основні характеристики ефективності біометричної аутентифікації за одним циклом ЕКГ та часова обчислювальна складність алгоритмів для навчання та тестування класифікаторів.

Таблиця 4.2. Середні основні характеристики ефективності біометричної аутентифікації та часова обчислювальна складність алгоритмів

	Classifier type							
	SIC	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	0.953	0.999	1.0	1.0	0.996	0.984	1.0	1.0
Balanced Accuracy	0.952	0.999	1.0	1.0	0.996	0.984	1.0	1.0
F1 score	0.951	0.999	1.0	1.0	0.997	0.985	1.0	1.0
Training time (ms)	3.57	18.73	55.07	80149.1	332.64	43.54	3460.73	4367.72
Testing time (ms)	12.65	158.81	6.09	557.642	3.41	6.12	8.61	34.49

Як видно з таблиць 4.1 та 4.2, для всіх типів класифікаторів спостерігається висока ефективність аутентифікації особи за одним циклом ЕКГ. З іншого боку, класифікатори значно різняться за часовою обчислювальною складністю. Статистичний інтервальний класифікатор має незначний час навчання порівняно з іншими класифікаторами, що вказує на його перспективне використання в переносних портативних системах з обмеженими обчислювальними ресурсами.

Таблиця 4.3. Основні характеристики ефективності біометричної аутентифікації за одним циклом СКГ та часова обчислювальна складність алгоритмів для особи 10

	Classifier type							
	SIC	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	0.874	0.998	0.999	0.999	0.995	0.972	1.0	0.996
Balanced Accuracy	0.861	0.998	0.999	0.999	0.995	0.972	1.0	0.996
F1 score	0.876	0.998	0.999	0.999	0.995	0.972	1.0	0.996
Training time (ms)	3.412	18.28	57.23	109885.1	381.27	43.25	5753.53	5359.69
Testing time (ms)	12.32	143.05	8.37	529.65	3.31	9.44	8.12	39.48

Таблиця 4.4 показує середні значення для 18 осіб — середні основні характеристики ефективності біометричної аутентифікації за одним циклом СКГ та часова обчислювальна складність алгоритмів для навчання та тестування класифікаторів.

Таблиця 4.4. Середні основні характеристики ефективності біометричної аутентифікації та часова обчислювальна складність алгоритмів

	Classifier type							
	SIC	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	0.891	0.996	0.995	0.997	0.987	0.978	0.997	0.996
Balanced Accuracy	0.853	0.996	0.995	0.997	0.987	0.979	0.997	0.997
F1 score	0.893	0.996	0.995	0.997	0.987	0.978	0.997	0.996
Training time (ms)	4.02	22.12	79.51	57489.2	432.04	49.02	5896.69	5673.18
Testing time (ms)	15.11	150.11	11.53	593.211	3.81	6.69	9.91	47.47

Як видно із таблиць 4.1 - 4.4 розроблений експрес-метод біометричної аутентифікації особи за ЕКГ та СКГ має високі показники ефективності, що вказує на перспективність його використання в практичних аутентифікаційних задачах.

4.4 Дослідження класифікаторів в системах медичної діагностики аритмій за ЕКГ

Обґрунтовані вище методи ритмоадаптивного опрацювання ЕКГ сигналів можуть бути застосовані і для задач медичної діагностики стану серця, зокрема, для виявлення аритмій. Основні характеристики ефективності та часова обчислювальна складність алгоритмів для навчання та тестування класифікаторів у системах медичної діагностик аритмій за ЕКГ представлені у таблицях 4.5-4.7.

Таблиця 4.5. Середні показники ефективності діагностики аритмії, на базі оцінки математичного сподівання ЕКГ, та часова обчислювальна складність використаних алгоритмів класифікації

	Classifier type							
	SIC	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Balanced Accuracy	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
F1 score	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Training time (ms)	1.61	4.73	4.92	141.71	15.13	53.22	638.11	19.53
Testing time (ms)	11.38	3.71	1.45	4.07	3.71	6.96	3.76	4.16

Таблиця 4.6. Середні показники ефективності діагностики аритмії для сегментованих згідно функції ритму первинних даних та часова обчислювальна складність використаних алгоритмів класифікації

	Classifier type						
	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	0.813	1.0	0.625	0.875	1.0	1.0	0.937
Balanced Accuracy	0.751	1.0	0.501	0.833	1.0	1.0	0.917
F1 score	0.667	1.0	0.494	0.801	1.0	1.0	0.909
Training time (ms)	37.83	191.92	908.62	214.14	57.32	3078.39	249.14
Testing time (ms)	25.05	32.44	46.441	5.49	7.39	13.41	5.82

Таблиця 4.7. Середні показники ефективності діагностики аритмії на основі оцінки математичного сподівання в спектральній області з 30 коефіцієнтами ряду Фур'є та часова обчислювальна складність використаних алгоритмів класифікації

	Classifier type						
	k-Nearest Neighbors	Linear SVM	Decision Tree	Random Forest	Multilayer Perceptron	Adaptive Boosting	Naive Bayes
Accuracy	0.938	1.0	1.0	1.0	0.813	1.0	1.0
Balanced Accuracy	0.951	1.0	1.0	1.0	0.851	1.0	1.0
F1 score	0.923	1.0	1.0	1.0	0.801	1.0	1.0
Training time (ms)	384.62	384.45	492.21	386.25	426.59	720.23	386.71
Testing time (ms)	165.62	164.44	165.29	165.03	168.45	164.49	165.03

Як видно із таблиць 4.5 - 4.7 розроблений метод виявлення аритмій за ЕКГ має високі показники ефективності, що вказує на перспективність його використання в практичних задачах медичної діагностики.

4.5 Висновки до четвертого розділу

У четвертому розділі обґрунтовано математичну модель та ритмоадаптивні методи статистичного оцінювання інформативних характеристик кардіосигналів для вирішення задач біометричної аутентифікації особи за ЕКГ та СКГ сигналами, а також для вирішення задач медичної діагностики аритмій за ЕКГ сигналами. А саме, ґрунтуючись на теорій циклічних випадкових процесів як адекватних структурі кардіосигналів математичних моделей та методів їх статистичного опрацювання, розроблено методи та програмні засоби ефективного аналізу та класифікації ЕКГ та СКГ сигналів в системах біометричної аутентифікації та медичної діагностки аритмій.

Продемонстровано ефективне ритмоадаптивне статистичне оцінювання ймовірнісних характеристик ЕКГ та СКГ сигналів. Ці оцінки обґрунтовано як можливі інформативні характеристики для навчання та тестування систем біометричної аутентифікації та систем медичної діагностки аритмій.

Обчислювальні експерименти з використанням різних класифікаторів підтвердили високу точність та надійність методів біометричної аутентифікації особи на основі ЕКГ, демонструючи їхню придатність для застосування в системах контролю доступу та інших областях, де критично важлива висока надійність аутентифікації осіб. Також встановлено, що розроблений метод виявлення аритмій за ЕКГ має високі показники ефективності, що вказує на перспективність його використання в практичних задачах медичної діагностики.

ВИСНОВКИ

У дисертаційній роботі розв’язано актуальне наукове завдання, що має істотне значення для розвитку математичних моделей та методів ефективного опрацювання циклічних біомедичних сигналів в сучасних неівазивних нейроінтерфейсах, системах медичної діагностики та системах біометричної аутентифікації особи за її фізіологічними сигналами циклічної структури. Основні результати та висновки проведених теоретичних та експериментальних досліджень полягають у вирішенні таких задач:

1. Для сукупності ЕЕГ сигналів із різних відведень, зареєстрованих в умовах багаторазового повторення ментальних керуючих впливів оператора неінвазивного нейроінтерфейсу, побудовано нову математичну модель у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів, яка завдяки одночасному врахуванню стохастичності та циклічності, мінливості та спільності ритму досліджуваних біосигналів надала можливість досліджувати широкий клас ймовірнісних та статистистичних характеристик ЕЕГ сигналів в задачах розробки ефективних неінвазивних нейроінтерфейсних систем, що суттєво збагатило (доповнило) множину потенційно інформативних та чутливих до ментального впливу оператора нейроінтерфейсу характеристик сукупності ЕЕГ сигналів.

2. Обґрунтовано та верифіковано нові методи опрацювання сукупності ЕЕГ сигналів в неінвазивних нейроінтерфейсних системах, що уможливило розвиток ефективних (з точки зору характеристик точності та часової обчислювальної складності) інформаційних технологій детекції в структурі ЕЕГ сигналів ментальних керуючих впливів оператора нейроінтерфейсу. Зокрема, такі характеристики як Accuracy, Balanced Accuracy та F1-score для всіх досліджуваних класифікаторів у середньому є не нижчою як 96% для групи тренованих операторів.

3. Розроблено ритмоадаптивний метод біометричної аутентифікації особи за її ЕКГ, що ґрунтується на математичній моделі ЕКГ у вигляді циклічного

випадкового процесу. Основні характеристик точності цього методу аутентифікації, зокрема, Accuracy, Balanced Accuracy та F1-score, в середньому, не є нижчими ніж 99,2% для всіх досліджуваних осіб та для всіх типів бінарних класифікаторів (k-Nearest Neighbors, Linear SVM, Decision Tree, Random Forest, Multilayer Perceptron, Adaptive Boosting, Naive Bayes та Statistical Interval Classifier), які використовувалися для процедури класифікації, що вказує на високу точність такої аутентифікації.

4. Вперше, на основі серії проведених вимірювальних та обчислювальних експериментів, обґрунтовано оптимальні (з точки зору характеристик точності та часової обчислювальної складності) вектори інформативних ознак в нейроінтерфейсних системах та в системах біометричної аутентифікації особи за ЕКГ. А саме, використання лише перших 40 спектральних коефіцієнтів розкладу статистичних оцінок початкових моментних функцій ЕЕГ та ЕКГ сигналів у ряд Фур'є забезпечує найвищу точність (близько 93%) детекції ментального керуючого впливу опаратора неінвазивного інтерфейсу та найвищу точність (близько 99,9%) біометричної аутентифікації особи за її ЕКГ сигналом.

5. Розроблені у дисертації нові математичні моделі та ефективні методи опрацювання циклічних біомедичних сигналів втілено у систему комп'ютерних програм на мові Python, що дає змогу автоматизувати всі основні етапи (фільтрація завад, оцінювання ймовірнісних характеристик, спектральні розклади, класифікація) цифрового опрацювання ЕЕГ та ЕКГ сигналів в неінвазивних нейроінтерфейсних системах та системах біометричної аутентифікації особи за ЕКГ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lupenko, S.; Butsiy, R.; Shakhovska, N. Advanced Modeling and Signal Processing Methods in Brain–Computer Interfaces Based on a Vector of Cyclic Rhythmically Connected Random Processes. Sensors 2023, 23, 760. doi:https://doi.org/10.3390/s23020760
2. Butsiy, R.; Lupenko, S. Comparative Analysis of Neurointerface Technologies for the Problem of Their Reasonable Choice in Human-Machine Information Systems. Sci. J. TNTU (Tern.), 2020, 4, 135–148. doi:https://doi.org/10.33108/visnyk_tntu2020.04.135
3. Kawala-Sterniuk A, Pelc M, Martinek R and Wójcik GM (2022) Editorial: Currents in biomedical signals processing—methods and applications. Front. Neurosci. 16:989400. https://doi.org/10.3389/fnins.2022.989400
4. Kim, B.-H.; Pyun, J.-Y. ECG Identification For Personal Authentication Using LSTM-Based Deep Recurrent Neural Networks. Sensors 2020, 20, 3069. https://doi.org/10.3390/s20113069
5. Agrawal, V.; Hazratifard, M.; Elmiligi, H.; Gebali, F. Electrocardiogram (ECG)-Based User Authentication Using Deep Learning Algorithms. Diagnostics 2023, 13, 439. https://doi.org/10.3390/diagnostics13030439
6. Khan, M.U.; Aziz, S.; Iqtidar, K.; Saud, A.; Azhar, Z. Biometric Authentication System Based on Electrocardiogram (ECG). In Proceedings of the 2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS), Karachi, Pakistan, 2019; 1–6. https://doi.org/10.1109/MACS48846.2019.9024820
7. Prakash, A.J.; Patro, K.K.; Samantray, S.; Pławiak, P.; Hammad, M. A Deep Learning Technique for Biometric Authentication Using ECG Beat Template Matching. Information 2023, 14, 65. https://doi.org/10.3390/info14020065
8. Shih, J.J.; Krusienski, D.J.; Wolpaw, J.R. Brain-Computer Interfaces in Medicine. Mayo Clinic Proceedings, 2012, 87, 268-279. https://doi.org/10.1016/j.mayocp.2011.12.008

9. Butsiy, R.; Lupenko, S. Comparison of Modern Methods of Classification of EEG Patterns for Neurointerface Systems. In: Yang, XS., Sherratt, S., Dey, N., Joshi, A. (eds) Proceedings of Seventh International Congress on Information and Communication Technology. Lecture Notes in Networks and Systems; Springer, Singapore, 2022, 465, 345-354. https://doi.org/10.1007/978-981-19-2397-5_32
10. Lotte, F.; Bougrain, L.; Cichocki, A.; Clerc, M.; Congedo, M.; Rakotomamonjy, A.; Yger, F. A Review of Classification Algorithms for EEG-Based Brain–Computer Interfaces: A 10 Year Update. J. Neural Eng. 2018, 15, 031005. https://doi.org/10.1088/1741-2552/aab2f2
11. Mihajlović, V.; Grundlehner, B.; Vullers, R.; Penders, J. Wearable, Wireless EEG Solutions in Daily Life Applications: What Are We Missing?. IEEE J. Biomed. Health Inform. 2015, 19, 6-21. https://doi.org/10.1109/JBHI.2014.2328317
12. Rashid, M.; Sulaiman, N.; Abdul Majeed, A.P.P.; Musa, R.M.; Ab. Nasir, A.F.; Bari, B.S.; Khatun, S. Current Status, Challenges, and Possible Solutions of EEG-Based Brain-Computer Interface: A Comprehensive Review. Front. Neurorobot. 2020, 14. https://doi.org/10.3389%2Ffnbot.2020.00025
13. Li, S., Lyu, T., Chen, M., & Zhang, P. (2022). The Prospects of Using EEG in Tourism and Hospitality Research. Journal of Hospitality & Tourism Research, 46(1), 189-211. http://dx.doi.org/10.1177/1096348021996439
14. Plataniotis, K.N.; Hatzinakos, D.; Lee, J.K.M. ECG Biometric Recognition Without Fiducial Detection. In Proceedings of the 2006 Biometrics Symposium: Special Session on Research at the Biometric Consortium Conference, Baltimore, MD, USA, 2006. 1-6. https://doi.org/10.1109/BCC.2006.4341628
15. Wübbeler, G.; Stavridis, M.; Kreiseler, D.; Bousseljot, R.-D.; Elster, C. Verification of Humans Using the Electrocardiogram. Pattern Recognit. Lett. 2007, 28, 1172-1175. https://doi.org/10.1016/j.patrec.2007.01.014
16. Pereira, T.M.C.; Conceição, R.C.; Sebastião, R. Initial Study Using Electrocardiogram for Authentication and Identification. Sensors 2022, 22, 2202. https://doi.org/10.3390/s22062202

17. Pathoumvanh, S.; Airphaiboon, S.; Hamamoto, K. Robustness Study of ECG Biometric Identification in Heart Rate Variability Conditions. IEEJ Trans. Elec. Electron. Eng. 2014, 9, 294-301. http://dx.doi.org/10.1002/tee.21970
18. Eberz, S.; Paoletti, N.; Roeschlin, M.; Patané, A.; Kwiatkowska, M.; Martinovic, I. Broken Hearted: How To Attack ECG Biometrics. NDSS Symposium 2017. 2017. http://dx.doi.org/10.14722/ndss.2017.23408
19. Alves, A.; Carreiras, C. CardioWheel: ECG Biometrics on the Steering Wheel. In Proceedings of the European Conference: Machine Learning and Knowledge Discovery in Databases, Porto, Portugal, 7–11 September 2015; pp. 267–270.
20. Elgendi, M.; Eskofier, B.; Dokos, S.; Abbott, D. Revisiting QRS Detection Methodologies for Portable, Wearable, Battery-Operated, and Wireless ECG Systems. PLoS ONE 2014, 9(1), e84018. https://doi.org/10.1371/journal.pone.0084018
21. Egila, M.G.; El-Moursy, M.A.; El-Hennawy, A.E.; El-Simary, H.A.; Zaki, A. FPGA-Based Electrocardiography (ECG) Signal Analysis System Using Least-Square Linear Phase Finite Impulse Response (FIR) Filter. J. Electr. Syst. Inf. Technol. 2016, 3, 513-526. https://doi.org/10.1016/j.jesit.2015.07.001
22. Anapagamini, S.A.; Rajavel, R. Hardware Implementation of ECG Denoising System Using TMS320C6713 DSP Processor. Int. J. Biomed. Eng. Technol. 2016, 21, 95-110. http://dx.doi.org/10.1504/IJBET.2016.076735
23. Al-Khammasi, S.; Aboalayon, K.A.I.; Daneshzand, M.; Faezipour, M.; Faezipour, M. Hardware-Based FIR Filter Implementations for ECG Signal Denoising: A Monitoring Framework from Industrial Electronics Perspective. 2016 Annual Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA), Bridgeport, CT, USA, 2016, pp. 1-6. https://doi.org/10.1109/CT-IETA.2016.7868243
24. Denoising and Beat Detection of ECG Signal by Using FPGA. Int. J. High Speed Electron. Syst. 2017, 26(03), 1740016. http://dx.doi.org/10.1142/S012915641740016X

25. Ali, O.; Saif-ur-Rehman, M.; Dyck, S. et al. Enhancing the Decoding Accuracy of EEG Signals by the Introduction of Anchored-STFT and Adversarial Data Augmentation Method. Sci. Rep. 2022, 12, 4245. https://doi.org/10.1038/s41598-022-07992-w
26. Survey on the Research Direction of EEG-Based Signal Processing. Front. Neurosci. 2023, 17. https://doi.org/10.3389/fnins.2023.1203059
27. Bernal, G.; Hidalgo, N.; Russomanno, C.; Maes, P. Galea: A Physiological Sensing System for Behavioral Research in Virtual Environments. 2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), Christchurch, New Zealand, 2022, 66-76. https://doi.org/10.1109/VR51125.2022.00024
28. Lupenko, S. Theoretical Foundations of Modeling and Processing Cyclic Signals in Information Systems. 2nd ed. Stereotype. Lviv: Magnolia 2006 Publ., 2020; 340. (In Ukrainian) ISBN 978-617-574-108-5
29. Lupenko, S. The Mathematical Model of Cyclic Signals in Dynamic Systems as a Cyclically Correlated Random Process. Mathematics 2022, 10, 3406. https://doi.org/10.3390/math10183406
30. De Angelis, V.; Izzo, L.; Napolitano, A.; Tanda, M. Cyclostationarity-based parameter estimation of wide-band signals in mobile communications. In Proceedings of the IEEE/SP 13th Workshop on Statistical Signal Processing, Bordeaux, France, 17–20 July 2005
31. Estupiñan, E.; White, P.; SanMartin, C. A cyclostationary analysis applied to detection and diagnosis of faults in helicopter gearboxes. In Progress in Pattern Recognition, Image Analysis and Applications; Lecture Notes in Computer Science; Rueda, L., Mery, D., Kittler, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; 4756, 61–70.
32. Blachman, N.M. Beneficial effects of spectral correlation on synchronization. In Cyclostationarity in Communications and Signal Processing; Gardner, W.A., Ed.; IEEE Press: Piscataway, NJ, USA, 1994; 362–390.
33. Bloomfield, P.; Hurd, H.L.; Lund, R. Periodic correlation in stratospheric ozone time series. J. Time Ser. Anal. 1994, 15, 127–150.

34. Flagiello, F.; Izzo, L.; Napolitano, A. A computationally efficient and interference tolerant nonparametric algorithm for LTI system identification based on higher order cyclostationarity. IEEE Trans. Signal Process. 2000, 48, 1040–1052. https://doi.org/10.1109/78.827538
35. Dimc, F.; Baldini, G.; Kandeepan, S. Experimental detection of mobile satellite transmissions with cyclostationary features. Int. J. Satell. Commun. Netw. 2015, 33, 163–183. https://doi.org/10.1002/sat.1081
36. Dobre, O.; Oner, M.; Rajan, S.; Inkol, R. Cyclostationarity-based robust algorithms for QAM signal identification. IEEE Commun. Lett. 2012, 16, 12–15. https://doi.org/10.1109/LCOMM.2011.112311.112006
37. Bouillaut, L.; Sidahmed, M. Cyclostationary approach and bilinear approach: Comparison, applications to early diagnosis for helicopter gearbox and classification method based on HOCS. Mech. Syst. Signal Process. 2001, 15, 923–943.
38. Cheong, C.; Joseph, P. Cyclostationary spectral analysis for the measurement and prediction of wind turbine swishing noise. J. Sound Vib. 2014, 333, 3153–3176. https://doi.org/10.1016/j.jsv.2014.02.031
39. Capdessus, C.; Sidahmed, M.; Lacoume, J.L. Cyclostationary processes: Application in gear faults early diagnosis. Mech. Syst. Signal Process. 2000, 14, 371–385. https://doi.org/10.1006/mssp.1999.1260
40. Demorest, P.B. Cyclic spectral analysis of radio pulsars. Mon. Not. R. Astron. Soc. 2011, 416, 2821–2826. https://doi.org/10.1111/j.1365-2966.2011.19230.x
41. Chorna, L.B. Statistical estimators of a periodically correlated random process for a voiced speech signal. J. Acoust. Soc. Am. 2003, 113, 2271.
42. Thomas, L.J.; Clark, K.W.; Mead, C.N.; Ripley, K.L.; Spenner, B.F.; Oliver, G.C. Automated Cardiac Dysrhythmia Analysis. Proc. IEEE 1979, 67, 1322–1337. https://doi.org/10.1109/PROC.1979.11450
43. Hurd, H.L. Periodically correlated processes with discontinuous correlation function. Theory Probab. Its Appl. 1974, 19, 804–808. https://doi.org/10.1137/1119088

44. Hurd, H.L. Nonparametric time series analysis for periodically correlated processes. IEEE Trans. Inf. Theory 1989, 35, 350–359. https://doi.org/10.1109/18.32129
45. Marchenko, B.G. Linear Periodic Processes; Electrical Engineering; Pr. Institute of Electrodynamics, National Academy of Sciences of Ukraine: Kyiv, Ukraine, 1999; 165–182. (In Ukrainian)
46. Lupenko, S.; Pryimak, M.; Shcherbak, L. Simulation of Linear Periodic Random Processes. Sci. J. Ternopil State Tech. Univ. 2000, 5, 97–103. (In Ukrainian)
47. Lupenko, S.; Shcherbak, L. Constructive mathematical model of cardiac signals based on linear periodic random processes and fields. Sci. J. Ternopil State Tech. Univ. 2000, 5, 101–110. (In Ukrainian)
48. Lytvynenko, Y. Computer Modeling and Processing of the Cyclic Signals of the Heart. In Measuring and Computing Technics in Production Processes; Lytvynenko, Y., Lupenko, S., Shcherbak, L., Eds.; Khmelnytsky University: Khmelnytsky, Ukraine, 2000; 132–139. (In Ukrainian)
49. Lupenko, S. Mathematical modeling and methods of processing of cyclic signals of the heart in diagnostic systems of cardiometry. Sci. J. Ternopil State Tech. Univ. 2001, 6, 103–111. (In Ukrainian)
50. Lupenko, S. Modeling and Methods of Processing Cyclic Heart Signals on the Basis of Linear Random Functions: Dissertation: 01.05.02/Lupenko S.; Ternopil State Technical University named after Ivan Pulyuy: Ternopil, Ukraine, 2001; 256. (In Ukrainian)
51. Lupenko, S. The cyclic random process with variable rhythm. In Proceedings of the 9th Scientific Conference at Ternopil Ivan Pul'uj State Technical University, Ternopil, Ukraine, 16–18 September 2005; 61. (In Ukrainian).
52. Lupenko, S. Statistical Methods for Common Processing of the Joint Rhythmically Connected Cyclic Random Processes. In Measuring and Computing Technics in Production Processes; Khmelnytsky University: Khmelnytsky, Ukraine, 2005; 2, 80–84. (In Ukrainian)

53. Lupenko, S. Cyclic Functions and Their Classification in Problems of Modeling Cyclic Signals and Oscillatory Systems. In Measuring and Computing Technics in Production Processes; Khmelnytsky University: Khmelnytsky, Ukraine, 2005; 1, 177–185. (In Ukrainian)
54. Lupenko, S. Statistical Methods for Processing Cyclic Random Process. In Electronics and Managements Systems; National Aviation University: Kyiv, Ukraine, 2006; 59–65. (In Ukrainian)
55. Lupenko, S. Peculiarities of Discretization of Cyclic Functions. In Measuring and Computing Technics in Production Processes; Khmelnytsky University: Khmelnytsky, Ukraine, 2006; 1, 59–65. (In Ukrainian)
56. Lupenko, S. Cyclic and periodic random processes with a zone temporal structure and their probabilistic properties. Sci. J. Ternopil State Tech. Univ. 2006, 11, 150–155. (In Ukrainian)
57. Lupenko, S. The problem of interpolation of the rhythm function of a cyclic function with a known zone structure. In Electronics and Managements Systems; National Aviation University: Kyiv, Ukraine, 2007; 27–35. (In Ukrainian)
58. Lupenko, S. Scale transformation operator in the tasks of modeling and analysis of cyclic signals. Sci. J. Ternopil State Tech. Univ. 2007, 12, 141–152. (In Ukrainian)
59. Lupenko, S. Cyclic functional relation as the basis of the mathematical formalism of the theory of modeling and analysis of cyclic signals. Sci. J. Ternopil State Tech. Univ. 2007, 12, 183–195. (In Ukrainian)
60. Lupenko, S.; Demyanchuk, N.; Sverstiuk, A., Eds. Conceptual and Methodological Basis of Simulation of Cyclic Signals on a Computer Using Their Model as a Cycle Functional Relation. In Measuring and Computing Technics in Production Processes; Khmelnytsky University: Khmelnytsky, Ukraine, 2008; 2, 101–111. (In Ukrainian)
61. Lupenko, S.; Shcherbak, L. Some features and analytical dependences in the tasks of spectral analysis and synthesis of signals on the basis of their model in

the form of cyclic function. Sci. J. Ternopil State Tech. Univ. 2008, 13, 145–156. (In Ukrainian)

62. Lupenko, S. The Development of the Theory of Modeling and Processing of Cyclic Signals in Information Systems. Ph.D. Thesis, Lviv Polytechnic National University, Lviv, Ukraine, 2010; p. 479. (In Ukrainian)
63. Brillinger, D.R. Fourier analysis of stationary processes. Proc. IEEE 1974, 62, 1628–1643. https://doi.org/10.1109/PROC.1974.9682
64. Cramer, H. On the theory of stationary random processes. Ann. Math. 1940, 1, 215–230. https://doi.org/10.2307/1968827
65. Grenander, U.; Rosenblatt, M. Statistical Analysis of Stationary Time Series; Wiley: New York, NY, USA, 1957.
66. Benedetto, J.J. Harmonic Analysis and Applications; CRC Press: New York, NY, USA, 1996; 368.
67. Besicovitch, A.S. Almost Periodic Functions; Cambridge University Press: London, UK, 1932; 180.
68. Chen, G.; Beer, M.; Liu, Y. Modeling response spectrum compatible pulse-like ground motion. Mech. Syst. Signal Process. 2022, 177, 109177. https://doi.org/10.1016/j.ymssp.2022.109177
69. Darbas, M.; Lohrengel, S. Review on Mathematical Modelling of Electroencephalography (EEG). Jahresber. Dtsch. Math. Ver. 2019, 121, 3–39. https://doi.org/10.1365/s13291-018-0183-z
70. Grech, R.; Cassar, T.; Muscat, J.; Camilleri, K.P.; Fabri, S.G.; Zervakis, M.; Xanthopoulos, P.; Sakkalis, V.; Vanrumste, B. Review on solving the inverse problem in EEG source analysis. J. NeuroEng. Rehabil. 2008, 5, 25. https://doi.org/10.1186/1743-0003-5-25
71. Glomb, K.; Cabral, J.; Cattani, A.; Mazzoni, A.; Raj, A.; Franceschiello, B. Computational Models in Electroencephalography. Brain Topogr. 2022, 35, 142–161. https://doi.org/10.1007/s10548-021-00828-2

72. Abiri, R.; Borhani, S.; Sellers, E.W.; Jiang, Y.; Zhao, X. A comprehensive review of EEG-based brain–computer interface paradigms. J. Neural Eng. 2019, 16, 011001. https://doi.org/10.1088/1741-2552/aaf12e
73. Karlekar, M.; Gupta, A. Stochastic modeling of EEG rhythms with fractional Gaussian Noise. In Proceedings of the 2014 22nd European Signal Processing Conference (EUSIPCO), Lisbon, Portugal, 2–5 September 2014; 17, 2520–2524.
74. Al-Nashash, H.; Al-Assaf, Y.; Paul, J.; Thakor, N. EEG signal modeling using adaptive Markov process amplitude. IEEE Trans. Biomed. Eng. 2004, 51, 744–751. https://doi.org/10.1109/TBME.2004.826602
75. Mansouri, F.; Dunlop, K.; Giacobbe, P.; Downar, J.; Zariffa, J. A fast EEG forecasting algorithm for phase-locked transcranial electrical stimulation of the human brain. Front. Neurosci. 2017, 11, 401. https://doi.org/10.3389/fnins.2017.00401
76. Ghorbanian, P.; Ramakrishnan, S.; Simon, A.J.; Ashrafiuon, H. Stochastic Dynamic modeling of the human brain EEG signal. In Proceedings of the ASME 2013 Dynamic Systems and Control Conference, DSCC, Palo Alto, CA, USA, 21–23 October 2013.
77. Sanei, S.; Chambers, J.A. EEG Signal Processing; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2007.
78. Parra, L.C.; Spence, C.D.; Gerson, A.D.; Sajda, P. Recipes for the linear analysis of EEG. NeuroImage 2005, 28, 326–341. https://doi.org/10.1016/j.neuroimage.2005.05.032
79. Fryz, M.; Stadnyk, M. Justification of mathematical model of the steady-state visual evoked potential in a form of the linear random process. Electron. Control Systems. 2013, 1, 100–106. https://doi.org/10.18372/1990-5548.35.5797
80. Fryz, M. Conditional linear random process and random coefficient autoregressive model for EEG analysis. In Proceedings of the 2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON), Kyiv, Ukraine, 29 May–2 June 2017; 305–309.

81. Kramer, M.A.; Szeri, A.J.; Sleigh, J.W.; Kirsch, H.E. Mechanisms of seizure propagation in a cortical model. J. Comput. Neurosci. 2007, 22, 63–80. https://doi.org/10.1007/s10827-006-9508-5
82. Kramer, M.A.; Kirsch, H.E.; Szeri, A.J. Pathological pattern formation and cortical propagation of epileptic seizures. J. R. Soc. Interface 2005, 2, 113–127. https://doi.org/10.1098/rsif.2004.0028
83. Truong, N.C.D.; Wang, X.; Wanniarachchi, H.; Lang, Y.; Nerur, S.; Chen, K.; Liu, H. Mapping and understanding of correlated electroencephalogram (EEG) responses to the newsvendor problem. Sci. Rep. 2022, 12, 13800. https://doi.org/10.1038/s41598-022-17970-x
84. Ulrich, A.; Achim, V.B. Adaptive sequential segmentation of piecewise stationary time series. Inf. Sci. 1983, 29, 27–56.
85. Fingelkurts, Al.A.; Fingelkurts, An.A. Short-term EEG spectral pattern as a single event in EEG phenomenology. Open Neuroimag. J. 2010, 4, 130–156. https://doi.org/10.2174/1874440001004010130
86. Gupta, C.N.; Palaniappan, R. Enhanced detection of visual-evoked potentials in brain-computer interface using genetic algorithm and cyclostationary analysis. Comput. Intell. Neurosci. 2007, 2007, 28692. https://doi.org/10.1155/2007/28692
87. Pawuś, D.; Paszkiel, S. The Application of Integration of EEG Signals for Authorial Classification Algorithms in Implementation for a Mobile Robot Control Using Movement Imagery—Pilot Study. Appl. Sci. 2022, 12, 2161. https://doi.org/10.3390/app12042161
88. Gospodinova, E.; Lebamovski, P.; Georgieva-Tsaneva, G.; Negreva, M. Evaluation of the Methods for Nonlinear Analysis of Heart Rate Variability. Fractal Fract. 2023, 7, 388. https://doi.org/10.3390/fractalfract7050388
89. Toichi, M.; Sugiura, T.; Murai, T.; Sengoku, A. A New Method of Assessing Cardiac Autonomic Function and Its Comparison with Spectral Analysis and Coefficient of Variation of R–R Interval. J. Auton. Nerv. Syst. 1997, 62, 1–2, 79–84.

90. Jorna, P.G. Spectral Analysis of Heart Rate and Psychological State: A Review of Its Validity as a Workload Index. Biol. Psychol. 1992, 34, 237–257.
91. McKusick, V.A.; Talbot, S.A.; Webb, G.N. Spectral Phonocardiography: Problems and Prospects in Application of Bell Sound Spectrograph to Phonocardiography. Johns Hork. Hops. 1954, 187–198.
92. Merri, M.; Farden, D.; Mottley, J.; Titlebaum, E. Sampling Frequency of the Electrocardiogram for the Spectral Analysis of Heart Rate Variability. IEEE Trans. Biomed. Eng. 1990, 99–106.
93. Mallat, S. A Theory for Multiresolutional Signal Decomposition: The Wavelet Representation. IEEE Trans. Pattern Anal. Mach. Intell. 1989, 7, 674–693.
94. Osuhivs'ka, H.; Kyslak, I. Random Processes Statistic Application for Cardiosignals Characteristics Determination. In Proceedings of the VIth International Conference on Mathematical Methods in Electromagnetic Theory (MMET'96), 1996, 264–266.
95. Schapaugh, A.; Tyre, A.J. A simple method for dealing with large state spaces. Methods Ecol. Evol. 2012, 3, 949–957. https://doi.org/10.1111/j.2041-210X.2012.00242.x
96. Reichel, K.; Bahier, V.; Midoux, C.; Parisey, N.; Masson, J.-P.; Stoeckel, S. Interpretation and approximation tools for big, dense Markov chain transition matrices in population genetics. Algorithms Mol. Biol. 2015, 10, 31. https://doi.org/10.1186/s13015-015-0061-5
97. Papadopoulos, C.T.; Li, J.; O'Kelly, M.E.J. A classification and review of timed Markov models of manufacturing systems. Comput. Ind. Eng. 2019, 128, 219–244. https://doi.org/10.1016/j.cie.2018.12.019
98. Hadžić, N.; Ložar, V.; Abdulaj, F. A Finite State Method in the Performance Evaluation of the Bernoulli Serial Production Lines. Appl. Sci. 2020, 10, 6602. https://doi.org/10.3390/app10186602
99. Graichen, U.; Eichardt, R.; Fiedler, P.; Strohmeier, D.; Haueisen, J. Adaptive Spatial Harmonic Analysis of EEG Data using Laplacian Eigenspace. In Proceedings of the 2011 8th International Symposium on Noninvasive

Functional Source Imaging of the Brain and Heart and the 2011 8th International Conference on Bioelectromagnetism, Banff, AB, Canada, 13–16 May 2011; pp. 18–21.

100. Kumar, N.; Alam, K.; Siddiqi, A.H. Wavelet Transform for Classification of EEG Signal using SVM and ANN. Bi-Omed. Pharmacol. J. 2017, 10, 2061–2069. https://doi.org/10.13005/bpj/1328

101. Bajaj, N. Wavelets for EEG Analysis. In Wavelet Theory; IntechOpen: London, UK, 2020.

102. Al-Nashash, H.A.; Thakor, N.V. Monitoring of global cerebral ischemia using wavelet entropy rate of change. IEEE Trans. Biomed. Eng. 2005, 52, 2119–2122. https://doi.org/10.1109/TBME.2005.857634

103. Avramidis, K.; Zlatintsi, A.; Garoufis, C.; Maragos, P. Multiscale Fractal Analysis on EEG Signals for Music-Induced Emotion Recognition. In Proceedings of the 2021 29th European Signal Processing Conference (EUSIPCO), Dublin, Ireland, 23–27 August 2021; 1316–1320.

104. Łysiak, A.; Paszkiel, S. A Method to Obtain Parameters of One-Column Jansen–Rit Model Using Genetic Algorithm and Spectral Characteristics. Appl. Sci. 2021, 11, 677. https://doi.org/10.3390/app11020677

105. Lupenko, S.; Butsiy, R. Isomorphic Multidimensional Structures of the Cyclic Random Process in Problems of Modeling Cyclic Signals with Regular and Irregular Rhythms. Fractal Fract. 2024, 8, 203. https://doi.org/10.3390/fractalfract8040203

106. Lupenko, S.; Lutsyk, N.; Lapusta, Y. Cyclic linear random process as a mathematical model of cyclic signals. Acta Mech. Autom. 2015, 9, 219–224. https://doi.org/10.1515/ama-2015-0035

107. Lupenko, S.; Lytvynenko, I.; Sverstiuk, A.; Horkunenko, A.; Shelestovskyi, B. Software for statistical processing and modeling of a set of synchronously registered cardio signals of different physical nature. CEUR Workshop Proc. 2021, 2864, 194–205. https://doi.org/10.32782/cmis/2864-17

108. Lytvynenko, I.; Lupenko, S.; Marushchak, P. Analysis of multiple cracking of nanocoating as a cyclic random process. Optoelectron. Instrum. Data Process. 2013, 49, 164–170. https://doi.org/10.3103/S8756699013020088

109. Lytvynenko, I.; Lupenko, S.; Nazarevych, O.; Shymchuk, G.; Hotovych, V. Mathematical Model of Gas Consumption Process in the Form of Cyclic Random Process. In Proceedings of the 16th IEEE International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 2021; 232–235.

110. Lupenko, S.; Stadnyk, N.; Nnamene, C. An Approach to Constructing a Taxonomic Tree of Models Cyclic Signals in the Tasks of Developing an Onto-Oriented System for Decisions Supporting of Models Choice. In Proceedings of the 2019 9th International Conference on Advanced Computer Information Technologies (ACIT), Ceske Budejovice, Czech Republic, 5–7 June 2019; pp. 89–92.

111. Nnamene, C.; Lupenko, S.; Volyanyk, O.; Orobchuk, O. Computer Ontology of Mathematical Models of Cyclic Space-Time Structure Signals. In Proceedings of the Intelligent Information Technologies & Systems of Information Security 2022 (IntelITSIS 2022), Khmelnytskyi, Ukraine, 23–25 March 2022; Khmelnytskyi National University, Computer Engineering & Information Systems Department. 103–118.

112. Lupenko, S.; Pasichnyk, V.; Kunanets, N. Axiomatic-Deductive Strategy of the Organization of the Content of Academic Discipline in the Field of Information Technologies Using the Ontological Approach. In Proceedings of the 13th IEEE International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, September 11-14, 2018; 1, 387–390.

113. Lupenko, S.; Pasichnyk, V.; Kunanets, N. Organization of the Content of Academic Discipline in the Field of Information Technologies Using Ontological Approach. In Proceedings of The International Conference on Computer Science and Information Technologies, Advances in Intelligent

Systems and Computing III, CSIT 2018, September 11-14, Lviv, Ukraine, 312–327.

114. Lupenko, S.; Pasichnyk V.; Kunanets N. Axiomatic-Deductive Strategy for IT Discipline Content Formation. Inf. Technol. Learn. Tools 2019, 73, 149–160.

115. Lupenko, S.; Butsiy, R.; Volyanyk, O.; Stadnyk, N. Advanced Signal Processing and Classification of EEG Patterns in Neurointerface Systems. In Proceedings of the 3rd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2023), Ternopil, Ukraine, Opole, Poland, 22–24 November 2023, 3628, 156–164.

116. Butsiy, R.; Lupenko, S.; Zozulya, A. Comprehensive justification for the choice of software development tools and hardware components of a multi-channel neurointerface system. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), 2021, 1, 309-312. https://doi.org/10.1109/CSIT52700.2021.9648788

117. Delorme, A.; Makeig, S. EEGLAB: An Open Source Toolbox for Analysis of Single-Trial EEG Dynamics Including Independent Component Analysis. J. Neurosci. Methods 2004, 134, 9–21. https://doi.org/10.1016/j.jneumeth.2003.10.009

118. Brunet, D.; Murray, M.M.; Michel, C.M. Spatiotemporal Analysis of Multichannel EEG: CARTOOL. Comput. Intell. Neurosci. 2011, 2011. http://dx.doi.org/10.1155/2011/813870

119. Oostenveld, R.; Fries, P.; Maris, E.; Schoffelen, J.M. FieldTrip: Open Source Software for Advanced Analysis of MEG, EEG, and Invasive Electrophysiological Data. Comput. Intell. Neurosci. 2011, 2011. https://doi.org/10.1155/2011/156869

120. Tadel, F.; Baillet, S.; Mosher, J.C.; Pantazis, D.; Leahy, R.M. Brainstorm: A User-Friendly Application for MEG/EEG Analysis. Comput. Intell. Neurosci. 2011, 2011. https://doi.org/10.1155/2011/879716

121. Kruschwitz, J.D.; List, D.; Waller, L.; Rubinov, M.; Walter, H. GraphVar: A User-Friendly Toolbox for Comprehensive Graph Analyses of Functional Brain

Connectivity. J. Neurosci. Methods 2015, 245, 107–115. http://dx.doi.org/10.1016/j.jneumeth.2015.02.021

122. Hassan, M.; Shamas, M.; Khalil, M.; Falou, W.E.; Wendling, F. EEGNET: An Open Source Tool for Analyzing and Visualizing M/EEG Connectome. PLoS ONE 2015, 10, e0138297. https://doi.org/10.1371/journal.pone.0138297

123. Homer, M.; Irvine, J.M.; Wendelken, S. A model-based approach to human identification using ECG. In Proceedings of the SPIE Conference on Optics and Photonics for Global Homeland Security V and Biometric Technology for Human Identification VI, Orlando, FL, USA, 13–17 April 2009; 7306, 730625. http://dx.doi.org/10.1117/12.819327

124. Benouis, M.; Mostefai, L.; Costen, N.; Regouid, M. ECG-based biometric identification using one-dimensional local difference pattern. Biomed. Signal Process. Control. 2021, 64, 102226. https://doi.org/10.1016/j.bspc.2020.102226

125. Irvine, J.M.; Wiederhold, B.K.; Gavshon, L.W.; Israel, S.; McGehee, S.B.; Meyer, R.; Wiederhold, M.D. Heart rate variability: A new biometric for human identification. In Proceedings of the International Conference on Artificial Intelligence (IC-AI'01), Las Vegas, NV, USA, 7–9 November 2001; 1106–1111.

126. Wan, Y.; Yao, J. A neural network to identify human subjects with electrocardiogram signals. In Proceedings of the World Congress on Engineering and Computer Science, San Francisco, CA, USA, 22–24 October 2008; 1–4.

127. Lupenko, S.; Butsiy, R. Express Method of Biometric Person Authentication Based on One Cycle of the ECG Signal. Sci. J. TNTU (Tern.), 2024, 113, 100–110. https://doi.org/10.33108/visnyk_tntu2024.01.100

128. Pereira, T.M.C.; Conceição, R.C.; Sebastião, R. Initial Study Using Electrocardiogram for Authentication and Identification. Sensors 2022, 22, 2202 https://doi.org/10.3390/s22062202

129. García-González, M.A.; Argelagós-Palau, A. Data from: combined measurement of ECG, breathing, and Seismocardiograms. PhysioNet. 2013. https://doi.org/10.13026/C2KW23

130. García-González, M.A.; Argelagós-Palau, A.; Fernández-Chimeno, M.; Ramos-Castro, J. A comparison of heartbeat detectors for the seismocardiogram. In Computing in Cardiology 2013; September 22–25, 2013; 461–464.

131. Goldberger, A.; Amaral, L.; Glass, L.; Hausdorff, J.; Ivanov, P.C.; Mark, R.; et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation 2000, 101, e215–e220. https://doi.org/10.1161/01.cir.101.23.e215

ДОДАТКИ

Додаток А

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА ЗА ТЕМОЮ ДИСЕРТАЦІЇ ТА ВІДОМОСТІ ПРО АПРОБАЦІЮ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОЇ РОБОТИ

Праці, в яких опубліковано основні наукові результати дисертації:

1. Lupenko, S.; Butsiy, R. Isomorphic Multidimensional Structures of the Cyclic Random Process in Problems of Modeling Cyclic Signals with Regular and Irregular Rhythms. Fractal Fract. 2024, 8, 203. [Article, Scopus, Web of Science, Google Scholar, JCR – Q1, CiteScore - Q1, SJR – Q2, Impact Factor 5.4] doi:https://doi.org/10.3390/fractalfract8040203

2. Lupenko, S.; Butsiy, R.; Shakhovska, N. Advanced Modeling and Signal Processing Methods in Brain–Computer Interfaces Based on a Vector of Cyclic Rhythmically Connected Random Processes. Sensors 2023, 23, 760. [Article, Scopus, Web of Science, Google Scholar, JCR - Q2, CiteScore - Q1, SJR – Q2, Impact Factor 3.9] doi:https://doi.org/10.3390/s23020760

3. Lupenko, S.; Butsiy, R.; Volyanyk, O.; Stadnyk, N. Advanced Signal Processing and Classification of EEG Patterns in Neurointerface Systems. In Proceedings of the 3rd International Workshop on Information Technologies: Theoretical and Applied Problems (ITTAP 2023), Ternopil, Ukraine, Opole, Poland, 22–24 November 2023, 3628, 156–164. [Article, Scopus, Web of Science, Google Scholar, ISSN 1613-0073]

4. Butsiy, R.; Lupenko, S. Comparison of Modern Methods of Classification of EEG Patterns for Neurointerface Systems. In: Yang, XS., Sherratt, S., Dey, N., Joshi, A. (eds) Proceedings of Seventh International Congress on Information and Communication Technology. Lecture Notes in Networks and Systems; Springer, Singapore, 2022, 465, 345-354. [Article, Scopus, Google Scholar, ISSN 2367-3370] doi:https://doi.org/10.1007/978-981-19-2397-5_32

5. Butsiy, R.; Lupenko, S.; Zozulya, A. Comprehensive justification for the choice of software development tools and hardware components of a multi-channel neurointerface system. 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), 2021, 1, 309-312. [Article, Scopus, Google Scholar, ISSN 2766-3639]
doi:https://doi.org/10.1109/CSIT52700.2021.9648788

6. Butsiy, R.; Lupenko, S. Comparative Analysis of Neurointerface Technologies for the Problem of Their Reasonable Choice in Human-Machine Information Systems. Sci. J. TNTU (Tern.), 2020, 4, 135–148. [Article, Index Copernicus, Google Scholar, ISSN 2522-4433]
doi:https://doi.org/10.33108/visnyk_tntu2020.04.135

7. Lupenko, S.; Butsiy, R. Express Method of Biometric Person Authentication Based on One Cycle of the ECG Signal. Sci. J. TNTU (Tern.), 2024, 113, 100–110. [Article, Index Copernicus, ISSN 2522-4433]
doi:https://doi.org/10.33108/visnyk_tntu2024.01.100

Праці, які засвідчують апробацію матеріалів дисертації:

8. Лупенко, С.; Буцій, Р. Сучасні нейроінтерфейсні технології: актуальність, перспективи та складності. У Матеріалах міжнародної наукової конференції “Іван Пулюй: життя в ім’я науки та України” (до 175-ліття від дня народження), 28-30 вересня 2020 р.; ТНТУ: Тернопіль, Україна, 2020, 81–82. [Conference Paper, Google Scholar]

9. Буцій, Р.; Лупенко, С. Аналіз основних характеристик комерційних нейроінтерфейсів. У Матеріалах IX Міжнародної науково-технічної конференції молодих учених та студентів, 25-26 листопада 2020 р.; ТНТУ: Тернопіль, Україна, 2020, 2, 9–10. [Conference Paper, Google Scholar]

10. Катеринюк, І.; Лупенко, С.; Буцій, Р. Аудіоінтерфейсні та нейроінтерфейсні технології вводу діагностичної інформації в інформаційну систему “Імідж-Терапевт” для народної медицини. У Матеріалах IX Міжнародної науково-технічної конференції молодих учених та студентів, 25-26 листопада

2020 р.; ТНТУ: Тернопіль, Україна, 2020, 2, 24–25. [Conference Paper, Google Scholar]

11. Буцій, Р.; Лупенко, С. Аналіз методів для задач опрацювання сигналів нейроінтерфейсних систем. У Матеріалах VIII науково-технічної конференції “Інформаційні моделі, системи та технології”, 9-10 грудня 2020 р.; ТНТУ: Тернопіль, Україна, 2020, 3. [Conference Paper, Google Scholar]

12. Буцій, Р.; Лупенко, С. Підхід до побудови доступних за ціною дослідницьких нейроінтерфейсних систем. У Матеріалах XX Міжнар. наук.-практ. конф. “Сучасні інформаційні технології управління екологічною безпекою, природокористуванням, заходами в надзвичайних ситуаціях”, 4-8 жовтня 2021 р.; Юстон: Київ, Україна, 2021, 131–134. [Conference Paper]

13. Буцій, Р.; Лупенко, С. Принцип керування роботизованою рукою зі зворотним зв’язком за допомогою нейроінтерфейсу. У Матеріалах IX Науково-технічної конференції “Інформаційні моделі, системи та технології”, 8-9 грудня 2021 р.; Тернопіль, Україна, 2021, 3. [Conference Paper]

14. Лупенко, С; Буцій, Р. Математична модель векторної ЕЕГ у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів. У Матеріалах XXI Міжнар. наук.-практ. конф. "Інформаційно-комунікаційні технології та сталий розвиток", 14–16 листопада 2022 р.; Юстон: Київ, Україна, 2022, 49–52.
[Conference Paper, Google Scholar]

15. Лупенко, С; Буцій, Р. Дослідження класифікаторів для інтерфейсів мозок-комп'ютер на основі сигналів ЕЕГ. У Матеріалах XXII Міжнар. наук.-практ. конф. “Інформаційно-комунікаційні технології та сталий розвиток”, 14–15 листопада 2023 р.; Юстон: Київ, Україна, 2023, 57–59. [Conference Paper]

Додаток Б

АКТИ ВПРОВАДЖЕННЯ

ЗАТВЕРДЖУЮ
Проректор закладу вищої освіти з
наукової роботи Тернопільського
національного медичного університету
ім. І.Я. Горбачевського

_____ проф. І. Кліщ
“ 14 ” _____ 2024 р.

АКТ ВПРОВАДЖЕННЯ

результатів дисертаційної роботи Буція Романа Андрійовича

Даний акт складений про те, що в науково-дослідній роботі «Інформаційні технології Data Science та Big Data в кіберфізичних системах медико-біологічних процесів» (№ держреєстрації 0122U000030), яка проводиться на кафедрі медичної інформатики в Тернопільському національному медичному університеті ім. І.Я. Горбачевського, використано наступні результати дисертаційного дослідження Р.А. Буція “Моделювання та методи ефективного опрацювання циклічних сигналів в нейроінтерфейсних та кардіодіагностичних системах”:

- Нову математичну модель векторної ЕЕГ у вигляді вектора циклічних ритмічно пов’язаних випадкових процесів, яка дозволяє ефективно аналізувати циклічні біомедичні сигнали з урахуванням їх стохастичності та циклічності.
- Методи ритмоадаптивного статистичного аналізу ЕЕГ сигналів, що дозволяють з високою точністю аналізувати статистичні характеристики ЕЕГ сигналів для розробки ефективних нейроінтерфейсних систем.
- Ритмоадаптивний метод біометричної аутентифікації особи за одним циклом ЕКГ, який демонструє високу точність і надійність в ідентифікації особи.
- Систему комп’ютерних програм для автоматизованого аналізу ЕЕГ та ЕКГ сигналів, яка включає інструменти для ритмоадаптивного статистичного аналізу та біометричної аутентифікації, які забезпечують інструменти для ефективного аналізу циклічних біомедичних сигналів.

Прийнято до впровадження та подальшого сумісного використання і практичної реалізації зазначених позицій є підставою для підвищення рівня інформативності автоматизованого аналізу циклічних біомедичних сигналів в сучасних комп’ютеризованих системах для медицини.

Завідувач закладу вищої освіти
кафедри медичної інформатики
Тернопільського національного медичного університету
ім. І.Я. Горбачевського, д.б.н., професор Дмитро Вакуленко

ЗАТВЕРДЖУЮ
Проректор з наукової роботи
Тернопільського національного
технічного університету
ім. І. Пулюя
_____ Г.О. Марущак
"___" _____ 2024 р.

АКТ

про впровадження результатів дисертаційної роботи "Моделювання та методи ефективного опрацювання циклічних сигналів в нейроінтерфейсних та кардіодіагностичних системах" аспіранта Буція Романа Андрійовича, представленої на здобуття наукового ступеня доктора філософії за спеціальністю 113 "Прикладна математика", в навчальному процесі ТНТУ ім. І. Пулюя.

Цим актом підтверджується, що результати дисертаційної роботи Буція Романа Андрійовича використано при проведенні лекційних та лабораторних занять із дисципліни "Математичне забезпечення комп'ютерних систем та мереж" для студентів спеціальності 123 "Комп'ютерна інженерія" другого (магістерського) рівня вищої освіти.

На основі розробленої математичної моделі векторної ЕЕГ у вигляді вектора циклічних ритмічно пов'язаних випадкових процесів та методів їх ритмоадаптивного статистичного аналізу, а також ритмоадаптивного методу біометричної аутентифікації особи за одним циклом ЕКГ, було впроваджено інноваційний підхід до аналізу та опрацювання циклічних біомедичних сигналів.

Використання вказаних результатів у навчальному процесі сприяє глибшому розумінню студентами та фахівцями можливостей застосування інформаційних технологій для передових методів в аналізі та моделюванні біомедичних сигналів, відкриваючи їх нові перспективи, зокрема у розробці нейроінтерфейсних систем та методів біометричної ідентифікації.

Завідувач кафедри
комп'ютерних систем та мереж,
к.т.н., доцент Г.М. Осухівська

Додаток В

ЛІСТИНГ ПРОГРАМИ

Файл main.py (Точка входу в програму)

```
from diff.diff_confusion_matrics import DiffConfusionMatrix
from get_config.eeg_config import EEGConfig
from loguru import logger
import argparse
from my_helpers.classifiers import Classifiers
from my_helpers.classifiers_int import ClassifiersInt
from my_helpers.data_preparation import DataPreparation
from my_helpers.mathematical_statistics import MathematicalStatistics
from my_helpers.plot_classifiers import PlotClassifiers
from my_helpers.plot_statistics import PlotStatistics

from my_helpers.rhythm_function import RhythmFunction
from no_classifires.no_classifires import NoClassidire
from no_classifires.no_classifires_all_chanels import NoClassidireAllChanels
from no_classifires.no_classifires_fourier import NoClassidireFourier
from no_classifires.no_classifires_fourier_all_chanels import NoClassidireFourierAllChanels
from statistics_distance.statistics_distance import StatisticsDistance
from statistics_distance.statistics_distance_fourier import StatisticsDistanceFourier

if __name__ == '__main__':
    logger.info('START of execution')
    parser = argparse.ArgumentParser()
    parser.add_argument('action',    choices=('avg-diff-confusion-matrix',    'diff-get-max',
'diff-all', 'train-classifier-int', 'plot-diff-fourier-time', 'diff-fourier-time', 'diff-variance',
'diff-confusion-matrix', 'statistics-distance-f', 'statistics-distance', 'no-all-test-fd', 'no-all-
sigma-fd', 'no-all-test-d', 'no-all-sigma-d', 'no-all-test-f', 'no-all-sigma-f', 'no-all-mean-f',
'no-all-test', 'no-all-sigma', 'plot-fr', 'plot-eeg', 'plot-statistics', 'plot-fourier-statistics',
'train-classifier', 'plot-classifier'))
    parser.add_argument('-c', type=str, required=True)
    parser.add_argument('-d', type=str)
    parser.add_argument('-s', type=str)
    a = parser.parse_args()
    config_block = a.c
    logger.debug("Read config file")
    eeg_config = EEGConfig(config_block)
    logger.debug(eeg_config)
    if a.action == "plot-fr":
        RhythmFunction(eeg_config).plotFr()
    if a.action == "plot-eeg":
        RhythmFunction(eeg_config).plotEEG()
    if a.action == "plot-statistics":
        data = DataPreparation(eeg_config)
        statistics_passivity, statistics_activity = [], []
        for passivity, activity in zip(*data.getPreparedData()):
            statistics_passivity.append(MathematicalStatistics(passivity))
            statistics_activity.append(MathematicalStatistics(activity))
```

```
                PlotStatistics(statistics_passivity,      data.getModSamplingRate(),      eeg_config,
"passivity").plotAllStatistics()
                PlotStatistics(statistics_activity,      data.getModSamplingRate(),      eeg_config,
"activity").plotAllStatistics()

            if a.action == "plot-fourier-statistics":
                data = DataPreparation(eeg_config)
                statistics_passivity, statistics_activity = [], []
                for passivity, activity in zip(*data.getPreparedData()):
                    statistics_passivity.append(MathematicalStatistics(passivity))
                    statistics_activity.append(MathematicalStatistics(activity))
                PlotStatistics(statistics_passivity,      data.getModSamplingRate(),      eeg_config,
"passivity").plotAllFourierStatistics()
                PlotStatistics(statistics_activity,      data.getModSamplingRate(),      eeg_config,
"activity").plotAllFourierStatistics()

            if a.action == "train-classifier":
                data = DataPreparation(eeg_config)
                for ccc in [1, 2, 3]:
                    for power in [1, 2, 3, 4]:
                        for fourier_type in ["an", "bn", "an_bn"]:
                            Classifiers(eeg_config, data, fourier_type, power, ccc)

            if a.action == "train-classifier-int":
                data = DataPreparation(eeg_config)
                for power in [1, 2, 3, 4]:
                    for fourier_type in ["an", "bn", "an_bn"]:
                        ClassifiersInt(eeg_config, data, fourier_type, power)
            if a.action == "plot-classifier":
                for fourier_type in ["an_bn"]:
                    PlotClassifiers(eeg_config, None, fourier_type, 1)
            if a.action == "no-all-test":
                data = DataPreparation(eeg_config)
                NoClassidire(eeg_config, data).NoTest()
            if a.action == "no-all-sigma":
                data = DataPreparation(eeg_config)
                NoClassidire(eeg_config, data).NoAllSigma()
            if a.action == "no-all-test-f":
                data = DataPreparation(eeg_config)
                for fourier_type in ["an", "bn", "an_bn"]:
                    NoClassidireFourier(eeg_config, data, fourier_type).NoTest()
            if a.action == "no-all-sigma-f":
                data = DataPreparation(eeg_config)
                NoClassidireFourier(eeg_config, data).NoAllSigma()
            if a.action == "no-all-test-d":
                data = DataPreparation(eeg_config)
                for power in [1, 2, 3, 4]:
                    NoClassidireAllChanels(eeg_config, data, power).NoTest()

            if a.action == "no-all-sigma-d":
                data = DataPreparation(eeg_config)
                NoClassidireAllChanels(eeg_config, data).NoAllSigma()
```

```
            if a.action == "no-all-test-fd":
                data = DataPreparation(eeg_config)
                for power in [1]:
                    for fourier_type in ["an_bn"]:
                        for terms in [3, 4, 5, 7, 10, 15, 20, 25, 30, 35, 45, 50, 55, 60, 65, 70, 75,
80, 85, 90, 95, 100]:
                            NoClassidireFourierAllChanels(eeg_config, data, fourier_type, terms,
power).NoTest()
            if a.action == "no-all-sigma-fd":
                data = DataPreparation(eeg_config)
                NoClassidireFourierAllChanels(eeg_config, data).NoAllSigma()

            if a.action == "statistics-distance":
                data = DataPreparation(eeg_config)
                StatisticsDistance(eeg_config, data).Calculate()
            if a.action == "statistics-distance-f":
                data = DataPreparation(eeg_config)
                for power in [1]:
                    for fourier_type in ["an_bn"]:
                        for terms in [3, 4, 5, 7, 10, 15, 20, 25, 30, 35, 45, 50, 55, 60, 65, 70, 75,
80, 85, 90, 95, 100]:
                            StatisticsDistanceFourier(eeg_config, data, fourier_type, terms,
power).Calculate()
            if a.action == "diff-confusion-matrix":
                for channel_type in ["1"]:
                    for power in [1]:
                        for fourier_type in ["an_bn"]:
                            for terms in [3, 4, 5, 7, 10, 15, 20, 25, 30, 35, 45, 50, 55, 60, 65, 70,
75, 80, 85, 90, 95, 100]:
                                DiffConfusionMatrix(eeg_config, None, fourier_type, terms, power,
channel_type).DiffConfusionMatrix()
            if a.action == "avg-diff-confusion-matrix":
                for channel_type in ["1", "2", "3", "int"]:
                    for power in [1, 2, 3, 4]:
                        for fourier_type in ["an_bn"]:
                            for terms in [3, 4, 5, 7, 10, 15, 20, 25, 30, 35, 45, 50, 55, 60, 65, 70,
75, 80, 85, 90, 95, 100]:
                                DiffConfusionMatrix(eeg_config, None, fourier_type, terms, power,
channel_type).AVGDiffConfusionMatrix()

            if a.action == "diff-variance":
                for channel_type in ["1", "2", "3", "int"]:
                    for power in [1, 2, 3, 4]:
                        for fourier_type in ["an", "bn", "an_bn"]:
                            for terms in [40, 999]:
                                DiffConfusionMatrix(eeg_config, None, fourier_type, terms, power,
channel_type).DiffVariance()

            if a.action == "diff-all":
                DiffConfusionMatrix(eeg_config, None, None, None, None, None).DiffAll()
            if a.action == "plot-diff-fourier-time":
                DiffConfusionMatrix(eeg_config, None, None, None, None, None).PlotDiffFourierTime()
            if a.action == "diff-get-max":
```

```
DiffConfusionMatrix(eeg_config, None, None, None, None, None).DiffGetMax()
```

Файл eeg.config (Фрагмент конфігураційного файлу)

```
[DEFAULT]
data_path = DATA_EEG
img_path = img

[OPERATOR-11]
file_name = Operator 11/OBCI_B0.TXT
multiplier = 1
data_type = openbci
sigma = 1.8, 0.9

[WITH_TRAINING]
file_name = NULL/NULL.TXT
multiplier = 1
data_type = openbci
sigma = 0, 0

[WITHOUT_TRAINING]
file_name = NULL/NULL.TXT
multiplier = 1
data_type = openbci
sigma = 0, 0

[OPERATOR-1]
file_name = 30-09-2022/OBCI_39.TXT
multiplier = 1
data_type = openbci
sigma = 1.9, 0.5

[OPERATOR-2]
file_name = 06-11-2022/OBCI_3E.TXT
multiplier = 1
data_type = openbci
sigma = 1.8, 0.9

[OPERATOR-3]
file_name = Operator 3/OBCI_5A.TXT
multiplier = 1
data_type = openbci
sigma = 2.2, 2.0

[OPERATOR-4]
file_name = Operator 4/OBCI_5E.TXT
multiplier = 1
data_type = openbci
sigma = 2.0, 1.0

[OPERATOR-5]
file_name = Operator 5/OBCI_62.TXT
```

```
multiplier = 1
data_type = openbci
sigma = 1.8, 2.4

[OPERATOR-6]
file_name = Operator 6/OBCI_7C.TXT
multiplier = 1
data_type = openbci
sigma = 1.7, 2.3

[OPERATOR-7]
file_name = Operator 7/OBCI_87.TXT
multiplier = 1
data_type = openbci
sigma = 1.7, 2.3

[OPERATOR-8]
file_name = Operator 8/OBCI_95.TXT
multiplier = 1
data_type = openbci
sigma = 2.3, 0.9

[OPERATOR-9]
file_name = Operator 9/OBCI_A0.TXT
multiplier = 1
data_type = openbci
sigma = 2.3, 1.0

[OPERATOR-10]
file_name = Operator 10/OBCI_AB.TXT
multiplier = 1
data_type = openbci
sigma = 1.9, 0.9


[H_P001_B001_1_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/b001_1
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_B001_2_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/b001_2
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_1_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_1
multiplier = 0.072
```

```
pathology = 1
data_type = mat

[H_P001_M001_2_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_2
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_3_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_3
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_4_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_4
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_5_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_5
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_6_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_6
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_7_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_7
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_8_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_8
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_9_S2]
```

```
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_9
multiplier = 0.072
pathology = 1
data_type = mat

[H_P001_M001_10_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_001/m001_10
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_B002_1_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/b002_1
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_B002_2_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/b002_2
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_1_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_1
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_2_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_2
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_3_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_3
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_4_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_4
multiplier = 0.072
pathology = 1
```

```
data_type = mat

[H_P002_M002_5_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_5
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_6_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_6
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_7_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_7
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_8_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_8
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_9_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_9
multiplier = 0.072
pathology = 1
data_type = mat

[H_P002_M002_10_S2]
sig_name = 1
file_name = /home/MyFiles2/ECG/cebsdb_2minutes/mat/Person_002/m002_10
multiplier = 0.072
pathology = 1
data_type = mat
```

Файл eeg_config_validate.py (Валідація конфігурації)

```
from configparser import ConfigParser
from pathlib import Path

class EEGConfigException(Exception):
    pass
```

```
class EEGConfigConfig(ConfigParser):
    def __init__(self, config_file, config_block):
        super(EEGConfigConfig, self).__init__()

        if not Path(config_file).is_file():
            raise EEGConfigException(
                    'The config file %s does not exist' % config_file)

        self.read(config_file)
        self.validate_config(config_block)

    def validate_config(self, config_block):
        required_values = {
            'DEFAULT': {
                'data_path' : None,
                'img_path' : None
            },
            '%s' % (config_block): {
                'file_name': None,
                'multiplier': None,
                'data_type': ('openbci'),
                'sigma': None
            }
        }

        for section, keys in required_values.items():
            if section not in self:
                raise EEGConfigException(
                    'Missing section "%s" in the config file' % section)

            for key, values in keys.items():
                if key not in self[section] or self[section][key] == '':
                    raise EEGConfigException((
                        'Missing value for "%s" under section "%s" in ' +
                        'the config file') % (key, section))

                if values:
                    if self[section][key] not in values:
                        raise EEGConfigException((
                            'Invalid value for "%s" under section "%s" in ' +
                            'the config file') % (key, section))
```

Файл eeg_config.py (Інтерпритація конфігурації)

```
from loguru import logger
import sys

from get_config.eeg_config_validate import EEGConfigConfig, EEGConfigException

class EEGConfig:

    CONFIG_FILE_PATH = "src/config/ecg.config"
```

```
        def __init__(self, config_block, config_file_path=None):
            if config_file_path is not None:
                self.CONFIG_FILE_PATH = config_file_path
            logger.info("Read config file: {}", self.CONFIG_FILE_PATH)
            logger.debug("Config: {}", config_block)

            config = {}
            try:
                config = EEGConfigConfig(self.CONFIG_FILE_PATH, config_block)
            except EEGConfigException as e:
                logger.error("Invalid config file: {}", e)
                sys.exit(1)

            self.file_name = config[config_block]["file_name"].strip()
            self.multiplier = float(config[config_block]["multiplier"].strip())
            self.data_type = config[config_block]["data_type"].strip()
            self.sigma = [float(x) for x in config[config_block]["sigma"].strip().split(',')]
            self.config_block = config_block
            self.data_path = config["DEFAULT"]["data_path"].strip()
            self.img_path = config["DEFAULT"]["img_path"].strip()

        def getDataType(self):
            return self.data_type

        def getFileName(self):
            return self.file_name

        def getMultiplier(self):
            return self.multiplier

        def getConfigBlock(self):
            return self.config_block

        def getDataPath(self):
            return self.data_path

        def getImgPath(self):
            return self.img_path

        def getSigma(self):
            return self.sigma

        def __str__(self):
            logger.debug("toString {}", self.config_block)
            return ((
                    "\nConfig block: {0}\n" +
                    "File mame: {1}\n" +
                    "Multiplier: {2}\n" +
                    "Default data path {3}\n" +
                    "Default images path {4}\n"
                    ).format(self.config_block, self.file_name, self.multiplier, self.data_path,
self.img_path))
```

Файл read_data_file.py (Читаємо дані з файлу)

```
from loguru import logger
import pandas as pd
from pathlib import Path
from my_helpers.read_data.read_openbci_file import ReadOpenBCIFile

class ReadDataFile:

    def __init__(self, eeg_config):
        self.eeg_config = eeg_config
        data_type = eeg_config.getDataType()
        if data_type == 'openbci':
            self.instance = ReadOpenBCIFile(eeg_config)
        else:
            raise ValueError(f"Invalid data type: {data_type}")

    def __getattr__(self, name):
        return self.instance.__getattribute__(name)

    def getData(self):
        fr_path                                                                                    =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/function_rhythm.csv'
        if not Path(fr_path).is_file():
            e = 'The rhythm function file %s does not exist' % fr_path
            logger.error(e)
            raise FileNotFoundError(e)

        eeg_fr = pd.read_csv(fr_path)
        self.D_c = eeg_fr["D_c"]
        self.D_z = eeg_fr["D_z"]

        self.matrix_passivity = [[] for _ in range(len(self.signals))]
        self.matrix_activity = [[] for _ in range(len(self.signals))]

        D_c_scaled = [(dc - 1) * self.sampling_rate for dc in self.D_c]
        D_z_scaled = [(dz - 1) * self.sampling_rate for dz in self.D_z]

        for channel_number, signal in enumerate(self.signals):
            for i in range(len(self.D_z) - 1):
                passivity_start = int(D_c_scaled[i])
                passivity_end = int(D_z_scaled[i])

self.matrix_passivity[channel_number].append(signal[passivity_start:passivity_end])

                activity_start = int(D_z_scaled[i])
                activity_end = int(D_c_scaled[i + 1])

self.matrix_activity[channel_number].append(signal[activity_start:activity_end])
```

Файл read_openbci_file.py (Опис формату OpenBCI)

```
from loguru import logger
import csv
from scipy import signal
import numpy as np

class ReadOpenBCIFile:

    def __init__(self, ecg_config):
        data_path = f'{ecg_config.getDataPath()}/{ecg_config.getFileName()}'
        logger.info("Read OpenBCI file")
        logger.info(data_path)

        input_sample_rate = 250 # !!! 250 !!!
        band = (1,17)

        ADS1299_Vref = 4.5  #reference voltage for ADC in ADS1299.  set by its hardware
        ADS1299_gain = 24.  #assumed gain setting for ADS1299
        scale_fac_uVolts_per_count = ADS1299_Vref / ((float)(pow(2, 23)-1)) / ADS1299_gain *
1000000.
        input_headers                                                                    =
['id','ch1','ch2','ch3','ch4','ch5','ch6','ch7','ch8','accel1','accel2','accel3']
        output_data = []
        output_time_data = []
        output_val_data = []
        time_counter = 1
        time_increment = float(1) / float(input_sample_rate)

        with open(data_path, 'r') as csvfile:

            csv_input  =  csv.DictReader((line.replace('\0','')  for  line  in  csvfile),
fieldnames=input_headers, dialect='excel')
            row_count = 0

            for row in csv_input:
                row_count = row_count + 1
                if(row_count > 2):
                    output = {}
                    val_data = [0.] * 3
                    time_counter = time_counter + time_increment
                    output['time'] = time_counter
                    output_time_data.append(round(time_counter, 3))

                    for i in range(1, 4):
                        channel_key = 'ch'+str(i)
                        output[channel_key]    =    self.parseInt24Hex(row[channel_key])    *
scale_fac_uVolts_per_count
                        val_data[i    -    1]    =    self.parseInt24Hex(row[channel_key])    *
scale_fac_uVolts_per_count
                    output_val_data.append(val_data)
                    output_data.append(output)
```

```
                logger.debug('End read data from file')

                y_T = np.transpose(output_val_data)
                self.sampling_rate = input_sample_rate

                notch_channels = []
                for i in range(len(y_T)):
                    notch_channels.append(self.notch(50, y_T[i], fs = self.sampling_rate))

                bandpass_notch_channels = []
                for i in range(len(notch_channels)):
                    bandpass_notch_channels.append(self.bandpass(band[0],band[1],notch_channels[i],
fs = self.sampling_rate))

                self.signals = bandpass_notch_channels

                logger.info(f'Sampling rate: {self.sampling_rate}')


            def parseInt24Hex(self, hex):
                if hex is None:
                    return 0.
                if (hex[:1] > '7'):
                    hex = "FF" + hex
                else:
                    hex = "00" + hex
                n = int(hex, 16) & 0xffffffff
                return n | (-(n & 0x80000000))

            def bandpass(self, start, stop, data, fs):
                bp_Hz = np.array([start, stop])
                b, a = signal.butter(5, bp_Hz / (fs / 2.0), btype='bandpass')
                return signal.lfilter(b, a, data, axis=0)

            def notch(self, val, data, fs):
                notch_freq_Hz = np.array([float(val)])
                for freq_Hz in np.nditer(notch_freq_Hz):
                    bp_stop_Hz = freq_Hz + 3.0 * np.array([-1, 1])
                    b, a = signal.butter(3, bp_stop_Hz / (fs / 2.0), 'bandstop')
                    fin = data = signal.lfilter(b, a, data)
                return fin
```

Файл read_physionet_file.py (Опис формату PhysioNet)

```
from loguru import logger
import numpy as np
import wfdb

class ReadPhysionetFile:

    def __init__(self, ecg_config):
        # data_path = f'{ecg_config.getDataPath()}/{ecg_config.getFileName()}'
```

```
        data_path = f'{ecg_config.getFileName()}'
        logger.info("Read physionet file")
        logger.info(data_path)

        signals, self.fileds = wfdb.rdsamp(data_path)

        self.sampling_rate = self.fileds['fs']
        self.signals = signals.transpose()

        bandpass_notch_channels = []
        for i in self.signals:
            bandpass_notch_channels.append(self.bandpass(i, fs = self.sampling_rate))

        self.signals = bandpass_notch_channels

        logger.info(f'Fileds: {self.fileds["sig_name"]}')
        logger.info(f'Sampling rate: {self.sampling_rate}')


    def bandpass(self, data, fs):
        m = np.mean(data)
        data = (data - m)
        t = 1
        if np.max(data) > 1000:
            t = 1000.0
        return  data / t
```

Файл read_mat_file.py (Опис формату MATLAB)

```
from loguru import logger
import scipy.io

class ReadMatFile:

    def __init__(self, ecg_config):
        data_path = f'{ecg_config.getFileName()}.{ecg_config.getDataType()}'
        logger.info("Read Mat file")
        logger.info(data_path)

        data = scipy.io.loadmat(data_path)

        if "signal" in data:

            self.signals = data["signal"].transpose()

            result = [arr.item().strip() for arr in data["siginfo"]["Description"][0]]

            self.sampling_rate = int((data["Fs"])[0][0])
            logger.info(f'Fileds: {result}')
            logger.info(f'Sampling rate: {self.sampling_rate}')

        elif "OUTPUT" in data:
```

```
                output = self.get_nested(data, "OUTPUT", "ECGanalysis", 0, 0, "x", 0, 0)
                self.FourierSeries_FourierCoefficients = self.get_nested(output, "FourierSeries",
0, 0, "FourierCoefficients", 0, 0, 0) # Fourier coefficients
                self.FourierSeries_Harmonics = self.get_nested(output, "FourierSeries", 0, 0,
"Harmonics", 0, 0, 0) # Harmonics
                self.cycRSCP1_Ryxa = self.get_nested(output, "cycRSCP1", 0, 0, "Ryxa", 0, 0, 0)
# cyclic autocorrelation of x(t)
                self.cycRSCP1_Cyxa = self.get_nested(output, "cycRSCP1", 0, 0, "Cyxa", 0, 0, 0)
# cyclic autocovaraince of x(t)
                self.cycRSCP1_Tau1 = self.get_nested(output, "cycRSCP1", 0, 0, "Tau1", 0, 0, 0)
# tau axis for cyclic autocorrelation and autocovariance
                self.cycRSCP1_Syxa = self.get_nested(output, "cycRSCP1", 0, 0, "Syxa", 0, 0, 0)
# cyclic spectrum
                self.cycRSCP1_Pyxa = self.get_nested(output, "cycRSCP1", 0, 0, "Pyxa", 0, 0, 0)
# 2nd-order cyclic polyspectrum
                self.cycRSCP1_F1 = self.get_nested(output, "cycRSCP1", 0, 0, "F1", 0, 0, 0)     #
frequency axis for cyclic spectrum and 2nd-order cyclic polyspectrum

                self.stationary_cycRSCP1_Ryxa = self.get_nested(output, "stationary", 0, 0,
"cycRSCP1", 0, 0, "Ryxa", 0, 0, 0)  # autocorrelation of x(t)
                self.stationary_cycRSCP1_Cyxa = self.get_nested(output, "stationary", 0, 0,
"cycRSCP1", 0, 0, "Cyxa", 0, 0, 0)  # autocovariance of x(t)
                self.stationary_cycRSCP1_Tau1 = self.get_nested(output, "stationary", 0, 0,
"cycRSCP1", 0, 0, "Tau1", 0, 0, 0)  # tau axis for autocorrelation and autocovariance
                self.stationary_cycRSCP1_Syxa = self.get_nested(output, "stationary", 0, 0,
"cycRSCP1", 0, 0, "Syxa", 0, 0, 0)  # power spectral density
                self.stationary_cycRSCP1_Pyxa = self.get_nested(output, "stationary", 0, 0,
"cycRSCP1", 0, 0, "Pyxa", 0, 0, 0)  # 2nd-order polyspectrum
                self.stationary_cycRSCP1_F1 = self.get_nested(output, "stationary", 0, 0,
"cycRSCP1", 0, 0, "F1", 0, 0, 0)     # frequency axis for power spectral density and 2nd-order
polyspectrum
            else:
                self.RD_Ryxa = self.get_nested(data, "OUTPUT_RD", 0, 0, "Ryxa", 0) # cyclic
autocorrelation of x(t)
                self.RD_Tau1 = self.get_nested(data, "OUTPUT_RD", 0, 0, "Tau1", 0) # tau axis for
cyclic autocorrelation
                self.RD_Pyxa = self.get_nested(data, "OUTPUT_RD", 0, 0, "Pyxa", 0) # 2nd-order
cyclic polyspectrum
                self.RD_F1 = self.get_nested(data, "OUTPUT_RD", 0, 0, "F1", 0) # frequency axis
for 2nd-order cyclic polyspectrum
                self.RD_FourierCoefficients = self.get_nested(data, "OUTPUT_RD", 0, 0,
"FourierCoefficients", 0) # Fourier coefficients
                self.RD_Harmonics = self.get_nested(data, "OUTPUT_RD", 0, 0, "Harmonics", 0) #
Harmonics


        def get_nested(self, data, *path):
            for key in path:
                try:
                    data = data[key]
                except (KeyError, IndexError):
                    return None
            return data
```

Файл generate_rhythm_function_ecg.py (Функція ритму для ЕКГ)

```
from loguru import logger
from my_helpers.read_data.read_data_file import ReadDataFile
import pandas as pd
import neurokit2 as nk
import numpy as np
from pathlib import Path
import matplotlib.pyplot as plt

class GenerateRhythmFunction(ReadDataFile):

    def __init__(self, ecg_config):
        super().__init__(ecg_config)

    def genFr(self):
        logger.info("Get ECG Peaks")
        path = f'{self.ecg_config.getImgPath()}/{self.ecg_config.getConfigBlock()}'
        path_all = f'{path}/FR-{self.ecg_config.getConfigBlock()}.csv'
        Path(path).mkdir(parents=True, exist_ok=True)
        if Path(path_all).is_file():
            logger.warning(f'File {path_all} is exist. Create a backup and continue')
            return
        i = self.ecg_config.getSigName()

        _, rpeaks = nk.ecg_peaks(self.signals[i], sampling_rate=self.sampling_rate)
        _,         waves         =         nk.ecg_delineate(self.signals[i],         rpeaks,
sampling_rate=self.sampling_rate)
        ECG_P_Peaks = list(np.round(np.array(waves["ECG_P_Peaks"]) / self.sampling_rate, 4))
        ECG_Q_Peaks = list(np.round(np.array(waves["ECG_Q_Peaks"]) / self.sampling_rate, 4))
        ECG_R_Peaks = list(np.round(np.array(rpeaks["ECG_R_Peaks"]) / self.sampling_rate, 4))
        ECG_S_Peaks = list(np.round(np.array(waves["ECG_S_Peaks"]) / self.sampling_rate, 4))
        ECG_T_Peaks = list(np.round(np.array(waves["ECG_T_Peaks"]) / self.sampling_rate, 4))

        ecg_fr  =  pd.DataFrame({"ECG_P_Peaks"  :  ECG_P_Peaks,  "ECG_Q_Peaks"  :  ECG_Q_Peaks,
"ECG_R_Peaks" : ECG_R_Peaks, "ECG_S_Peaks" : ECG_S_Peaks, "ECG_T_Peaks" : ECG_T_Peaks})
        nk.write_csv(ecg_fr, path_all)

    def genIntervals(self):
        logger.info("Gen All ECG Intervals")
        path = f'{self.ecg_config.getImgPath()}/{self.ecg_config.getConfigBlock()}/Intervals'
        Path(path).mkdir(parents=True, exist_ok=True)
        path_all = f'{path}/All.csv'
        if Path(path_all).is_file():
            logger.warning(f'File {path_all} is exist. Create a backup and continue')
            return

        i = self.ecg_config.getSigName()

        _, rpeaks = nk.ecg_peaks(self.signals[i], sampling_rate=self.sampling_rate)
```

```
                _,          waves          =          nk.ecg_delineate(self.signals[i],          rpeaks,
sampling_rate=self.sampling_rate)

                ECG_P_Onsets = list(np.array(waves["ECG_P_Onsets"]) / self.sampling_rate)
                ECG_P_Peaks = list(np.array(waves["ECG_P_Peaks"]) / self.sampling_rate)
                ECG_P_Offsets = list(np.array(waves["ECG_P_Offsets"]) / self.sampling_rate)
                # ECG_R_Onsets = list(np.array(waves["ECG_R_Onsets"]) / self.sampling_rate)
                ECG_Q_Peaks = list(np.array(waves["ECG_Q_Peaks"]) / self.sampling_rate)
                ECG_R_Peaks = list(np.array(rpeaks["ECG_R_Peaks"]) / self.sampling_rate)
                ECG_S_Peaks = list(np.array(waves["ECG_S_Peaks"]) / self.sampling_rate)
                # ECG_R_Offsets = list(np.array(waves["ECG_R_Offsets"]) / self.sampling_rate)
                ECG_T_Onsets = list(np.array(waves["ECG_T_Onsets"]) / self.sampling_rate)
                ECG_T_Peaks = list(np.array(waves["ECG_T_Peaks"]) / self.sampling_rate)
                ECG_T_Offsets = list(np.array(waves["ECG_T_Offsets"]) / self.sampling_rate)

                # ecg_fr = pd.DataFrame({"ECG_P_Onsets" : ECG_P_Onsets, "ECG_P_Peaks" : ECG_P_Peaks,
"ECG_P_Offsets" : ECG_P_Offsets,
                #                        "ECG_Q_Peaks" : ECG_Q_Peaks,
                #                        "ECG_R_Peaks" : ECG_R_Peaks,
                #                        "ECG_S_Peaks" : ECG_S_Peaks,
                #                         "ECG_T_Onsets" : ECG_T_Onsets, "ECG_T_Peaks" : ECG_T_Peaks,
"ECG_T_Offsets" : ECG_T_Offsets})
                ecg_fr = pd.DataFrame({"ECG_P_Peaks" : ECG_P_Peaks,
                                       "ECG_Q_Peaks" : ECG_Q_Peaks,
                                       "ECG_R_Peaks" : ECG_R_Peaks,
                                       "ECG_S_Peaks" : ECG_S_Peaks,
                                       "ECG_T_Peaks" : ECG_T_Peaks,})

                nk.write_csv(ecg_fr, path_all)

            def plotECG(self):
                logger.info("Plot ECG")
                path = f'{self.ecg_config.getImgPath()}/{self.ecg_config.getConfigBlock()}/Intervals'
                Path(path).mkdir(parents=True, exist_ok=True)

                start = 0
                end = 5000
                ECG_data = np.round(self.signals[self.ecg_config.getSigName()], 4)
                time = np.arange(0, len(ECG_data), 1) / self.sampling_rate

                # data = pd.DataFrame({"ECG_data" : ECG_data})
                # data.index = time
                # data.index.name = "Time"
                # nk.write_csv(data, f'{path}/ECG_data.csv')

                plt.clf()
                plt.rcParams.update({'font.size': 15})
                f, axis = plt.subplots(1)
                f.tight_layout()
                f.set_size_inches(19, 6)
                axis.grid(True)
                axis.plot(time, ECG_data, linewidth=3, label=r"$\xi_{{\omega}} (t), mV$")
                axis.set_xlabel("$t, s$", loc = 'right')
```

```
        axis.legend(loc='upper right')
        # axis.axis(ymin = -6, ymax = 6)
        axis.axis(xmin = 0.3, xmax = 9.5)
        plt.savefig(f'{path}/ECG_data.png', dpi=300)

    def plotFr(self, debug = False):
        logger.info("Plot a rhythm function")

        self.getData()

        plot_path = f'{self.ecg_config.getFrImgPath()}/{self.ecg_config.getConfigBlock()}'
        Path(plot_path).mkdir(parents=True, exist_ok=True)

        T1_ECG_T_Peaks = []
        T1_ECG_P_Peaks = []
        T1_ECG_R_Peaks = []
        T1_ECG_S_Peaks = []
        T1_ECG_Q_Peaks = []
        T1_X = []
        T1_Y = []

        for i in range(len(self.ECG_T_Peaks)-1):
            T1_ECG_T_Peaks.append(round(self.ECG_T_Peaks[i+1] - self.ECG_T_Peaks[i], 2))

        for i in range(len(self.ECG_P_Peaks)-1):
            T1_ECG_P_Peaks.append(round(self.ECG_P_Peaks[i+1] - self.ECG_P_Peaks[i], 2))

        for i in range(len(self.ECG_R_Peaks)-1):
            T1_ECG_R_Peaks.append(round(self.ECG_R_Peaks[i+1] - self.ECG_R_Peaks[i], 2))

        for i in range(len(self.ECG_P_Peaks) - 1):
            T1_X.append(self.ECG_P_Peaks[i])
            T1_X.append(self.ECG_R_Peaks[i])
            T1_X.append(self.ECG_T_Peaks[i])

        for i in range(len(T1_ECG_P_Peaks)):
            T1_Y.append(T1_ECG_P_Peaks[i])
            T1_Y.append(T1_ECG_R_Peaks[i])
            T1_Y.append(T1_ECG_T_Peaks[i])

        # T1_Y = list(map(lambda x: x - 0.81413, T1_Y))

        plt.clf()
        plt.rcParams.update({'font.size': 14})
        f, axis = plt.subplots(1)
        f.tight_layout()
        f.set_size_inches(19, 6)
        axis.grid(True)
        axis.plot(T1_X, T1_Y, linewidth=3)
        axis.set_xlabel("$t, s$", loc = 'right')
        axis.legend(['$T(t, 1), s$'])
        axis.axis(ymin = -0.2, ymax = 1.2)
        axis.axis(xmin = T1_X[0], xmax = T1_X[-1])
```

```
        plt.savefig(f'{plot_path}/FR-{self.ecg_config.getConfigBlock()}.png', dpi=300)
```

Файл rhythm_function.py (Функція ритму для ЕЕГ)

```
from loguru import logger
from my_helpers.read_data.read_data_file import ReadDataFile
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np

class RhythmFunction(ReadDataFile):

    def __init__(self, ecg_config):
        super().__init__(ecg_config)
        self.plot_path = f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}'
        Path(self.plot_path).mkdir(parents=True, exist_ok=True)

    def plotFr(self):
        logger.info("Plot a rhythm function")

        self.getData()

        T1_D_c = []
        T1_D_z = []
        T1_X = []
        T1_Y = []

        for i in range(len(self.D_c) - 1):
            T1_D_c.append(round(self.D_c[i+1] - self.D_c[i], 2))

        for i in range(len(self.D_z) - 1):
            T1_D_z.append(round(self.D_z[i+1] - self.D_z[i], 2))

        for i in range(len(self.D_c) - 1):
            T1_X.append(self.D_c[i])
            T1_X.append(self.D_z[i])

        for i in range(len(T1_D_c)):
            T1_Y.append(T1_D_c[i])
            T1_Y.append(T1_D_z[i])

        plt.clf()
        plt.rcParams.update({'font.size': 14})
        f, axis = plt.subplots(1)
        f.tight_layout()
        f.set_size_inches(10, 6)
        axis.grid(True)
        axis.plot(T1_X, T1_Y, linewidth=2)
        axis.set_xlabel("$t, s$", loc = 'right')
        axis.legend(['$T(t, 1), s$'])
        axis.axis(ymin = 0, ymax = round(np.max(T1_Y)) + 0.5)
        axis.axis(xmin = T1_X[0], xmax = T1_X[-1])
```

```
                plt.savefig(f"{self.plot_path}/function_rhythm.png", dpi=300)

            def plotEEG(self):
                logger.info("Plot a EEG")

                self.getData()

                f, axis = plt.subplots(len(self.signals))
                f.tight_layout()
                f.set_size_inches(17, 9)
                time = np.arange(0, len(self.signals[0]), 1) / 250

                xtext = "$t, s$"
                ytext=r"$\xi_{{\omega {}}} (t), \mu V$"

                for i in range(len(self.signals)):
                    axis[i].grid(True)
                    axis[i].plot(time, self.signals[i])
                    axis[i].set_xlabel(xtext, loc = 'right')
                    axis[i].set_title(ytext.format(i+1), loc = 'left', fontsize=10, position=(-0.07,
0))

                for i in range(len(self.signals)):
                    axis[i].axis(xmin = 7.2, xmax = 20)
                    axis[i].axis(ymin = -10, ymax = 10)
                plt.savefig(f"{self.plot_path}/Графік.png", dpi=300)
```

Файл data_preparation.py (Підготовка даних ЕЕГ)

```
from loguru import logger
from my_helpers.read_data.read_data_file import ReadDataFile
import numpy as np
import scipy.interpolate as interp

class DataPreparation(ReadDataFile):

    def __init__(self, eeg_config):
        super().__init__(eeg_config)
        self.getData()

        self.mod_sampling_rate = int(self.sampling_rate * self.eeg_config.getMultiplier())

        matrix_activity_size = matrix_passivity_size = self.mod_sampling_rate

        self.interp_matrix_passivity = [[] for _ in range(len(self.signals))]
        self.interp_matrix_activity = [[] for _ in range(len(self.signals))]

        self.interp_matrix_passivity       =       self.interp_matrix(self.matrix_passivity,
matrix_passivity_size)
        self.interp_matrix_activity        =        self.interp_matrix(self.matrix_activity,
matrix_activity_size)
```

```
    def interp_matrix(self, matrix, size):
        interpolated_matrix = []
        for channel in matrix:
            interp_channel = []
            for segment in channel:
                arr = np.array(segment)
                arr_interp = interp.interp1d(np.arange(arr.size), arr)
                arr_stretch = arr_interp(np.linspace(0, arr.size - 1, size))
                interp_channel.append(arr_stretch)
            interpolated_matrix.append(interp_channel)
        return interpolated_matrix

    def getNewMatrixSize(self, matrix):
        n = 0
        for i in range(len(matrix)):
            n = n + len(matrix[i])
        n = int((n / len(matrix)) * self.eeg_config.getMultiplier())
        return n

    def getModSamplingRate(self):
        return self.mod_sampling_rate

    def getPreparedData(self):
        return self.interp_matrix_passivity, self.interp_matrix_activity
```

Файл data_preparation.py (Підготовка даних ЕКГ)

```
from loguru import logger
from my_helpers.read_data.read_data_file import ReadDataFile
import numpy as np
import scipy.interpolate as interp
from pathlib import Path
import matplotlib.pyplot as plt

class DataPreparation(ReadDataFile):

    def __init__(self, ecg_config):
        super().__init__(ecg_config)
        self.getData()

        self.mod_sampling_rate = int(self.sampling_rate * self.ecg_config.getMultiplier())

        matrix_T_P_size = self.getNewMatrixSize(self.matrix_T_P)
        matrix_P_R_size = self.getNewMatrixSize(self.matrix_P_R)
        matrix_R_T_size = self.getNewMatrixSize(self.matrix_R_T)

        interp_matrix_T_P = []
        interp_matrix_P_R = []
        interp_matrix_R_T = []
        self.interp_matrix_all = []

        for i in range(len(self.matrix_T_P)):
```

```
                arr = np.array(self.matrix_T_P[i])
                arr_interp = interp.interp1d(np.arange(arr.size), arr)
                arr_stretch = arr_interp(np.linspace(0, arr.size - 1, matrix_T_P_size))
                interp_matrix_T_P.append(arr_stretch)

            for i in range(len(self.matrix_P_R)):
                arr = np.array(self.matrix_P_R[i])
                arr_interp = interp.interp1d(np.arange(arr.size), arr)
                arr_stretch = arr_interp(np.linspace(0, arr.size - 1, matrix_P_R_size))
                interp_matrix_P_R.append(arr_stretch)

            for i in range(len(self.matrix_R_T)):
                arr = np.array(self.matrix_R_T[i])
                arr_interp = interp.interp1d(np.arange(arr.size), arr)
                arr_stretch = arr_interp(np.linspace(0, arr.size - 1, matrix_R_T_size))
                interp_matrix_R_T.append(arr_stretch)

            interp_matrix_all = np.concatenate((interp_matrix_P_R, interp_matrix_R_T,
interp_matrix_T_P), axis=1)

            for i in range(len(interp_matrix_all)):
                arr = np.array(interp_matrix_all[i])
                arr_interp = interp.interp1d(np.arange(arr.size), arr)
                arr_stretch = arr_interp(np.linspace(0, arr.size - 1, self.mod_sampling_rate))
                self.interp_matrix_all.append(arr_stretch)

        def plotAllCycles(self):
            plot_path = f'{self.ecg_config.getImgPath()}/{self.getSigNameDir()}'

            Path(plot_path).mkdir(parents=True, exist_ok=True)
            plt.clf()
            plt.rcParams.update({'font.size': 14})
            f, axis = plt.subplots(1)
            f.tight_layout()
            f.set_size_inches(19, 6)
            axis.grid(True)
            axis.set_xlabel("$t, s$", loc = 'right')
            time = np.arange(0, len(self.interp_matrix_all[0]), 1) / self.mod_sampling_rate
            for i in self.interp_matrix_all:
                axis.plot(time, i, linewidth=2)
            plt.savefig(f'{plot_path}/__All_Cycles.png', dpi=300)


        def getNewMatrixSize(self, matrix):
            n = 0
            for i in range(len(matrix)):
                n = n + len(matrix[i])
            n = int((n / len(matrix)) * self.ecg_config.getMultiplier())
            # n = int(len(matrix[0]) * self.ecg_config.getMultiplier())
            return n

        def getModSamplingRate(self):
            return self.mod_sampling_rate
```

```
    def getPreparedData(self):
        return self.interp_matrix_all
```

Файл confusion_matrix.py (Розрахунок матриці невідповідностей)

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import class_likelihood_ratios
import numpy as np

class ConfusionMatrix():

    def __init__(self, y_true, y_pred, ltime, ttime):

        CM = confusion_matrix(y_true, y_pred)
        TN, FP, FN, TP = CM.ravel()

        # True Positive Rate
        TPR = TP / (TP + FN)

        # True Negative Rate
        TNR = TN / (TN + FP)

        # Positive Predictive Value
        PPV = TP / (TP + FP)

        # Negative Predictive Value
        NPV = TN / (TN + FN)

        # False Negative Rate
        FNR = FN / (FN + TP)

        # False Positive Rate
        FPR = FP / (FP + TN)

        # False Discovery Rate
        FDR = FP / (FP + TP)

        # False Omission Rate
        FOR = FN / (FN + TN)

        # Positive Likelihood Ratio
        LR_P_ = TPR / FPR

        # Negative Likelihood Ratio
        LR_N_ = FNR / TNR

        # Prevalence Threshold
        PT = np.sqrt(FPR) / (np.sqrt(TPR) + np.sqrt(FPR))

        # Threat Score
```

```
                TS = TP / (TP + FN + FP)

                # Accuracy
                ACC = (TP + TN) / (TP + TN + FP + FN)

                # Balanced Accuracy
                BA = (TPR + TNR) / 2.0

                # F1 score
                F1 = 2 * ((PPV * TPR) / (PPV + TPR))

                # Matthews Correlation Coefficient
                # MCC = ((TP * TN) - (FP * FN)) / (np.sqrt((TP + FP) * (TP + FN) * (TN + FP) * (TN +
FN)))
                MCC = matthews_corrcoef(y_true, y_pred)

                # Fowlkes-Mallows index
                FM = np.sqrt(PPV * TPR)

                # Bookmaker Informedness
                BM = TPR + TNR - 1

                # Markedness
                MK = PPV + NPV - 1

                # print(class_likelihood_ratios(y_true, y_pred))

                # Diagnostic Odds Ratio
                DPR = LR_P_ / LR_N_

                self.CM = CM
                self.TN = TN
                self.FP = FP
                self.FN = FN
                self.TP = TP
                self.TPR = TPR
                self.TNR = TNR
                self.PPV = PPV
                self.NPV = NPV
                self.FNR = FNR
                self.FPR = FPR
                self.FDR = FDR
                self.FOR = FOR
                self.LR_P_ = LR_P_
                self.LR_N_ = LR_N_
                self.PT = PT
                self.TS = TS
                self.ACC = ACC
                self.BA = BA
                self.F1 = F1
                self.MCC = MCC
                self.FM = FM
                self.BM = BM
```

```
        self.MK = MK
        self.DPR = DPR
        self.ltime = ltime
        self.ttime = ttime

    def getCM(self):
        return self.CM

    def getTN(self):
        return self.TN

    def getFP(self):
        return self.FP

    def getFN(self):
        return self.FN

    def getTP(self):
        return self.TP

    def getTPR(self):
        return self.TPR

    def getTNR(self):
        return self.TNR

    def getPPV(self):
        return self.PPV

    def getNPV(self):
        return self.NPV

    def getFNR(self):
        return self.FNR

    def getFPR(self):
        return self.FPR

    def getFDR(self):
        return self.FDR

    def getFOR(self):
        return self.FOR

    def getLR_P_(self):
        return self.LR_P_

    def getLR_N_(self):
        return self.LR_N_

    def getPT(self):
        return self.PT
```

```
        def getTS(self):
            return self.TS

        def getACC(self):
            return self.ACC

        def getBA(self):
            return self.BA

        def getF1(self):
            return self.F1

        def getMCC(self):
            return self.MCC

        def getFM(self):
            return self.FM

        def getBM(self):
            return self.BM

        def getMK(self):
            return self.MK

        def getDPR(self):
            return self.DPR

        def getAllVariables(self):
            return [
                self.TPR, self.TNR, self.PPV, self.NPV,
                self.FNR, self.FPR, self.FDR, self.FOR, self.LR_P_, self.LR_N_, self.PT, self.TS,
                self.ACC,  self.BA,  self.F1,  self.MCC,  self.FM,  self.BM,  self.MK,  self.DPR,
np.round(self.ltime, 3), np.round(self.ttime, 3)
            ]
```

Файли mathematical_statistics.py та mathematical_statistics_data.py (Розрахунок початкових та центральних моментних функцій)

```
from my_helpers.mathematical_statistics_data import MathematicalStatisticsData
import numpy as np
from scipy.integrate import simps

class MathematicalStatistics(MathematicalStatisticsData):
    def __init__(self, data):
        data = np.transpose(data)
        #Mathematical expectation
        self.m_ = [np.mean(i) for i in data]
        #Initial moments of the second order
        self.m_2_ = [np.sum(np.array(i)**2) / len(i) for i in data]
        #Initial moments of the third order
        self.m_3_ = [np.sum(np.array(i)**3) / len(i) for i in data]
```

```
                #Initial moments of the fourth order
                self.m_4_ = [np.sum(np.array(i)**4) / len(i) for i in data]
                #Variance
                self.m__2   =   [sum((data[i]   -   self.m_[i])**2)   /   len(data[i])   for   i   in
range(len(self.m_))]
                #Central moment functions of the fourth order
                self.m__4   =   [sum((data[i]   -   self.m_[i])**4)   /   len(data[i])   for   i   in
range(len(self.m_))]

            def getMathematicalStatistics(self):
                return   MathematicalStatisticsData(self.m_,   self.m_2_,   self.m_3_,   self.m_4_,
self.m__2, self.m__4)

            def setSamplingRate(self, sampling_rate):
                self.sampling_rate = sampling_rate

        class MathematicalStatisticsData():

            def __init__(self, m_, m_2_, m_3_, m_4_, m__2, m__4):
                self.m_ = m_
                self.m_2_ = m_2_
                self.m_3_ = m_3_
                self.m_4_ = m_4_
                self.m__2 = m__2
                self.m__4 = m__4

            def getMathematicalExpectation(self):
                return self.m_

            def getInitialMomentsSecondOrder(self):
                return self.m_2_

            def getInitialMomentsThirdOrder(self):
                return self.m_3_

            def getInitialMomentsFourthOrder(self):
                return self.m_4_

            def getVariance(self):
                return self.m__2

            def getCentralMomentFunctionsFourthOrder(self):
                return self.m__4
```

Файл plot_statistics.py (Вивід результатів розрахунків початкових та центральних моментних функцій)

```
from my_helpers.mathematical_statistics_data import MathematicalStatisticsData
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
```

```
        from loguru import logger
        import neurokit2 as nk
        import pandas as pd

        class PlotStatistics():
            def __init__(self, statistics, sampling_rate, ecg_config, zone_type):
                self.sampling_rate = sampling_rate
                self.ecg_config = ecg_config
                self.statistics = statistics
                self.zone_type = zone_type
                self.plot_path = f'{self.ecg_config.getImgPath()}/{self.ecg_config.getConfigBlock()}'

            def plotAllStatistics(self):
                logger.info("Plot Mathematical Statistics")
                mathematical_statistics = [statistic.getMathematicalStatistics() for statistic in
self.statistics]
                xtext = "$t, s$"
                self.plot_to_png([statistic.getMathematicalExpectation() for statistic in
mathematical_statistics], f"1 Mathematical Expectation {self.zone_type}", xtext=xtext,
ytext=r"$m_{{\xi {}}} (t), \mu V$", ylim=(-4.5, 4.5))
                self.plot_to_png([statistic.getInitialMomentsSecondOrder() for statistic in
mathematical_statistics], f"2 Initial Moments Second Order {self.zone_type}", xtext=xtext,
ytext=r"$d_{{\xi {}}} (t), \mu V^2$")
                self.plot_to_png([statistic.getInitialMomentsThirdOrder() for statistic in
mathematical_statistics], f"3 Initial Moments Third Order {self.zone_type}", xtext=xtext,
ytext=r"$d_{{\xi {}}} (t), \mu V^3$")
                self.plot_to_png([statistic.getInitialMomentsFourthOrder() for statistic in
mathematical_statistics], f"4 Initial Moments Fourth Order {self.zone_type}", xtext=xtext,
ytext=r"$d_{{\xi {}}} (t), \mu V^4$")
                self.plot_to_png([statistic.getVariance() for statistic in mathematical_statistics],
f"5 Variance {self.zone_type}", xtext=xtext, ytext=r"$d_{{\xi {}}} (t), \mu V^2$")
                self.plot_to_png([statistic.getCentralMomentFunctionsFourthOrder() for statistic in
mathematical_statistics], f"6 Central Moment Functions Fourth Order {self.zone_type}", xtext=xtext,
ytext=r"$d_{{\xi {}}} (t), \mu V^4$")

                self.plot_to_csv([statistic.getMathematicalExpectation() for statistic in
mathematical_statistics], f"1 Mathematical Expectation {self.zone_type}")
                self.plot_to_csv([statistic.getInitialMomentsSecondOrder() for statistic in
mathematical_statistics], f"2 Initial Moments Second Order {self.zone_type}")
                self.plot_to_csv([statistic.getInitialMomentsThirdOrder() for statistic in
mathematical_statistics], f"3 Initial Moments Third Order {self.zone_type}")
                self.plot_to_csv([statistic.getInitialMomentsFourthOrder() for statistic in
mathematical_statistics], f"4 Initial Moments Fourth Order {self.zone_type}")
                self.plot_to_csv(mathematical_statistics.getVariance(), "5 Variance")
                self.plot_to_csv(mathematical_statistics.getCentralMomentFunctionsFourthOrder(), "6
Central Moment Functions Fourth Order")

            def plotAllFourierStatistics(self):
                logger.info("Plot Mathematical Statistics Fourier")
                self.statistics = [statistic.setSamplingRate(self.sampling_rate) or statistic for
statistic in self.statistics]
                mathematical_statistics = [statistic.getMathematicalStatisticsFourierSeries() for
statistic in self.statistics]
```

```
                xtext = "$n$"
                self.fs_plot_to_png([statistic.getMathematicalExpectation()    for    statistic    in
mathematical_statistics], f"1 Mathematical Expectation {self.zone_type}", xtext=xtext, ytext=(r"$a_n,
\mu V$", r"$b_n, \mu V$"))
                self.fs_plot_to_png([statistic.getInitialMomentsSecondOrder()    for    statistic    in
mathematical_statistics],  f"2  Initial  Moments  Second  Order  {self.zone_type}",  xtext=xtext,
ytext=(r"$a_n, \mu V^2$", r"$b_n, \mu V^2$"))
                self.fs_plot_to_png([statistic.getInitialMomentsThirdOrder()    for    statistic    in
mathematical_statistics],  f"3  Initial  Moments  Third  Order  {self.zone_type}",  xtext=xtext,
ytext=(r"$a_n, \mu V^3$", r"$b_n, \mu V^3$"))
                self.fs_plot_to_png([statistic.getInitialMomentsFourthOrder()    for    statistic    in
mathematical_statistics],  f"4  Initial  Moments  Fourth  Order  {self.zone_type}",  xtext=xtext,
ytext=(r"$a_n, \mu V^4$", r"$b_n, \mu V^4$"))
                self.fs_plot_to_png([statistic.getVariance()         for         statistic         in
mathematical_statistics], f"5 Variance {self.zone_type}", xtext=xtext, ytext=(r"$a_n, \mu V^2$",
r"$b_n, \mu V^2$"))
                self.fs_plot_to_png([statistic.getCentralMomentFunctionsFourthOrder() for statistic
in  mathematical_statistics],  f"6  Central  Moment  Functions  Fourth  Order  {self.zone_type}",
xtext=xtext, ytext=(r"$a_n, \mu V^4$", r"$b_n, \mu V^4$"))

                self.fs_plot_to_csv(mathematical_statistics.getMathematicalExpectation(),          "1
Mathematical Expectation")
                self.fs_plot_to_csv(mathematical_statistics.getInitialMomentsSecondOrder(),        "2
Initial Moments Second Order")
                self.fs_plot_to_csv(mathematical_statistics.getInitialMomentsThirdOrder(), "3 Initial
Moments Third Order")
                self.fs_plot_to_csv(mathematical_statistics.getInitialMomentsFourthOrder(),        "4
Initial Moments Fourth Order")
                self.fs_plot_to_csv(mathematical_statistics.getVariance(), "5 Variance")
                self.fs_plot_to_csv(mathematical_statistics.getCentralMomentFunctionsFourthOrder(),
"6 Central Moment Functions Fourth Order")

            def fs_plot_to_png(self, plot2, name, xlim = None, ylim = None, ytext=(r"$a_n, \mu V$",
r"$b_n, \mu V$"), xtext="", size=(9, 9)):
                path = f'{self.plot_path}/Mathematical Statistics Fourier'
                Path(path).mkdir(parents=True, exist_ok=True)
                logger.info(f"Plot {name} an.png")
                plt.clf()
                plt.rcParams.update({'font.size': 14})
                f, axis = plt.subplots(len(plot2))
                f.tight_layout()
                f.set_size_inches(size)
                for i in range(len(plot2)):
                    an, _ = plot2[i]
                    axis[i].set_xlabel(xtext, loc = 'right')
                    axis[i].set_title(ytext[0], loc = 'left', fontsize=15, position=(-0.05, 0))
                    axis[i].grid(True)
                    _, stemlines, _ = axis[i].stem([0, *an[1:]])
                    plt.setp(stemlines, 'linewidth', 2)
                plt.savefig(f'{path}/{name} an.png', dpi=300)

                logger.info(f"Plot {name} bn.png")
```

```
                plt.clf()
                plt.rcParams.update({'font.size': 14})
                f, axis = plt.subplots(len(plot2))
                f.tight_layout()
                f.set_size_inches(size)
                for i in range(len(plot2)):
                    _, bn = plot2[i]
                    axis[i].set_xlabel(xtext, loc = 'right')
                    axis[i].set_title(ytext[1], loc = 'left', fontsize=15, position=(-0.05, 0))
                    axis[i].grid(True)
                    _, stemlines, _ = axis[i].stem([0, *bn])
                    plt.setp(stemlines, 'linewidth', 2)
                plt.savefig(f'{path}/{name} bn.png', dpi=300)

            def plot_to_png(self, plot, name, xlim = None, ylim = None, ytext="", xtext="", size=(7,
9)):
                logger.info(f"Plot {name}.png")
                path = f'{self.plot_path}/Mathematical Statistics'
                Path(path).mkdir(parents=True, exist_ok=True)
                plt.clf()
                # plt.rcParams.update({'font.size': 14})
                f, axis = plt.subplots(len(plot))
                f.tight_layout()
                f.set_size_inches(size[0], size[1])
                time = np.arange(0, len(plot[0]), 1) / self.sampling_rate
                for i in range(len(plot)):
                    axis[i].grid(True)
                    axis[i].plot(time, plot[i], linewidth=2)
                    axis[i].set_xlabel(xtext, loc = 'right')
                    axis[i].set_title(ytext.format(i+1), loc = 'left', fontsize=10, position=(-0.07,
0))
                    if xlim is not None:
                        axis[i].axis(xmin = xlim[0], xmax = xlim[1])
                    if ylim is not None:
                        axis[i].axis(ymin = ylim[0], ymax = ylim[1])
                plt.savefig(f'{path}/{name}.png', dpi=300)

            def plot_to_csv(self, plot, name):
                logger.info(f"Save {name}.csv")
                path = f'{self.plot_path}/Mathematical Statistics/CSV'
                Path(path).mkdir(parents=True, exist_ok=True)
                plot = np.transpose(plot)
                time = np.arange(0, len(plot), 1) / self.sampling_rate
                data = pd.DataFrame({'Time': time, **{f'Data_{i}': plot[:, i] for i in
range(plot.shape[1])}})
                nk.write_csv(data, f'{path}/{name}.csv')
```

Файл classifiers.py (Класифікація)

```
import numpy as np
import time
from scipy.integrate import simps
```

```
        from sklearn.model_selection import train_test_split
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.pipeline import make_pipeline
        from sklearn.preprocessing import StandardScaler
        from sklearn.neural_network import MLPClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.svm import SVC
        from sklearn.tree import DecisionTreeClassifier
        from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
        from sklearn.naive_bayes import GaussianNB
        import pandas as pd
        from pathlib import Path

        from classification_metrics.confusion_matrix import ConfusionMatrix

        class Classifiers():

            def __init__(self, eeg_config, data, fourier_type, power, ccc):
                print("Train Classifiers")

                names = [
                    "Nearest Neighbors",
                    "Linear SVM",
                    "Decision Tree",
                    "Random Forest",
                    "Neural Net (MLP)",
                    "AdaBoost",
                    "Naive Bayes"
                ]

                confusion_matrix_names = [
                    "True Positive Rate",
                    "True Negative Rate", "Positive Predictive Value", "Negative Predictive Value",
"False Negative Rate",
                    "False Positive Rate", "False Discovery Rate", "False Omission Rate", "Positive
Likelihood Ratio",
                    "Negative Likelihood Ratio", "Prevalence Threshold", "Threat Score", "Accuracy",
"Balanced Accuracy",
                    "F1 score", "Matthews Correlation Coefficient", "Fowlkes-Mallows index",
"Bookmaker Informedness",
                    "Markedness", "Diagnostic Odds Ratio", "Learning_time", "Testing_time"
                ]

                classifiers = [
                    KNeighborsClassifier(3),
                    SVC(kernel="linear", C=0.025),
                    DecisionTreeClassifier(max_depth=5),
                    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
                    MLPClassifier(alpha=1, max_iter=1000),
                    AdaBoostClassifier(),
                    GaussianNB()
                ]
```

```
                self.sampling_rate = data.getModSamplingRate()

                channel = ccc

                data_matrix_passivity, data_matrix_activity = data.getPreparedData()
                data_matrix_passivity_1 = np.power(data_matrix_passivity[channel-1], power)
                data_matrix_activity_1 = np.power(data_matrix_activity[channel-1], power)

                # for i in [40, 999]:

                for i in [3, 4, 5, 7, 10, 15, 20, 25, 30, 35, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90,
95, 100]:

                    fa_target_matrix = [0] * len(data_matrix_passivity_1)
                    fa_target_matrix_2 = [1] * len(data_matrix_activity_1)
                    fstart = time.time()
                    if i == 999:
                        matrix = [*data_matrix_passivity_1, *data_matrix_activity_1]
                    else:
                        fa_matrix = [self.getFourierSeries(m, fourier_type, terms=i) for m in
data_matrix_passivity_1]
                        fa_matrix_2 = [self.getFourierSeries(m, fourier_type, terms=i) for m in
data_matrix_activity_1]
                        matrix = [*fa_matrix, *fa_matrix_2]
                    fend = time.time()
                    ftime = (fend-fstart)*10**3


                    target_matrix = [*fa_target_matrix, *fa_target_matrix_2]

                    data_train, data_test, target_values_train, target_values_test =
train_test_split(matrix, target_matrix, test_size=0.3, random_state=42)

                    _cm = []

                    for name, clf in zip(names, classifiers):
                        clf = make_pipeline(StandardScaler(), clf)
                        lstart = time.time()
                        clf.fit(data_train, target_values_train)
                        lend = tstart = time.time()
                        y_true = np.array(target_values_test)
                        y_pred = clf.predict(data_test)
                        tend = time.time()
                        ltime = (lend-lstart)*10**3 + ftime
                        ttime = (tend-tstart)*10**3 + ftime

                        confusion_matrix = ConfusionMatrix(y_true, y_pred, ltime, ttime)
                        res = confusion_matrix.getAllVariables()

                        _cm.append(res)

                        print(("%s: %.2f" % (name,  confusion_matrix.getACC() * 100)))
```

```
                    path    =    f'{eeg_config.getImgPath()}/{eeg_config.getConfigBlock()}/Confusion
matrix/c_{channel}/{fourier_type}/{power}'
                    Path(path).mkdir(parents=True, exist_ok=True)

                    df = pd.DataFrame(np.transpose(np.round(_cm, 2)), index=confusion_matrix_names,
columns=names)
                    df.to_csv(f'{path}/n-{i}.csv')



        def getFourierSeries(self, y, fourier_type, terms=40, L=1):
            x = np.linspace(0, L, self.sampling_rate, endpoint=False)
            a0 = 2./L*simps(y, x)
            n_values = np.arange(1, terms + 1)
            cos_vals = np.cos(2. * np.pi * n_values[:, None] * x[None, :] / L)
            sin_vals = np.sin(2. * np.pi * n_values[:, None] * x[None, :] / L)

            list_a = 2.0 / L * np.abs(np.array([simps(y * cos_n, x) for cos_n in cos_vals]))
            list_b = 2.0 / L * np.abs(np.array([simps(y * sin_n, x) for sin_n in sin_vals]))

            if fourier_type == "an":
                return [a0, *list_a]
            if fourier_type == "bn":
                return [0, *list_b]
            return [0, *list_a, *list_b]
```

Файл statistics_distance_fourier.py (Реалізація класифікатора SPC)

```
from loguru import logger
import pandas as pd
import numpy as np
import neurokit2 as nk
from pathlib import Path
import matplotlib.pyplot as plt
from scipy.integrate import simps
import re
import sys
import time

from classification_metrics.confusion_matrix import ConfusionMatrix
from my_helpers.mathematical_statistics import MathematicalStatistics

class StatisticsDistanceFourier():

    def __init__(self, eeg_config, data, fourier_type, terms, power):
        self.power = power
        self.eeg_config = eeg_config
        self.terms = terms
        self.fourier_type = fourier_type
        logger.debug("Statistics Distance")

        self.channel_type = "1"
```

```
                self.sampling_rate = data.getModSamplingRate()
                data_matrix_passivity_, data_matrix_activity_ = data.getPreparedData()
                data_matrix_activity = np.power(data_matrix_activity_, power)
                data_matrix_passivity = np.power(data_matrix_passivity_, power)

                fstart = time.time()
                self.all_matrix_passivity      =      [[self.getFourierSeries(m,      fourier_type,
terms=self.terms) for m in data_matrix_passivity[0]]]
                self.all_matrix_activity = [[self.getFourierSeries(m, fourier_type, terms=self.terms)
for m in data_matrix_activity[0]]]
                fend = time.time()
                self.ftime = (fend-fstart)*10**3

                self.confusion_matrix_names = [
                    "True Positive Rate",
                    "True Negative Rate", "Positive Predictive Value", "Negative Predictive Value",
"False Negative Rate",
                    "False Positive Rate", "False Discovery Rate", "False Omission Rate", "Positive
Likelihood Ratio",
                    "Negative Likelihood Ratio", "Prevalence Threshold", "Threat Score", "Accuracy",
"Balanced Accuracy",
                    "F1  score",  "Matthews  Correlation  Coefficient",  "Fowlkes-Mallows  index",
"Bookmaker Informedness",
                    "Markedness", "Diagnostic Odds Ratio", "Learning_time", "Testing_time"
                ]

                self.GetStatistic()

            def Calculate(self):
                logger.debug("Calculate")
                num_channels = len(self.all_matrix_passivity)
                path                                                                                =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Mathematical          Statistics
Fourier/{self.fourier_type}/CSV'

                matrix = []
                target_matrix = []

                channels_range  =  range(num_channels)  if  'channels_range'  not  in  locals()  and
'channels_range' not in globals() else channels_range

                for channel in range(num_channels):
                    passivity_data = self.all_matrix_passivity[channel]
                    activity_data = self.all_matrix_activity[channel]

                    channel_matrix = [*passivity_data, *activity_data]
                    channel_target_matrix   =   [*[False]   *   len(passivity_data),   *[True]   *
len(activity_data)]

                    matrix.append(channel_matrix)
                    target_matrix.append(channel_target_matrix)
```

```
                res_col = ["Average"]
                res_cm = []

                statistic_p, statistic_a = self.read_channel_data(path, "Mathematical Expectation",
num_channels)
                rmsestart = time.time()
                rmse_p = self.rmse(statistic_p, matrix, num_channels)
                rmse_a = self.rmse(statistic_a, matrix, num_channels)
                compare_rmse = rmse_p > rmse_a
                rmseend = time.time()
                y_true_all = np.concatenate([target_matrix[i] for i in channels_range])
                y_pred_all = np.concatenate([compare_rmse[i] for i in channels_range])
                allend = time.time()
                allttime = (allend-rmsestart)*10**3 + self.ftime
                confusion_matrix = ConfusionMatrix(y_true_all, y_pred_all, self.ftime, allttime)
                print(("%s: %.2f" % ("Accuracy Average",  confusion_matrix.getACC() * 100)))
                res_cm.append(confusion_matrix.getAllVariables())

                path = f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/c_{self.channel_type}/{self.fourier_type}/{self.power}'
                df = pd.read_csv(f'{path}/n-{self.terms}.csv')
                df["SPC"] = np.transpose(np.round(res_cm, 2)) #SPC

                df.to_csv(f'{path}/n-{self.terms}.csv', index=False)


            def rmse(self, statistic, all_matrix, num_channels):
                rmse_results = []
                for channel_idx in range(num_channels):
                    channel_expectation = statistic[channel_idx]
                    channel_signals = all_matrix[channel_idx]
                    channel_rmse = [np.sqrt(np.mean((signal - channel_expectation) ** 2)) for signal
in channel_signals]
                    rmse_results.append(channel_rmse)
                return np.array(rmse_results)



            def read_channel_data(self, base_path, file_pattern, num_channels):
                passivity_file = pd.read_csv(f'{base_path}/{file_pattern} passivity.csv')
                activity_file = pd.read_csv(f'{base_path}/{file_pattern} activity.csv')

                passivity_channel_data = [passivity_file[f'Data_{channel}'] for channel in
range(num_channels)]
                activity_channel_data = [activity_file[f'Data_{channel}'] for channel in
range(num_channels)]

                return np.array(passivity_channel_data), np.array(activity_channel_data)

            def GetStatistic(self):
                logger.debug("Mean Fourier")
```

```
                path                                                                                    =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Mathematical          Statistics
Fourier/{self.fourier_type}/CSV'

                self.process_matrix(self.all_matrix_passivity, "passivity", path)
                self.process_matrix(self.all_matrix_activity, "activity", path)

            def process_matrix(self, matrices, matrix_type, path):
                statistics_matrix = []
                for matrix in zip(matrices):

statistics_matrix.append(MathematicalStatistics(matrix).getMathematicalStatistics())
                self.plot_to_csv(path,  [statistic.getMathematicalExpectation()  for  statistic  in
statistics_matrix], f"Mathematical Expectation {matrix_type}")
```

Файл no_classifires_fourier_all_chanels.py (Реалізація класифікатора SIC)

```
from loguru import logger
import pandas as pd
import numpy as np
import neurokit2 as nk
from pathlib import Path
import matplotlib.pyplot as plt
from scipy.integrate import simps
import time

from classification_metrics.confusion_matrix import ConfusionMatrix

class NoClassidireFourierAllChanels():

    def __init__(self, eeg_config, data, fourier_type, terms, power):

        self.power = power
        self.terms = terms

        self.channel_type = "1"

        self.eeg_config = eeg_config
        logger.debug("No Classifires Fourier")
        self.sampling_rate = data.getModSamplingRate()
        data_matrix_passivity_, data_matrix_activity_ = data.getPreparedData()
        data_matrix_activity = np.power(data_matrix_activity_, power)
        data_matrix_passivity = np.power(data_matrix_passivity_, power)
        self.n_sigma = eeg_config.getSigma()[1]

        fstart = time.time()
        self.all_matrix_passivity      =      [[self.getFourierSeries(m,      fourier_type,
terms=self.terms) for m in data_matrix_passivity[0]]]
        self.all_matrix_activity = [[self.getFourierSeries(m, fourier_type, terms=self.terms)
for m in data_matrix_activity[0]]]
        fend = time.time()
        self.ftime = (fend-fstart)*10**3
```

```
                self.confusion_matrix_names = [
                    "True Positive Rate",
                    "True Negative Rate", "Positive Predictive Value", "Negative Predictive Value",
"False Negative Rate",
                    "False Positive Rate", "False Discovery Rate", "False Omission Rate", "Positive
Likelihood Ratio",
                    "Negative Likelihood Ratio", "Prevalence Threshold", "Threat Score", "Accuracy",
"Balanced Accuracy",
                    "F1 score", "Matthews Correlation Coefficient", "Fowlkes-Mallows index",
"Bookmaker Informedness",
                    "Markedness", "Diagnostic Odds Ratio", "Learning_time", "Testing_time"
                ]
                self.fourier_type = fourier_type
                self.ltime = []
                self.NoAllSigma()

            def NoTest(self):
                logger.debug("No Test Sigma")

                num_channels = len(self.all_matrix_passivity)
                path = f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/All Mean
Fourier/{self.fourier_type}/CSV'

                p_sigmas = self.read_channel_data(path, "Passivity Sigma", num_channels)
                a_sigmas = self.read_channel_data(path, "Activity Sigma", num_channels)
                p_means = self.read_channel_data(path, "Passivity Mathematical Expectation",
num_channels)
                a_means = self.read_channel_data(path, "Activity Mathematical Expectation",
num_channels)

                # channels_range = [0, 1]

                res_all = []
                matrix = []
                cm_all = []
                cm_col = []
                target_matrix = []
                average_relative_overlaps = []
                channels_range = range(num_channels) if 'channels_range' not in locals() and
'channels_range' not in globals() else channels_range

                for channel in range(num_channels):
                    passivity_data = self.all_matrix_passivity[channel]
                    activity_data = self.all_matrix_activity[channel]

                    channel_matrix = [*passivity_data, *activity_data]
                    channel_target_matrix = [*[False] * len(passivity_data), *[True] *
len(activity_data)]

                    matrix.append(channel_matrix)
                    target_matrix.append(channel_target_matrix)
```

```
                n_sigma = self.n_sigma

                lstart = time.time()
                p_upper_bounds = p_means + (n_sigma * p_sigmas)
                p_lower_bounds = p_means - (n_sigma * p_sigmas)
                a_upper_bounds = a_means + (n_sigma * a_sigmas)
                a_lower_bounds = a_means - (n_sigma * a_sigmas)
                p_lower_bounds[p_lower_bounds < 0] = 0
                a_lower_bounds[a_lower_bounds < 0] = 0
                lend = time.time()
                ltime = np.sum(np.array(self.ltime))  + ((lend-lstart)*10**3) + self.ftime

                p_res_by_channel_all = []
                a_res_by_channel_all = []

                tstart = time.time()
                for channel in range(num_channels):
                    p_res_by_channel = []
                    a_res_by_channel = []
                    res_by_channel = []
                    for mean in matrix[channel]:

                        p_within_bounds   =   (mean   >=   p_lower_bounds[channel])   &   (mean   <=
p_upper_bounds[channel])
                        a_within_bounds   =   (mean   >=   a_lower_bounds[channel])   &   (mean   <=
a_upper_bounds[channel])

                        res_by_channel.append(not(np.sum(p_within_bounds) > np.sum(a_within_bounds)))
                        p_res_by_channel.append(np.sum(p_within_bounds))
                        a_res_by_channel.append(np.sum(a_within_bounds))

                    p_res_by_channel_all.append(p_res_by_channel)
                    a_res_by_channel_all.append(a_res_by_channel)
                    res_all.append(res_by_channel)
                tend = time.time()

                p_res_sum_all = np.sum([p_res_by_channel_all[i] for i in channels_range], axis=0)
                a_res_sum_all = np.sum([a_res_by_channel_all[i] for i in channels_range], axis=0)

                y_pred_all = np.logical_not(p_res_sum_all > a_res_sum_all)
                as_tend = time.time()
                ttime = ((as_tend-tstart)*10**3) + self.ftime

                confusion_matrix = ConfusionMatrix(target_matrix[0], y_pred_all, ltime, ttime)
                cm_all.append(confusion_matrix.getAllVariables())
                cm_col.append('Accuracy Sum')
                print(("%s: %.2f" % ("Accuracy Sum",  confusion_matrix.getACC() * 100)))

                path = f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/c_{self.channel_type}/{self.fourier_type}/{self.power}'
                df = pd.read_csv(f'{path}/n-{self.terms}.csv')
                df["SIC"] = np.transpose(np.round(cm_all, 2)) #SPC
```

```
                df.to_csv(f'{path}/n-{self.terms}.csv', index=False)


            def NoAllSigma(self):
                logger.debug("No All Mean Fourier")
                xtext = "$n$"
                path = f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/All Mean
Fourier/{self.fourier_type}'

                self.process_matrix(self.all_matrix_passivity, "Passivity", path, xtext)
                self.process_matrix(self.all_matrix_activity, "Activity", path, xtext)

            def process_matrix(self, matrices, matrix_type, path, xtext):
                for index, matrix in enumerate(matrices):
                    lstart = time.time()
                    data_matrix = np.transpose(matrix)
                    all_mean_data = np.mean(data_matrix, axis=1)
                    lend = time.time()
                    self.ltime.append((lend-lstart)*10**3)
                    title = f"{matrix_type} Mathematical Expectation {index}"
                    #       self.plot_to_png(path,       all_mean_data,       title,       xtext=xtext,
ytext=self.fourier_type)
                    self.plot_to_csv(path, all_mean_data, title)
                    all_sigma_data = np.std(data_matrix, axis=1, ddof=1)
                    title = f"{matrix_type} Sigma {index}"
                    #       self.plot_to_png(path,       all_sigma_data,       title,       xtext=xtext,
ytext=self.fourier_type)
                    self.plot_to_csv(path, all_sigma_data, title)

            def read_channel_data(self, base_path, file_pattern, num_channels):
                channel_data = []
                for channel in range(num_channels):
                    file_name = f'{file_pattern} {channel}.csv'
                    full_path = f'{base_path}/{file_name}'
                    channel_data.append(pd.read_csv(full_path)["Data"])
                return np.array(channel_data)

            def plot_to_png(self, path, plot, name, xlim = None, ylim = None, ytext="", xtext="",
size=(12, 6)):
                logger.info(f"Plot {name}.png")
                Path(path).mkdir(parents=True, exist_ok=True)
                plt.clf()
                plt.rcParams.update({'font.size': 14})
                f, axis = plt.subplots(1)
                f.tight_layout()
                f.set_size_inches(size[0], size[1])
                axis.grid(True)
                time = np.arange(0, len(plot), 1)
                # axis.plot(time, plot, 'o-', markersize=10)
                _, stemlines, _ = axis.stem(time, plot)
                axis.set_xlabel(xtext, loc = 'right')
                axis.set_title(ytext, loc = 'left', fontsize=14, position=(-0.06, 0))
                if xlim is not None:
```

```
            axis.axis(xmin = xlim[0], xmax = xlim[1])
        if ylim is not None:
            axis.axis(ymin = ylim[0], ymax = ylim[1])
        plt.savefig(f'{path}/{name}.png', dpi=300)

    def plot_to_csv(self, path, plot, name):
        logger.info(f"Save {name}.csv")
        path = f'{path}/CSV'
        Path(path).mkdir(parents=True, exist_ok=True)
        time = np.arange(0, len(plot), 1) / self.sampling_rate
        data = pd.DataFrame({"Time" : time, "Data" : plot})
        nk.write_csv(data, f'{path}/{name}.csv')
```

Файл diff_confusion_matrics.py (Розраховуємо середні значення для користувачів Група 1 та Група 2)

```
from loguru import logger
import pandas as pd
import numpy as np
from pathlib import Path
import diff.operators as used_operators
import matplotlib.pyplot as plt
import time as sys_time
from itertools import combinations

class DiffConfusionMatrix():

    def __init__(self, eeg_config, data, fourier_type, terms, power, channel_type):
        self.eeg_config = eeg_config
        self.power = power
        self.terms = terms
        self.fourier_type = fourier_type
        self.channel_type = channel_type
        logger.debug("Diff Confusion Matrix")

        self.names = [
            "Nearest Neighbors",
            "Linear SVM",
            "Decision Tree",
            "Random Forest",
            "Neural Net (MLP)",
            "AdaBoost",
            "Gaussian Naive Bayes",
            "SIC",
            "SPC"
        ]

        self.Onames = [
            "Unnamed: 0",
            "k-Nearest Neighbors",
            "Linear SVM",
```

```
                "Decision Tree",
                "Random Forest",
                "Multilayer Perceptron",
                "Adaptive Boosting",
                "Naive Bayes",
                "SIC",
                "SPC"
            ]

            self.names2 = [
                "Nearest Neighbors",
                "Linear SVM",
                "Decision Tree",
                "Random Forest",
                "Neural Net (MLP)",
                "AdaBoost",
                "Gaussian Naive Bayes",
                "SIC",
                "SPC"
            ]

            self.confusion_matrix_names = [
                "True Positive Rate",
                "True Negative Rate", "Positive Predictive Value", "Negative Predictive Value",
"False Negative Rate",
                "False Positive Rate", "False Discovery Rate", "False Omission Rate", "Positive
Likelihood Ratio",
                "Negative Likelihood Ratio", "Prevalence Threshold", "Threat Score", "Accuracy",
"Balanced Accuracy",
                "F1  score",  "Matthews  Correlation  Coefficient",  "Fowlkes-Mallows  index",
"Bookmaker Informedness",
                "Markedness", "Diagnostic Odds Ratio", "Learning time", "Testing time"
            ]

            self.selected_rows = [
                 "Accuracy", "Balanced Accuracy", "F1 score", "Learning time", "Testing time"
            ]

            self.selected_rows_accuracy = [
                 "Accuracy", "Balanced Accuracy", "F1 score"
            ]

            self.selected_rows_time = [
                 "Learning time", "Testing time"
            ]

            self.selected_rows2 = [
                "Accuracy", "Balanced Accuracy", "F1 score", "Learning time", "Testing time"
            ]

            self.selected_array = used_operators.arrays.get(self.eeg_config.getConfigBlock(), [])

        def DiffGetMax(self):
```

```
                logger.debug("DiffGetMax")
                file_paths = []
                channel_types = ["1", "2", "3", "int"]
                powers = [1, 2, 3, 4]
                fourier_types = ["an_bn"]
                term_counts = [40, 999]
                file_paths = [
                    dict(channel_type = f"c_{channel_type}", fourier_type = f"{fourier_type}", power
= f"{power}", term = f"n-{terms}")
                    for channel_type in channel_types
                    for power in powers
                    for fourier_type in fourier_types
                    for terms in term_counts
                ]

                max_accuracy = 99999999999999
                max_accuracy_file = ""
                # test = "Accuracy"
                # test = "Balanced Accuracy"
                # test = "F1 score"
                # test = "Learning_time"
                test = "Testing_time"

                for i, file_path in enumerate(file_paths):
                    path1                                                                                =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/{file_path["channel_type"]}/{file_path["fourier_type"]}/{file_path["power"]}/{file_path["term
"]}.csv'
                    df = pd.read_csv(path1)
                    accuracy_row = df[df['Unnamed: 0'] == test]
                    if not accuracy_row.empty:
                        # current_max_accuracy = accuracy_row.iloc[:, 1:].max().max()
                        current_max_accuracy = accuracy_row.iloc[:, 1:].min().min()
                        if current_max_accuracy < max_accuracy:
                            max_accuracy = current_max_accuracy
                            max_accuracy_file = file_path
                print(max_accuracy)
                print(max_accuracy_file)



            def PlotDiffFourierTime(self):
                logger.debug("PlotDiffFourierTime")
                logger.debug("DiffAll")
                file_paths = []
                channel_types = ["1"]
                powers = [1]
                # channel_types = ["int"]
                # powers = [1]
                fourier_types = ["an_bn"]
                term_counts = [40, 999]
                file_paths = [
```

```
                    dict(channel_type = f"c_{channel_type}", fourier_type = f"{fourier_type}", power
= f"{power}", term = f"n-{terms}")
                    for channel_type in channel_types
                    for power in powers
                    for fourier_type in fourier_types
                    for terms in term_counts
                    # if (fourier_type != "an_bn" or terms != 999) and (fourier_type != "bn" or terms !=
999)
                ]

                for i, file_path1 in enumerate(file_paths):
                    for file_path2 in file_paths[i+1:]:
                        path1                                                                    =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/{file_path1["channel_type"]}/{file_path1["fourier_type"]}/{file_path1["power"]}/variancen_2-
{file_path1["term"]}.csv'
                        path2                                                                    =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/{file_path2["channel_type"]}/{file_path2["fourier_type"]}/{file_path2["power"]}/variancen_2-
{file_path2["term"]}.csv'
                        res_path                                                                 =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Diff/{file_path1["channel_type"]
}-{file_path2["channel_type"]}/{file_path1["fourier_type"]}-
{file_path2["fourier_type"]}/{file_path1["power"]}-{file_path2["power"]}/{file_path1["term"]}-
{file_path2["term"]}'
                        diff = pd.read_csv(f'{res_path}/diff.csv')
                        v_df1 = pd.read_csv(path1)
                        v_df2 = pd.read_csv(path2)
                        if 'AVG' in diff.columns:
                            diff = diff.drop('AVG', axis=1)
                        if 'AVG' in v_df1.columns:
                            v_df1 = v_df1.drop('AVG', axis=1)
                        if 'AVG' in v_df2.columns:
                            v_df2 = v_df2.drop('AVG', axis=1)
                        v_df1_index = "Time" if file_path1["term"] == "n-999" else "Fourier"
                        v_df2_index = "Time" if file_path2["term"] == "n-999" else "Fourier"
                        print(diff)
                        diff.columns = self.Onames
                        v_df1.columns = self.Onames
                        v_df2.columns = self.Onames
                        #   v_df1_index   =   "Time   1"   if   (file_path1["term"]   ==   "n-999")   and
(file_path2["term"] == "n-999") else "Fourier 1"
                        #   v_df2_index   =   "Time   2"   if   (file_path1["term"]   ==   "n-999")   and
(file_path2["term"] == "n-999") else "Fourier 2"

                        accuracy = diff.loc[diff['Unnamed: 0'].isin(self.selected_rows_accuracy)]
                        time = diff.loc[diff['Unnamed: 0'].isin(self.selected_rows_time)]
                        v_df1_accuracy                  =                  v_df1.loc[v_df1['Unnamed:
0'].isin(self.selected_rows_accuracy)]
                        v_df1_time = v_df1.loc[v_df1['Unnamed: 0'].isin(self.selected_rows_time)]
                        v_df2_accuracy                  =                  v_df2.loc[v_df2['Unnamed:
0'].isin(self.selected_rows_accuracy)]
                        v_df2_time = v_df2.loc[v_df2['Unnamed: 0'].isin(self.selected_rows_time)]
```

```
                    for idx, row in v_df1_accuracy.iterrows():
                        if row['Unnamed: 0'] in self.selected_rows_accuracy:
                            v_df1_accuracy.loc[idx, 'Unnamed: 0'] = f"{v_df1_index} SD
{row['Unnamed: 0']}"
                    for idx, row in v_df1_time.iterrows():
                        if row['Unnamed: 0'] in self.selected_rows_time:
                            v_df1_time.loc[idx, 'Unnamed: 0'] = f"{v_df1_index} SD {row['Unnamed:
0']}"
                    for idx, row in v_df2_accuracy.iterrows():
                        if row['Unnamed: 0'] in self.selected_rows_accuracy:
                            v_df2_accuracy.loc[idx, 'Unnamed: 0'] = f"{v_df2_index} SD
{row['Unnamed: 0']}"
                    for idx, row in v_df2_time.iterrows():
                        if row['Unnamed: 0'] in self.selected_rows_time:
                            v_df2_time.loc[idx, 'Unnamed: 0'] = f"{v_df2_index} SD {row['Unnamed:
0']}"
                    result_accuracy = pd.concat([accuracy, v_df1_accuracy, v_df2_accuracy])
                    result_time = pd.concat([time, v_df1_time, v_df2_time])

                    df_melted = result_accuracy.melt(id_vars='Unnamed: 0', var_name='Method',
value_name='Diff')
                    plt.clf()
                    plt.rcParams.update({'font.size': 14})
                    f, axis = plt.subplots(1)
                    f.set_size_inches(19, 6)
                    axis.grid(True)
                    for metric in result_accuracy['Unnamed: 0']:
                        subset = df_melted[df_melted['Unnamed: 0'] == metric]
                        plt.plot(subset['Method'], subset['Diff'], marker='o', label=metric)
                    plt.xticks(rotation=45)
                    plt.legend(loc='upper right')
                    current_ymax = plt.gca().get_ylim()[1]
                    plt.ylim(top=current_ymax + 0.5)
                    plt.tight_layout()
                    plt.savefig(f'{res_path}/diff_accuracy.png', dpi=300)
                    plt.close(f)

                    df_melted = result_time.melt(id_vars='Unnamed: 0', var_name='Method',
value_name='Diff')
                    plt.clf()
                    plt.rcParams.update({'font.size': 14})
                    f, axis = plt.subplots(1)
                    f.set_size_inches(19, 6)
                    axis.grid(True)
                    for metric in result_time['Unnamed: 0']:
                        subset = df_melted[df_melted['Unnamed: 0'] == metric]
                        plt.plot(subset['Method'], subset['Diff'], marker='o', label=metric)
                    plt.xticks(rotation=45)
                    plt.legend(loc='right')
                    plt.tight_layout()
                    plt.savefig(f'{res_path}/diff_time.png', dpi=300)
                    plt.close(f)
```

```
            def DiffConfusionMatrix(self):
                logger.debug("DiffConfusionMatrix") # Mean
                all_data = []
                for operator in self.selected_array:
                    read_path         =         f'{self.eeg_config.getImgPath()}/{operator}/Confusion
matrix/c_{self.channel_type}/{self.fourier_type}/{self.power}'
                    df = pd.read_csv(f'{read_path}/n-{self.terms}.csv')
                    if 'Unnamed: 0' in df.columns:
                        df = df.drop('Unnamed: 0', axis=1)
                    all_data.append(df)
                res = np.round(np.mean(all_data, axis=0), 4)
                df   =   pd.DataFrame(res,   index=self.confusion_matrix_names,   columns=[*self.names,
"AVG"])
                path  =  f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/c_{self.channel_type}/{self.fourier_type}/{self.power}'
                Path(path).mkdir(parents=True, exist_ok=True)
                df.to_csv(f'{path}/n-{self.terms}.csv')

            def AVGDiffConfusionMatrix(self):
                logger.debug("AVGDiffConfusionMatrix")
                for operator in self.selected_array:
                    read_path         =         f'{self.eeg_config.getImgPath()}/{operator}/Confusion
matrix/c_{self.channel_type}/{self.fourier_type}/{self.power}'
                    df = df2 = pd.read_csv(f'{read_path}/n-{self.terms}.csv')
                    if 'Unnamed: 0' in df.columns:
                        df = df.drop('Unnamed: 0', axis=1)
                    if 'AVG' in df.columns:
                        df = df.drop('AVG', axis=1)
                    df2['AVG'] = np.round(np.mean(df, axis=1), 4)
                    df2.to_csv(f'{read_path}/n-{self.terms}.csv', index=False)

            def DiffAll(self):
                logger.debug("DiffAll")
                file_paths = []
                channel_types = ["1"]
                powers = [1]
                # channel_types = ["1"]
                # powers = [1]
                fourier_types = ["an_bn"]
                term_counts = [999, 40]
                file_paths = [
                    dict(channel_type = f"c_{channel_type}", fourier_type = f"{fourier_type}", power
= f"{power}", term = f"n-{terms}")
                    for channel_type in channel_types
                    for power in powers
                    for fourier_type in fourier_types
                    for terms in term_counts
                    # if (fourier_type != "an_bn" or terms != 999) and (fourier_type != "bn" or terms !=
999)
                ]

                # file_path1 = file_paths[0]
```

```
                #                                       path1                                                                 =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/{file_path1["channel_type"]}/{file_path1["fourier_type"]}/{file_path1["power"]}/{file_path1["
term"]}.csv'
                #                                     res_path                                                                =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Diff/{file_path1["channel_type"]
}-{file_path1["channel_type"]}/{file_path1["fourier_type"]}-
{file_path1["fourier_type"]}/{file_path1["power"]}-{file_path1["power"]}/{file_path1["term"]}-
{file_path1["term"]}'
                for i, file_path1 in enumerate(file_paths):
                    for file_path2 in file_paths[i+1:]:
                        path1                                                                                               =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/{file_path1["channel_type"]}/{file_path1["fourier_type"]}/{file_path1["power"]}/{file_path1["
term"]}.csv'
                        path2                                                                                               =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/{file_path2["channel_type"]}/{file_path2["fourier_type"]}/{file_path2["power"]}/{file_path2["
term"]}.csv'
                        #                                       res_path                                                    =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Diff/{file_path1["channel_type"]
}-{file_path2["channel_type"]}/{file_path1["fourier_type"]}-
{file_path2["fourier_type"]}/{file_path1["power"]}-{file_path2["power"]}/{file_path1["term"]}-
{file_path2["term"]}'
                        res_path                                                                                            =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Diff/{file_path2["channel_type"]
}-{file_path1["channel_type"]}/{file_path2["fourier_type"]}-
{file_path1["fourier_type"]}/{file_path2["power"]}-{file_path1["power"]}/{file_path2["term"]}-
{file_path1["term"]}'
                        Path(res_path).mkdir(parents=True, exist_ok=True)
                        df1 = pd.read_csv(path1)
                        df2 = pd.read_csv(path2)
                        df1 = df1.loc[df1['Unnamed: 0'].isin(self.selected_rows)]
                        df2 = df2.loc[df2['Unnamed: 0'].isin(self.selected_rows)]
                        df1 = df1.drop('Unnamed: 0', axis=1)
                        df2 = df2.drop('Unnamed: 0', axis=1)
                        # df = df1 - df2
                        df = df2 - df1
                        df = df.round(3)
                        df.index = self.selected_rows2
                        df.to_csv(f'{res_path}/diff.csv')

            def DiffVariance(self):
                logger.debug("DiffVariance")
                read_path_m                                                                                                 =
f'{self.eeg_config.getImgPath()}/{self.eeg_config.getConfigBlock()}/Confusion
matrix/c_{self.channel_type}/{self.fourier_type}/{self.power}'
                df_m = pd.read_csv(f'{read_path_m}/n-{self.terms}.csv')
                df_m = df_m.loc[df_m['Unnamed: 0'].isin(self.selected_rows)]
                if 'Unnamed: 0' in df_m.columns:
                    df_m = df_m.drop('Unnamed: 0', axis=1)
                res_m__2 = []
                res_m__ = []
```

```
                for operator in self.selected_array:
                    read_path        =        f'{self.eeg_config.getImgPath()}/{operator}/Confusion
matrix/c_{self.channel_type}/{self.fourier_type}/{self.power}'
                    df = pd.read_csv(f'{read_path}/n-{self.terms}.csv')
                    df = df.loc[df['Unnamed: 0'].isin(self.selected_rows)]
                    if 'Unnamed: 0' in df.columns:
                        df = df.drop('Unnamed: 0', axis=1)
                    df.columns = [*self.names, "AVG"]
                    m__2 = np.power(df - df_m, 2)
                    res_m__2.append(m__2)

                res1 = np.round(np.mean(res_m__2, axis=0), 3)
                res2 = np.round(np.sqrt(np.mean(res_m__2, axis=0)), 3)

                df_res1 = pd.DataFrame(res1, index=self.selected_rows, columns=[*self.names, "AVG"])
                df_res2 = pd.DataFrame(res2, index=self.selected_rows, columns=[*self.names, "AVG"])
                df_res1.to_csv(f'{read_path_m}/variancen-{self.terms}.csv')
                df_res2.to_csv(f'{read_path_m}/variancen_2-n-{self.terms}.csv')
```

Файл plot_classifiers.py (Візуалізація результатів класифікації)

```
from loguru import logger
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path

class PlotClassifiers():

    def __init__(self, ecg_config, data, fourier_type, power):
        logger.info("Plot Classifiers")

        path      =      f'{ecg_config.getImgPath()}/{ecg_config.getConfigBlock()}/Confusion
matrix/c_1/{fourier_type}/{power}'

        confusion_matrix_names = [
            "Accuracy", "Balanced Accuracy", "F1 score"
        ]

        names = [
            "k-Nearest Neighbors",
            "Linear SVM",
            "Decision Tree",
            "Random Forest",
            "Multilayer Perceptron",
            "Adaptive Boosting",
            "Naive Bayes",
            "SIC",
            "SPC"
        ]
```

```
            arr = [3, 4, 5, 7, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90,
95, 100]

            for cm_name in confusion_matrix_names:
                read_data = []
                for i in arr:
                    df = pd.read_csv(f'{path}/n-{i}.csv')
                    accuracy_row = df.loc[df['Unnamed: 0'] == cm_name]
                    if 'AVG' in df.columns:
                        df = df.drop('AVG', axis=1)
                    accuracy_array = accuracy_row.values.flatten()[1:]
                    read_data.append(accuracy_array)

                t2 = np.transpose(np.array(read_data))

                plt.clf()
                plt.rcParams.update({'font.size': 14})
                f, axis = plt.subplots(1)
                f.tight_layout()
                f.set_size_inches(19, 6)
                f.tight_layout()
                axis.grid(True)
                axis.set_xlabel("Number of coefficients", loc = 'right')
                # axis.set_title("$t, ms$", loc = 'left', fontsize=14, position=(-0.06, 0))
                for i, name in zip(t2[::-1], names[::-1]):
                    axis.plot(arr, i, linewidth=2, label=name)
                axis.legend(loc='best',prop={'size':10})

                max_value = np.nanmax(t2) if not np.isnan(np.nanmax(t2)) else 0
                min_value = np.nanmin(t2) if not np.isnan(np.nanmax(t2)) else 0
                ymin = 0
                ymax = 1.1

                if ymin > min_value:
                    ymin = min_value

                if ymax < max_value and not np.isinf(max_value):
                    ymax = max_value

                axis.axis(ymin = ymin, ymax = ymax)

                axis.axis(xmin = 3, xmax = 100)
                Path(f'{path}/img').mkdir(parents=True, exist_ok=True)
                plt.savefig(f'{path}/img/{cm_name}.png', dpi=300)
```

Файл authentication.py (Аутентифікація користувача за одним циклом ЕКГ)

```
import random
from authentication.confusion_matrix import ConfusionMatrix
from get_config.ecg_config import ECGConfig
from loguru import logger
from my_helpers.data_preparation import DataPreparation
```

```
from my_helpers.read_data.read_data_file import ReadDataFile
import numpy as np
import time
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
import pandas as pd
from pathlib import Path
import matplotlib.pyplot as plt
import neurokit2 as nk
import authentication.authentication_data as used_authentication

class Authentication():

    def __init__(self, ecg_config):
        logger.debug("Authentication Init")

        self.ecg_config = ecg_config
        self.ltime = 0
        self.a_path = f'{self.ecg_config.getImgPath()}/{self.ecg_config.getConfigBlock()}'
        Path(self.a_path).mkdir(parents=True, exist_ok=True)

        self.names = [
            "Nearest Neighbors",
            "Linear SVM",
            "Decision Tree",
            "Random Forest",
            "Neural Net (MLP)",
            "AdaBoost",
            "Naive Bayes"
        ]

        self.classifiers = [
            KNeighborsClassifier(3),
            SVC(kernel="linear", C=0.025),
            GaussianProcessClassifier(1.0 * RBF(1.0)),
            DecisionTreeClassifier(max_depth=5),
            RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
            MLPClassifier(alpha=1, max_iter=1000),
            AdaBoostClassifier(),
            GaussianNB(),
        ]
```

```
                self.confusion_matrix_names = [
                    "True Positive Rate",
                    "True Negative Rate", "False Negative Rate", "False Positive Rate", "Accuracy",
"Balanced Accuracy",
                    "F1 score", "Learning_time", "Testing_time"
                ]

            def Diff(self):
                logger.debug("Diff")
                # tmp_path = "Confusion matrix 0"
                # tmp_path = "Confusion matrix line 0"
                # tmp_path = "Confusion matrix line SCG"
                tmp_path = "Confusion matrix SCG"
                for average_elements in np.arange(1, 20 + 1):
                    all_data                                                                    =
[pd.read_csv(f'{self.ecg_config.getImgPath()}/{operator}/{tmp_path}/Authentication                n-
{average_elements}.csv').drop('Unnamed:    0',    axis=1,    errors='ignore')    for    operator    in
used_authentication.diff]
                    df       =       pd.DataFrame(np.round(np.mean(all_data,       axis=0),       3),
index=self.confusion_matrix_names, columns=[*self.names, "SIC"])
                    Path(f'{self.a_path}/{tmp_path}').mkdir(parents=True, exist_ok=True)
                    df.to_csv(f'{self.a_path}/{tmp_path}/Authentication n-{average_elements}.csv')

                #                                       all_data                                    =
[pd.read_csv(f'{self.ecg_config.getImgPath()}/{operator}/{tmp_path}/Sigma  mean.csv').drop('Unnamed:
0', axis=1, errors='ignore') for operator in used_authentication.diff]
                # df = pd.DataFrame(np.round(np.mean(all_data, axis=0), 4), columns=["N", "Data"])
                # df.to_csv(f'{self.a_path}/{tmp_path}/Sigma mean.csv')

            def Classifiers(self):
                logger.debug("Classifiers")
                selected_array = used_authentication.arrays.get(self.ecg_config.getConfigBlock(), [])
                # other_array = used_authentication.arrays.get("OTHER1", [])
                other_array            =            self.form_res_array(used_authentication.arrays,
self.ecg_config.getConfigBlock())

                all_data_in = []
                other_data_in = []

                DataPreparation(ECGConfig(selected_array[0])).getPreparedData()

                all_data_in.extend([item    for    conf    in    selected_array    for    item    in
DataPreparation(ECGConfig(conf)).getPreparedData()])
                for conf in other_array:

other_data_in.append(DataPreparation(ECGConfig(conf)).getPreparedData()[:int(len(all_data_in)      /
10)])


                sigma_mean = []

                for average_elements in np.arange(1, 20 + 1):
```

```
                    self.ltime = 0

                    other_data = []

                    all_data = self.average_elements_np(all_data_in, average_elements)

                    for item in other_data_in:
                        other_data.extend(self.average_elements_np(item, average_elements))

                    target_data = [1] * len(all_data)
                    other_target_data = [2] * len(other_data)

                    all_data_ = [*all_data, *other_data]
                    target_data_ = [*target_data, *other_target_data]

                    data_train,     data_test,     target_values_train,     target_values_test     =
train_test_split(all_data_, target_data_, test_size=0.3, random_state=42)

                    y_true = np.array(target_values_test)

                    # filtered_data = [data for data, target in zip(data_train, target_values_train)
if target == 1]
                    #   filtered_other_data   =   [data   for   data,   target   in   zip(data_train,
target_values_train) if target == 2]
                    self.NoAllSigma(self.a_path, all_data)

                    a_sigmas = self.read_channel_data(self.a_path, "All Sigma")
                    a_means = self.read_channel_data(self.a_path, "All Mathematical Expectation")

                    sigma_mean.append(np.mean(a_sigmas))

                    n_sigma = 2.6  #2.6

                    lstart = time.time()
                    a_upper_bounds = np.power(a_means + (n_sigma * a_sigmas), 1)
                    a_lower_bounds = np.power(a_means - (n_sigma * a_sigmas), 1)



                    # # mean = all_data[10]
                    # mean = other_data[10]

                    # plt.clf()
                    # plt.rcParams.update({'font.size': 15})
                    # f, axis = plt.subplots(1)
                    # f.set_size_inches(12, 6)
                    # f.tight_layout()
                    # axis.grid(True)
                    # m_time = np.arange(0, len(mean), 1) / 360
                    # axis.plot(m_time, mean, linewidth=3, label=r"$\xi_{{\omega}} (t), mV$")
                    # axis.plot(m_time, a_upper_bounds, linewidth=3, label=r"$Upper_{{\xi}} (t), mV$")
                    # axis.plot(m_time, a_lower_bounds, linewidth=3, label=r"$Lower_{{\xi}} (t), mV$")
                    # axis.set_xlabel("$t, s$", loc = 'right')
                    # axis.axis(xmin = 0, xmax = 1)
```

```
                    # axis.legend(loc='upper right')
                    # plt.savefig(f'{self.a_path}/Other-Authentication.png', dpi=300)
                    # # plt.savefig(f'{self.a_path}/Authentication.png', dpi=300)


                    lend = time.time()
                    ltime = self.ltime + (lend-lstart)*10**3

                    y_pred = []

                    tstart = time.time()
                    y_pred = [1 if (np.mean((mean >= a_lower_bounds) & (mean <= a_upper_bounds)) *
100) >= 93.0 else 2 for mean in data_test] #82---86
                    tend = time.time()
                    ttime = (tend-tstart)*10**3

                    confusion_matrix = ConfusionMatrix(y_true, y_pred, ltime, ttime)
                    sic_res = confusion_matrix.getAllVariables()
                    print(("%s: %.2f" % ("SIC", confusion_matrix.getACC() * 100)))

                    _cm = []
                    for name, clf in zip(self.names, self.classifiers):
                        clf = make_pipeline(StandardScaler(), clf)
                        lstart = time.time()
                        clf.fit(data_train, target_values_train)
                        lend = tstart = time.time()
                        y_true = np.array(target_values_test)
                        y_pred = clf.predict(data_test)
                        tend = time.time()
                        ltime = (lend-lstart)*10**3
                        ttime = (tend-tstart)*10**3
                        confusion_matrix = ConfusionMatrix(y_true, y_pred, ltime, ttime)
                        res = confusion_matrix.getAllVariables()
                        print(("%s: %.2f" % (name, confusion_matrix.getACC() * 100)))
                        _cm.append(res)

                    _cm.append(sic_res)

                    path = f'{self.a_path}/Confusion matrix'
                    # path = f'{self.a_path}/Confusion matrix line 0'
                    # path = f'{self.a_path}/Confusion matrix line SCG'
                    # path = f'{self.a_path}/Confusion matrix SCG'
                    Path(path).mkdir(parents=True, exist_ok=True)

                    df           =           pd.DataFrame(np.transpose(np.round(_cm,           4)),
index=self.confusion_matrix_names, columns=[*self.names, "SIC"])
                    df.to_csv(f'{path}/Authentication n-{average_elements}.csv')

                data = pd.DataFrame({"N" : np.arange(1, average_elements + 1), "Data" : sigma_mean})
                nk.write_csv(data, f'{path}/Sigma mean.csv')

            def Plot_n(self):
                logger.debug("Plot_n")
```

```
confusion_matrix_names = [
    # "Accuracy",
    "Accuracy", "Balanced Accuracy",
    "F1 score", "Learning_time", "Testing_time"
]
names = [*self.names, "SIC"]
# path = f'{self.a_path}/Confusion matrix'
# path = f'{self.a_path}/Confusion matrix line'
# path = f'{self.a_path}/Confusion matrix line SCG'
path = f'{self.a_path}/Confusion matrix SCG'
arr = np.arange(1, 20 + 1)
for cm_name in confusion_matrix_names:
    read_data = []
    for i in arr:
        df = pd.read_csv(f'{path}/Authentication n-{i}.csv')
        accuracy_row = df.loc[df['Unnamed: 0'] == cm_name]
        accuracy_array = accuracy_row.values.flatten()[1:]
        read_data.append(accuracy_array)
    t2 = np.transpose(np.array(read_data))
    plt.clf()
    plt.rcParams.update({'font.size': 14})
    f, axis = plt.subplots(1)
    f.set_size_inches(19, 6)
    f.tight_layout()
    axis.grid(True)
    axis.set_xlabel("N", loc = 'right')
    if "time" in cm_name:
        t2 = t2 / 1000.0
        axis.set_title("t, s", loc = 'left', fontsize=14, position=(-0.02, 0))
    for i, name in zip(t2[::-1], names[::-1]):
        axis.plot(arr, i, linewidth=2, label=name, marker='o')
    axis.legend(loc='best',prop={'size':10})

    max_value = np.nanmax(t2) if not np.isnan(np.nanmax(t2)) else 0
    min_value = np.nanmin(t2) if not np.isnan(np.nanmax(t2)) else 0
    ymin = 0
    ymax = 1.1
    if "time" in cm_name:
        ymax = 0.1

    if ymin > min_value:
        ymin = min_value

    if ymax < max_value and not np.isinf(max_value):
        ymax = max_value + (max_value / 11.0)

    axis.axis(ymin = ymin, ymax = ymax)
    axis.axis(xmin = 1, xmax = 20)
    plt.savefig(f'{path}/{cm_name}.png', dpi=300)

# df = pd.read_csv(f'{path}/Sigma mean.csv')
# # df2 = pd.read_csv(f'{path} line/Sigma mean.csv')
```

```
    # # df['Data'] /= df2['Data']

    # plt.clf()
    # plt.rcParams.update({'font.size': 14})
    # f, axis = plt.subplots(1)
    # f.tight_layout()
    # f.set_size_inches(12, 6)
    # axis.grid(True)
    # axis.plot(df['N'], df['Data'], marker='o')
    # axis.set_xlabel("N", loc = 'right')
    # axis.axis(xmin = 0.9, xmax = 20.1)
    # # axis.legend(loc='upper right')
    # plt.savefig(f'{path}/Sigma mean.png', dpi=300)


def average_elements_np(self, arr, i):
    if i == 1:
        return arr
    arr = np.array(arr)
    n_segments = arr.shape[0] // i
    averaged = np.mean(arr[:n_segments*i].reshape(-1, i, arr.shape[1]), axis=1)
    return averaged

def read_channel_data(self, base_path, file_pattern):
    file_name = f'{file_pattern}.csv'
    full_path = f'{base_path}/All Mean/CSV/{file_name}'
    data = pd.read_csv(full_path)["Data"]
    return np.array(data)

def NoAllSigma(self, a_path, all_matrix):
    logger.debug("No All Sigma")
    path = f'{a_path}/All Mean'
    self.process_matrix(all_matrix, "All", path)

def form_res_array(self, arrays, t):
    res = []
    i = 0
    for key, value in arrays.items():
        if i == 11:
            break
        if key != t:
            res.append(value[0])
        i = i + 1
    return res

def process_matrix(self, matrix, matrix_type, path):
    lstart = time.time()
    data_matrix = np.transpose(matrix)
    all_mean_data = np.mean(data_matrix, axis=1)
    all_sigma_data = np.std(data_matrix, axis=1, ddof=1)
    lend = time.time()
    self.ltime = (lend-lstart)*10**3
    title = f"{matrix_type} Mathematical Expectation"
```

```
                self.plot_to_csv(path, all_mean_data, title)
                # self.plot_to_png(path, all_mean_data, title)
                title = f"{matrix_type} Sigma"
                self.plot_to_csv(path, all_sigma_data, title)
                # self.plot_to_png(path, all_sigma_data, title)

            def plot_to_csv(self, path, plot, name):
                logger.info(f"Save {name}.csv")
                path = f'{path}/CSV'
                Path(path).mkdir(parents=True, exist_ok=True)
                time = np.arange(0, len(plot), 1) / 360
                data = pd.DataFrame({"Time" : time, "Data" : plot})
                nk.write_csv(data, f'{path}/{name}.csv')

            def plot_to_png(self, path, plot, name, xlim = None, ylim = None, ytext="", xtext="",
size=(12, 6)):
                logger.info(f"Plot {name}.png")
                Path(path).mkdir(parents=True, exist_ok=True)
                plt.clf()
                plt.rcParams.update({'font.size': 14})
                f, axis = plt.subplots(1)
                f.tight_layout()
                f.set_size_inches(size[0], size[1])
                axis.grid(True)
                time = np.arange(0, len(plot), 1) / 360
                axis.plot(time, plot, linewidth=3)
                axis.set_xlabel(xtext, loc = 'right')
                axis.set_title(ytext, loc = 'left', fontsize=14, position=(-0.06, 0))
                if xlim is not None:
                    axis.axis(xmin = xlim[0], xmax = xlim[1])
                if ylim is not None:
                    axis.axis(ymin = ylim[0], ymax = ylim[1])
                plt.savefig(f'{path}/{name}.png', dpi=300)
```