МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, програмної інженерії та комп'ютерних наук
Кафедра інформаційних технологій

Пояснювальна записка

до кваліфікаційної роботи
другого (магістерського) рівня

на тему Програмна реалізація відновлення трафіку
за допомогою ряду Котельникова

Виконав: студент 2 курсу, групи ІПЗ
спеціальності
121 Інженерія програмного забезпечення

Климчук Олексій Сергійович

Керівник Стрелковська І.В.

Рецензент І.М. Соловська

Одеса – 2023 рік

ДОВІДКА

кафедри інформаційних технологій про виконану магістерську роботу

студента 2 курсу, ФКПІ та КН групи ІПЗ

Кешмука Олексія Сергійовича
(прізвище, ім'я та по-батькові)

на тему: Програмна реалізація відновлення трафіку за допомогою ряду Котельникова.

Висновок нормоконтролера

Пояснювальна записка до кваліфікаційної роботи виконана з незначними порушеннями ДСТУ, робота виконана згідно вимог Внутрішнього положення НПУ.

Нормоконтролер викл. каф. ІТ ______ 15.12.2023 Клімішина І.В.
(науковий ступінь, вчене звання) (підпис, дата) (і. б. прізвище)

Висновок відповідального за перевірку на наявність академічного плагіату згідно сертифікату ID 1015677895 унікальність роботи підтверджено

Відповідальна особа викл. каф. ІТ ______ 15.12.2023 Клімішина І.В.
(науковий ступінь, вчене звання) (підпис, дата) (і. б. прізвище)

Попередній захист магістерської роботи

студ. Кешмука О.С. проведено «12» 12 2023 р.
(прізвище і.б.)

Висновки Магістерська робота виконана в повному обсязі. Студентом розроблено програмний модуль, який має певний потенціал для практичного використання. Кваліфікаційна робота відповідає вимогам до кваліфікаційних робіт та рекомендується до захисту.

Члени комісії ______ д.т.н, проф. Ярмоловська Л.В.
(підпис) (науковий ступінь, вчене звання, прізвище і.б.)

______ к.т.н, доц. Григор'єва Ж.Ф.
(підпис) (науковий ступінь, вчене звання, прізвище і.б.)

______ к.т.н, доц. Горбатов В.Е.
(підпис) (науковий ступінь, вчене звання, прізвище і.б.)

МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, програмної інженерії та комп'ютерних наук
Кафедра інформаційних технологій
Освітній рівень другий (магістерський)
Галузь знань 12 Інформаційні технології
Спеціальність 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
інформаційних технологій
_____ Т.І. Григор'єва
«25» 09 2023 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ

1. Тема роботи: Програмна реалізація відновлення трафіку за допомогою ряду Котельникова

Керівник роботи Стрелковська Ірина Вікторівна, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом закладу вищої освіти від «25» 09 2023 року № 1967

2. Строк подання здобувачем роботи 12.12.2023 р.

3. Вихідні дані до роботи

1. Виконати порівняльний аналіз із методами апроксимації на основі сплайнів
2. Враховувати можливість використання програмного модуля із іншим програмним забезпеченням
3. Забезпечити можливість зміни налаштувань програмного модуля

4. Зміст пояснювальної записки

1. Аналіз методів відновлення сигналів в інфокомунікаційних та комп'ютерних системах
2. Вибір способу програмної реалізації модуля відновлення трафіку
3. Програмна реалізація відновлення трафіку за допомогою ряду Котельникова

5. Перелік демонстраційних креслень: Презентація (8 – 15 слайдів).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз методів відновлення сигналів в інфокомунікаційних та комп'ютерних системах	2.10.2023 – 8.10.2023	вик
2	Вибір способу програмної реалізації модуля відновлення трафіку	09.10.2023 – 15.10.2023	вик
3	Програмна реалізація відновлення трафіку за допомогою ряду Котельникова	13.11.2023 – 19.11.2023	вик

Здобувач _____ О.С. Климчук

Керівник роботи _____ І.В. Стрелковська

ВІДГУК КЕРІВНИКА

на кваліфікаційну роботу другого (магістерського) рівня здобувача
Климчука Олексія Сергійовича
на тему: «Програмна реалізація відновлення трафіку
за допомогою ряду Котельникова»

Теорема Котельника відноситься до категорії фундаментальних теорем, що лежать в основі функціонування інфокомунікаційних систем. Однак незважаючи на наявність більш розвинених методів відновлення трафіку, наприклад, на основі кубічних сплайнів, відновлення трафіку за допомогою теореми Котельникова ще й досі використовується в сучасних інфокомунікаційних системах, оскільки цей метод відрізняється простотою та потребує меншої обчислювальної потужності. Таким чином, програмний модуль, розроблений Климчуком О.С. має певний потенціал для практичного використання.

Здобувач Климчук О.С. самостійно виконав теоретичну та практичну частини завдання до кваліфікаційної роботи. Графік консультацій не порушувався. Поставлене завдання виконано у повному обсязі. Пояснювальна записка та демонстраційні аркуші виконано охайно із дотриманням усіх необхідних вимог.

Під час виконання кваліфікаційної роботи здобувач Климчук О.С. розібрався з усіма поставленими питаннями та показав уміння користуватись технічною літературою, ставити та розв'язувати дослідницькі задачі.

Кваліфікаційна робота відповідає вимогам до кваліфікаційних робіт другого (магістерського) рівня та заслуговує оцінки «відмінно».

Студент Климчук О.С. заслуговує присвоєння кваліфікації магістр з інженерії програмного забезпечення за заявленою спеціальністю 121 «Інженерія програмного забезпечення».

Керівник
Декан факультету кібербезпеки
програмної інженерії
та комп'ютерних наук
д.т.н., професор І.В. Стрелковська

МІЖНАРОДНИЙ ГУМАНІТАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ, ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА КОМП'ЮТЕРНИХ НАУК

РЕЦЕНЗІЯ

на кваліфікаційну роботу другого (магістерського) рівня здобувача
Климчука Олексія Сергійовича
на тему: «Програмна реалізація відновлення трафіку
за допомогою ряду Котельникова»

Кваліфікаційна робота здобувача Климчука О.С. присвячена створенню засобів, що дозволяють аналізувати та досліджувати процеси відновлення трафіку в інфокомунікаційних та комп'ютерних системах, що використовують цифрові методи передачі даних. Розроблений програмний модуль наочно показує процеси, що відбуваються при відновленні сигналів при різних налаштуваннях, що дозволяє більш глибоко зрозуміти принципи роботи інфокомунікаційних систем. Це робить роботу Климчука О.С. актуальною і цінною, як з теоретичної, так і з практичної точок зору.

Здобувач Климчук О.С. продемонстрував високу теоретичну підготовку, добре володіння математичним апаратом та технологіями програмування. Кваліфікаційна робота відповідає завданню, в роботі використані усі вихідні дані. Текст роботи послідовний та зрозумілий, оформлення пояснювальної записки та демонстраційних аркушів якісне.

До недоліків роботи слід віднести:

– не виконано оцінку точності відновлення сигналу;

– недостатню увагу приділено можливостям інтеграції розробленого програмного модуля із реальними джерелами сигналів.

Проте, зазначені недоліки не знижують цінності виконаної роботи.

У цілому кваліфікаційна робота Климчук О.С відповідає вимогам до випускних кваліфікаційних робіт здобувачів другого (магістерського) рівня та заслуговує оцінки «відмінно».

Студент Климчук О.С. заслуговує присвоєння кваліфікації магістр з інженерії програмного забезпечення за заявленою спеціальністю 121 «Інженерія програмного забезпечення».

Рецензент
Завідувач кафедри комп'ютерних наук
к.т.н., доцент І.М. Соловська

UNICHECK
by Turnitin

Ім'я користувача:
Анна Серединко

Дата перевірки:
11.12.2023 23:55:06 MSK

Дата звіту:
12.12.2023 00:01:10 MSK

ID перевірки:
1015995244

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100001433

Назва документа: Климчук диплом

Кількість сторінок: 69 Кількість слів: 10435 Кількість символів: 85362 Розмір файлу: 1.67 MB ID файлу: 1015677895

26.7%
Схожість

Найбільша схожість: 14.9% з джерелом з Бібліотеки (ID файлу: 1015677890)

15.1% Джерела з Інтернету | 814 ..... Сторінка 71

16% Джерела з Бібліотеки | 34 ..... Сторінка 76

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи | 53

РЕФЕРАТ

Текстова частина магістерської роботи містить 51 с., 27 рис., 6 табл., 30 джерел, 3 додатки.

КЛЮЧОВІ СЛОВА: ВІДНОВЛЕННЯ ТРАФІКУ, ТЕОРЕМА КОТЕЛЬНИКОВА, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ДИНАМІЧНО-ЗВ'ЯЗАНА БІБЛІОТЕКА, ВІРТУАЛЬНА ЛАБОРАТОРІЯ, АПРОКСИМАЦІЯ, ПЕРЕДАЧА ІНФОРМАЦІЇ

Об'єкт дослідження – процеси відновлення трафіку в інфокомунікаційних системах.

Предмет дослідження – математичні моделі і програмне забезпечення для відновлення трафіку за допомогою ряду Котельникова.

Мета роботи – розробка програмного модуля для відновлення трафіку за допомогою ряду Котельникова.

Методи дослідження – методи теорії зв'язку.

У роботі виконано аналіз методів відновлення інформаційного трафіку, обґрунтовано використання для вирішення цієї задачі ряду Котельникова та проведено аналіз методів практичної реалізації математичних моделей. Створено програмний модуль, призначений для досліджень методів відновлення трафіку за допомогою ряду Котельникова. Проведено тестування його працездатності, яке підтвердило усі теоретичні розрахунки.

ABSTRACT

The text part of the master's thesis contains 51 pages, 27 figures, 6 tables, 30 sources, 3 appendices.

KEY WORDS: TRAFFIC RECOVERY, KOTELNIKOV'S THEOREM, SOFTWARE, DYNAMIC-LINKED LIBRARY, VIRTUAL LABORATORY, APPROXIMATION, INFORMATION TRANSFER

The object of research is the processes of traffic recovery in information communication systems.

The subject of the research is mathematical models and software for traffic recovery using the Kotelnikov's series.

The purpose of the work is to develop a software module for traffic recovery using the Kotelnikov's series.

The research methods are the communication theory methods.

In the paper, the analysis of information traffic recovery methods was performed, the use of the Kotelnikov's series to solve this problem was substantiated, and the methods of practical implementation of mathematical models were analyzed. A software module designed for research on traffic recovery methods using the Kotelnikov's series has been created. Testing of its performance was conducted, which confirmed all theoretical calculations.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

DLL	(Dynamic Link Library) – динамічно-зв'язана бібліотека
IDE	(Integrated Development Environment) – інтегроване середовище розробки
QoE	(Quality of Experience) – рівень задоволеності клієнтів
QoS	(Quality of Service) – якість сервісу
СКВ	Середньоквадратичне відхилення

ВСТУП

Постійний розвиток інфокомунікаційних систем призводить до стрімкої зміни технологій, періодичного оновлення протоколів та регулярного підвищення вимог до рівня якості послуг, що надаються (Quality of Service, QoS), та до рівня задоволеності клієнтів, що користуються цими послугами (Quality of Experience, QoE) [1]. Тому необхідно постійно шукати нові шляхи підвищення якості передавання інформації [2 – 4] та періодично переглядати і оновлювати відомі і перевіренні рішення.

Одними з основних завдань, які вирішуються в сучасних інфокомунікаційних системах та мережах, є аналіз та синтез інформаційних сигналів, яким притаманні як повна детермінованість, так і раптові непередбачувані «сплески» і швидкі осциляції. Відтворити сигнал з високою точністю на приймальній стороні можна за допомогою сучасних методів апроксимації, наприклад, сплайн-апроксимації (лінійної, квадратичної чи кубічної) [5 – 7], або за допомогою вейвлет-функцій, наприклад, на основі вейвлету Котельникова-Шенона [8 – 10].

Використання вейвлет- та сплайн-функцій дозволяє досягти більшої точності відтворення сигналу, проте більш прості методи відновлення сигналу на основі ряду Котельникова ще не втратили своєї актуальності і продовжують використовуватись в інфокомунікаційних системах. Ключовими перевагами відновлення сигналу за допомогою ряду Котельникова є простота, та менша обчислювальна складність, що дозволяє реалізовувати їх на апаратній основі з меншою продуктивністю і, відповідно, з меншою вартістю.

Однак, незважаючи на широку розповсюдженість методу відновлення сигналу за допомогою ряду Котельникова, кількість відомих програмних модулів, що підтримують швидке інтегрування в існуюче програмне забезпечення, наразі обмежена. Це і обумовило мету цієї роботи, яка полягає у створенні програмного модуля для відновлення трафіку за допомогою ряду Котельникова.

1 АНАЛІЗ МЕТОДІВ ВІДНОВЛЕННЯ СИГНАЛІВ В ІНФОКОМУНІКАЦІЙНИХ ТА КОМП'ЮТЕРНИХ СИСТЕМАХ

1.1 Відновлення сигналу за допомогою ряду Котельникова

Відомий англійський математик Едмунд Тейлор Віттекер свого часу показав, що функція кардинального синусу (sinc) як загальний член кардинального ряду грає центральну роль теорії інтерполяції на сітці рівновіддалених осі абсцис дискретних вузлів. Надалі основні результати Е. Уіттекера були використані в теорії зв'язку та теорії інформації, коли теоремою відліків Котельникова-Шеннона було доведено можливість заміни безперервного одновимірного сигналу з фінітним спектром послідовністю дискретних відліків без втрати інформації.

Відповідно до теореми Котельникова, для того, щоб відновити вихідний безперервний сигнал з дискретизованого з малими спотвореннями (похибками), необхідно правильно обрати крок дискретизації. Якщо найвища частота у спектрі сигналу *u(t)* не перевищує значення F_{max}, то сигнал *u(t)* повністю визначається послідовністю своїх значень у моменти часу, що віддаляються один від одного не більше ніж на $0{,}5/F_{max}$. Іншими словами, частота дискретизації безперервного сигналу $F_д$ має задовольняти умові $F_д \geq 2F_{max}$.

Якщо аналоговий сигнал має низькочастотний спектр, обмежений деякою верхньою частотою F_{max}, (тобто функція *u(t)* має вигляд кривої, що плавно змінюється, без різких змін амплітуди), то навряд чи на деякому невеликому часовому інтервалі дискретизації Δt амплітуда цієї функції може істотно змінитися. Очевидно, що точність відновлення аналогового сигналу за послідовністю його відліків залежить від величини інтервалу дискретизації Δt. Чим він коротший, тим менше відрізнятиметься функція *u(t)* від плавної кривої, що проходить через точки відліків. Однак із зменшенням інтервалу дискретизації Δt суттєво зростає складність та обсяг апаратури що приймає та оброблює

сигнали. При досить великому інтервалі дискретизації Δt зростає можливість спотворення чи втрати інформації при відновленні аналогового сигналу.

Оптимальна величина інтервалу дискретизації встановлюється теоремою Котельникова, доведеною ним у 1933 році (інші назви: – теорема відліків, теорема К. Шеннона, теорема Х. Найквіста). Вперше ця теорема була сформульована в математиці О. Коші, а потім описана повторно Д. Карсоном і Р. Хартлі. Теорема Котельникова має важливе теоретичне і практичне значення, оскільки вона дає можливість правильно здійснити дискретизацію аналогового сигналу і визначає оптимальний спосіб його відновлення на приймальні стороні по відліковим значенням

Згідно з однією з найвідоміших і найпростіших інтерпретацій теореми Котельникова, довільний сигнал *u(t)*, спектр якого обмежений деякою частотою $F_{\max}$ може бути повністю відновлений за послідовністю своїх відлікових значень, наступних з інтервалом часу

$$\Delta t = \frac{1}{2F_{\max}}. \tag{1.1}$$

Інтервал дискретизації Δt та частоту $F_{\max}$ в радіотехніці часто називають відповідно інтервалом та частотою Найквіста. Теорема Котельникова представляється рядом:

$$u(t) = \sum_{k=-\infty}^{\infty} u(k\Delta t)\frac{\sin \omega_{\max}(t - k\Delta t)}{\omega_{\max}(t - k\Delta t)}; \tag{1.2}$$

де k – номер відліку; $u(k\Delta t)$ – значення сигналу в точці відліку; $\omega_{\max}$ – максимальна циклічна (кругова) частота сигналу:

$$\omega_{\max} = 2\pi F_{\max} = \pi\Delta t. \tag{1.3}$$

Базисні функції $S_k(t)$ (рис. 1.1)

$$S_k(t) = \frac{\sin \omega_{\max}(t - k\Delta t)}{\omega_{\max}(t - k\Delta t)} \qquad (1.4)$$

є ортогональними одна одній на інтервалі часу –∞…∞. Вони називаються функціями відліків, базовими функціями, або функціями Котельникова. Кожна з базисних функцій $S_k(t)$ зміщена відносно подібної найближчої функції $S_{k-1}(t)$ або $S_{k+1}(t)$ на інтервал дискретизації Δt.

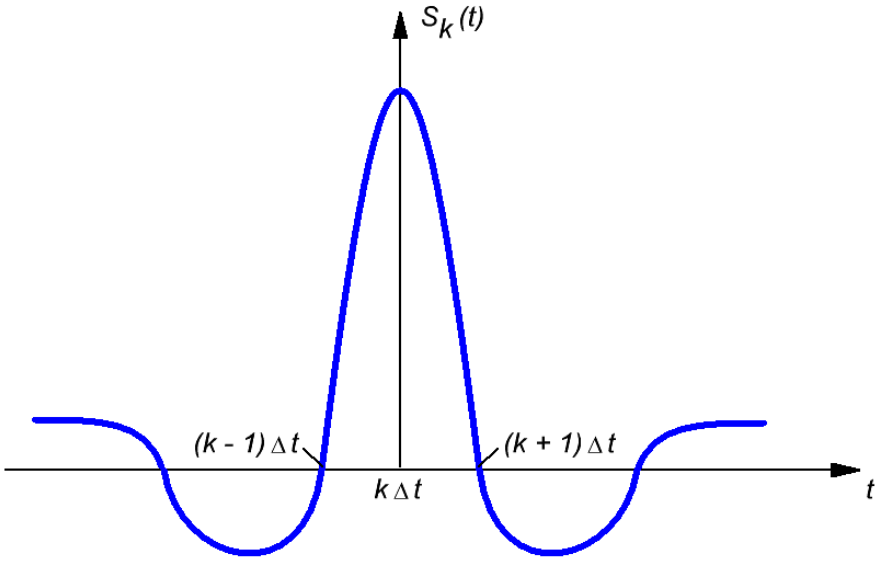


Рисунок 1.1 – Графік базисної функції Котельникова

Інтерпретація (а точніше – апроксимація) первинного безперервного сигналу $u(t)$ рядом Котельникова (1.2) дозволяє його відновити на приймальній стороні з певною точністю. Цей процес показаний на рис. 1.2, на якому базисні функції, для спрощення, показані без аргументу t і побудовані лише кілька перших членів ряду. При підсумовуванні цих членів ряду в будь-які відлікові моменти часу $k\Delta t$, безперервний сигнал абсолютно точно відновлюється незалежно від числа вибраних відліків. В інтервалі ж між відліками сигнал $u(t)$ апроксимується тим точніше, чим більше членів ряду Котельникова використовуються при обчислені суми.

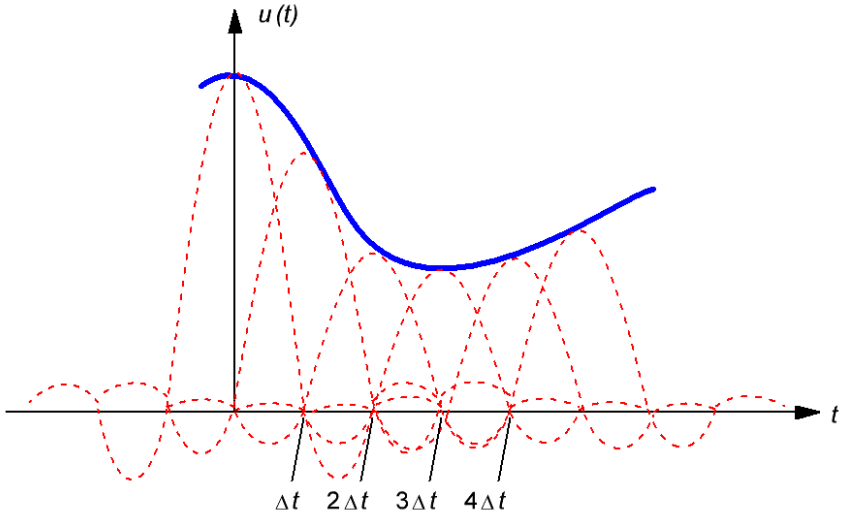


Рисунок 1.2 – Відновлення сигналу u(t) із відліків за допомогою базисних функцій $S_k(t)$, зміщених у часі на величину Δt

Таким чином, представлення аналогових неперервних сигналів у вигляді набору відліків дозволяє використовувати для передавання інформації цифрові системи передачі даних. При цьому на приймальній стороні за допомогою ряду Котельникова сигнал можна відновити з певною точністю. При цьому у моменти часу, що виповідають моментам дискретизації, сигнал відтворюється абсолютно точно, а в моменти часу, розташовані між відліками, точність відтворення сигналу зменшується (рис. 1.3).

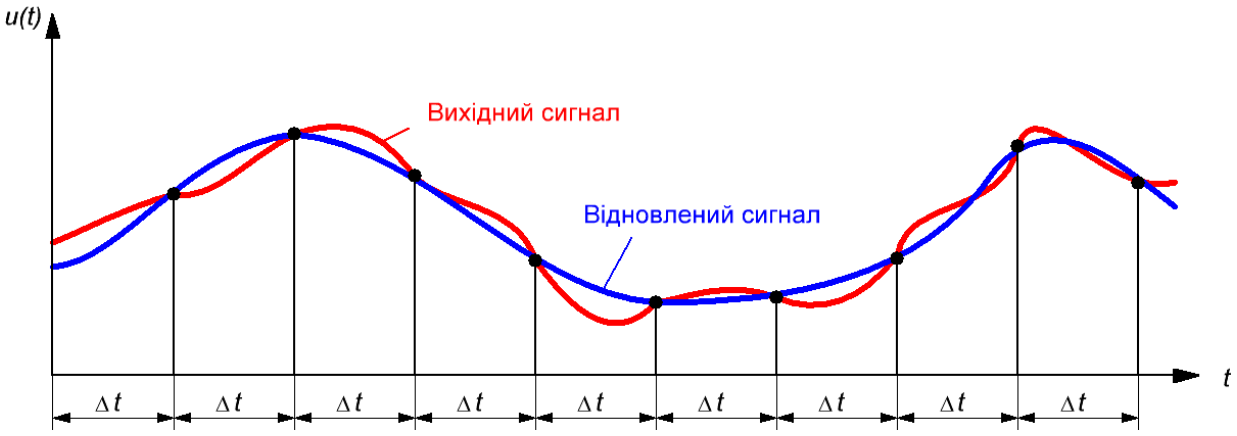


Рисунок 1.3 – Розбіжність вихідного та відновленого сигналу, зумовлена процесом дискретизації

Зменшення точності відновлення сигналу на приймальній стороні спонукає шукати інші математичні функції, які зможуть забезпечити більшу точність, порівняно із використанням ряду Котельникова.

1.2 Відновлення сигналу за допомогою сплайнів

В теоремі Котельникова йдеться про ідеальне відновлення сигналу, що є функцією часу, тобто аналітичної функції, яка може бути розкладена в нескінченний ряд з абсолютною збіжністю. Безумовно, для реальних сигналів ці вимоги не виконуються. Насамперед це пояснюється їх кінцевою тривалістю, оскільки реальні сигнали є функціями часу або кінцеві у просторі (якщо вони залежать від інших фізичних змінних, наприклад, відстаней в метрах, кілометрах тощо). Однією з основних проблем застосування ряду Котельникова для відновлення сигналу в реальних застосунках є необхідність його усічення, що поряд з іншими причинами вносить помилки в процеси відновлення сигналів.

Усі відомі формули оцінки помилок відновлення сигналу базуються на принципі фінітності спектру сигналу, який був покладений ще в основу теорем Котельникова-Шеннона. У той самий час багато дослідників цього питання відмічають, що кількість відліків у вибірках, коли незалежні аргументи є часом, може бути значно менше, ніж відліків функцій часу.

Типовим прикладом функцій, спектри яких є інфінітні, є базисні сплайни. Базисні сплайни мають багато спільного із загальним членом ряду Котельникова внаслідок дуальності (у сенсі рівноцінності аргументів) ядра інтегральних перетворень $K(x, \omega) = \sin(\omega x)/\omega x$. Для завдань формування вибірок сигналів два класи фінітних базисних функцій із компактними носіями – базисні сплайни (B-сплайни) та сімейства вейвлет-функцій, для яких існують алгоритми швидких спектральних перетворень. Поряд з іншими вейвлетами, у таких завданнях застосовуються сплайн-вейвлети.

Сплайн-вейвлети класифікуються за чотирма категоріями:

– ортогональні (Battle-Lemarie);

– напівортогональні (B-сплайни, Shoenberg);

– ортогональні при зрушеннях (Choui);

– біортогональні (Cohen-Daubechies).

Крім того сплайн-вейвлети розрізняються також за значеннями ступенів та/або порядків вейвлетів (рис. 1.4).

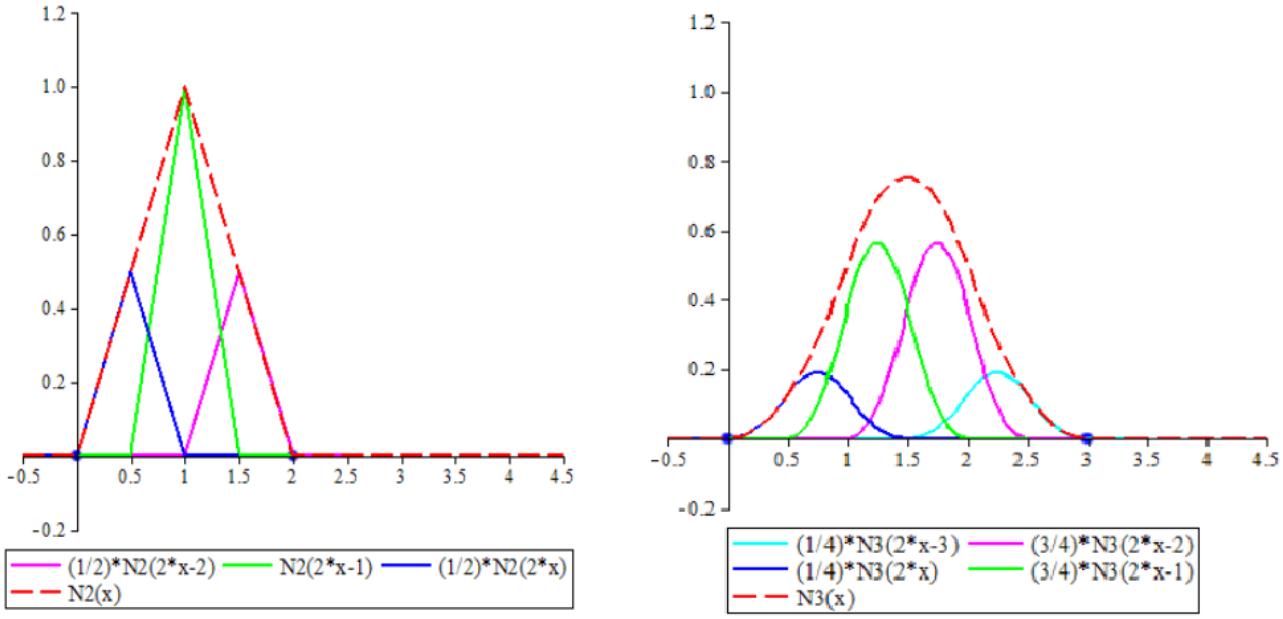


Рисунок 1.4 – Сплайн-вейвлети ступеню 1 порядку 2 (ліворуч)
та ступеню 2 порядку 3 (праворуч)

Для відновлення неперервних сигналів в інфокомунікаційних системах використовуються поліноміальна і дробно-раціональна апроксимації. Разом з тим ці методи мають певні обмеження. Наприклад, при їх використанні в околицях особливих точок можливі непрогнозовані локальні зв'язки, відзначається погана збіжність тощо. Успішно розвивається в останні роки апарат сплайн-апроксимації), яка є серйозним конкурентом традиційних типів апроксимації, оскільки має більшу точність та меншу кількість недоліків.

До числа основних переваг сплайн-апроксимації можна віднести наступне:

– сплайни більш стійкі відносно локальних збурень, але така поведінка сплайна в околицях точок не показана на наведенні сплайна в цілому, як, наприклад, це має місце при поліноміальній інтерполяції;

– краща збіжність сплайн-інтерполяції у відмінності від поліноміальної.

Зокрема, для функцій з нерегулярними властивостями гладкості використання сплайн-інтерполяцією є пріоритетним методом відновлення сигналів.

Окрім наведених особлистей, сплайни володіють корисними екстремальними властивостями. Важливою практичною перевагою є те, що сплайн-функції досить просто реалізуються за допомогою програмних продуктів.

1.3 Порівняння різних методів відновлення сигналів

При відновленні неперервного сигналу *s(t)* за його дискретними значеннями відновлений сигнал у точності відповідає вихідному сигналу у вузлових точках, але в інших (проміжних) точках, він може від нього відрізнятися. Отже, у цьому випадку, виникає питання мінімізації цієї відмінності, оскільки чим менше величина відхилення вихідного сигналу від відновленого, тим більш точніше можна відновити заданий сигнал.

Нехай на проміжку [0; T] заданий безперервний сигнал *s(t)*. Розіб'ємо цей відрізок [0; T] точками Δ: $0 = t_0 < t_1 < ... < t_N = T$ на часткові проміжки $[t_i,\ t_{i+1}]$, де $i = 0, N-1$, на кожному з яких побудуємо багаточлен із певним ступенем. Нехай у якості подібних багаточленів будуть використані кубічні сплайни. Таким чином, отримуємо на проміжку [0; T] кусково-безперервну функцію, яка і визначатиме кубічний сплайн.

Нехай на відрізку [0; T] у вузлах сітки Δ: $0 = t_0 < t_1 < ... < t_N = T$ задані значення деякої функції $s_i = s(t_i)$, $i = 0, N$.

Відповідно [11], інтерполяційним кубічним сплайном $S_3(t_i)$ називають сплайн, що відповідає наступним умовам:

$$S_3(t_i) = S_i, \quad S_3(t_{i+1}) = s_{i+1}, \quad i = 0, N-1 \tag{1.5}$$

– на кожному відрізку $[t_i,\ t_{i+1}]$, $i = 0, N$ він є багаточленом третього ступеня;

– на всьому відрізку [0; T] він має неперервні похідні:

$$S_3'(0+0) = f'(0),$$
$$S_3'(T-0) = f'(T).$$

Відповідно до теореми Котельникова [13], відліки беруться через крок дискретизації Δt, що визначається формулою (1.1). При цьому базисними функціями ряду Котельникова є функції відліків, що перетворюють ортогональну на нескінченному проміжку (–∞; +∞) систему функцій (1.2).

Нехай у будь-який момент часу на інтервалі [0; T] визначений неперервний сигнал *s(t)*, що задається функцією:

$$s(t) = 2\frac{\sin\left[16\pi\left(t-\frac{1}{2}\right)\right]}{16\pi\left(t-\frac{1}{2}\right)} - 3\frac{\sin\left[8\pi\left(t-\frac{1}{2}\right)\right]}{8\pi\left(t-\frac{1}{2}\right)} \qquad (1.6)$$

На проміжку [0; T] створимо рівномірну сітку, що розіб'є інтервал на наступні проміжки Δ: $0 = t_0 < t_1 < ... < t_N = T$, де $t_1 - t_0 = t_2 - t_1 = \ldots = t_N - t_{N-1} = \Delta t$ – крок дискретизації. Нехай в точках розбиття інтервалу відомі значення функції $s = s(t)$: $s(t_i) = s_i$, $i = 0, N$.

Відновлення безперервного вихідного сигналу $s(t)$ здійснимо за допомогою ряду Котельникова (1.2) і кубічним сплайном виду (1.5).

Розглянемо сигнал виду (1.6), що має спектр з обмеженою максимальною частотою $F_{\max} = 8$ кГц. Відповідно до теореми Котельникова, частота дискретизації буде дорівнювати:

$$F_д = 2F_{\max} = 2 \cdot 8 = 16 \text{ кГц}. \qquad (1.7)$$

При цьому інтервал дискретизації (тривалість інтервалу часу між відліками) буде дорівнювати:

$$\Delta t = \frac{1}{2F_{\max}} = \frac{1}{2 \cdot 8000} = 62{,}5 \text{ мкс}. \qquad (1.8)$$

У якості вузлів інтерполяції будемо розглядати відліки вхідного (первинного) сигналу *s(t)*.

Результати моделювання подібної системи, виконані у програмному середовищі Mathcad показані на рис. 1.5. На цьому рисунку цифрою 1 позначений графік вхідного сигналу *s(t)*, обчисленого за формулою (1.6), цифрою 2 – значення сигналу в точках інтерполяції, цифрою 3 – результати апроксимації сигналу рядом Котельникова, а цифрою 4 – апроксимація сигналу кубічним сплайном.

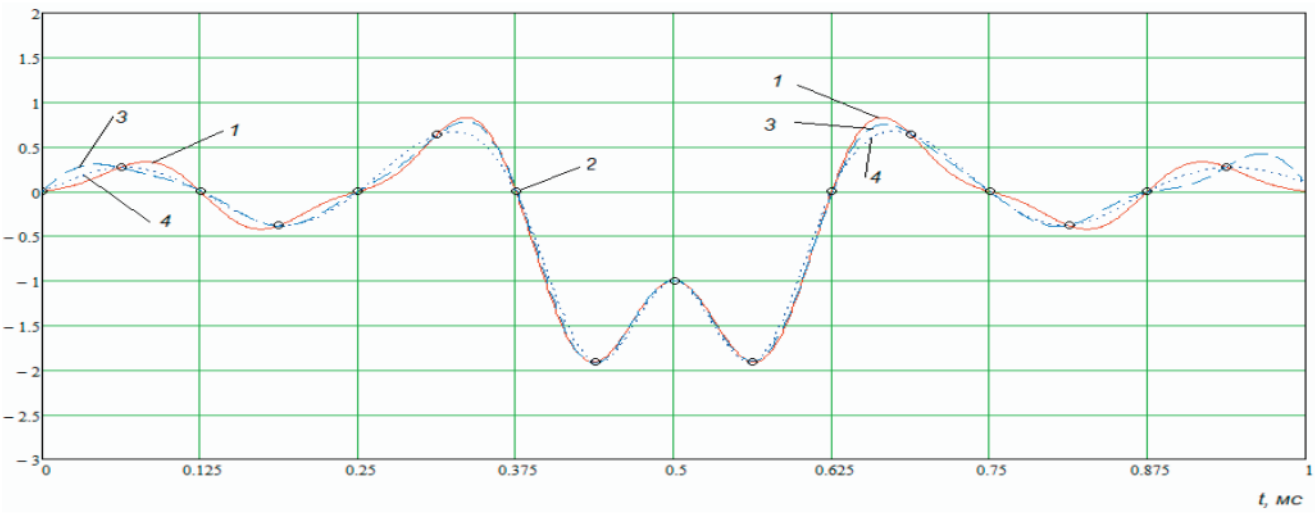


Рисунок 1.5 – Апроксимація сигналу *s(t)* за допомогою ряду Котельникова та кубічним сплайном

Для обчислення точності апроксимації відновлення сигналу позначимо $\varphi(t)$ – функція, що використовується для апроксимації. У нашому прикладі такою функцією є в одному випадку ряд Котельникова, а в іншому – кубічний сплайн. Відхилення значень відновленого сигналу від його вихідного значення при апроксимації поруч Котельникова (графік, позначений цифрою 1) та кубічним сплайном (графік, позначений цифрою 2) показано на рис. 1.6.

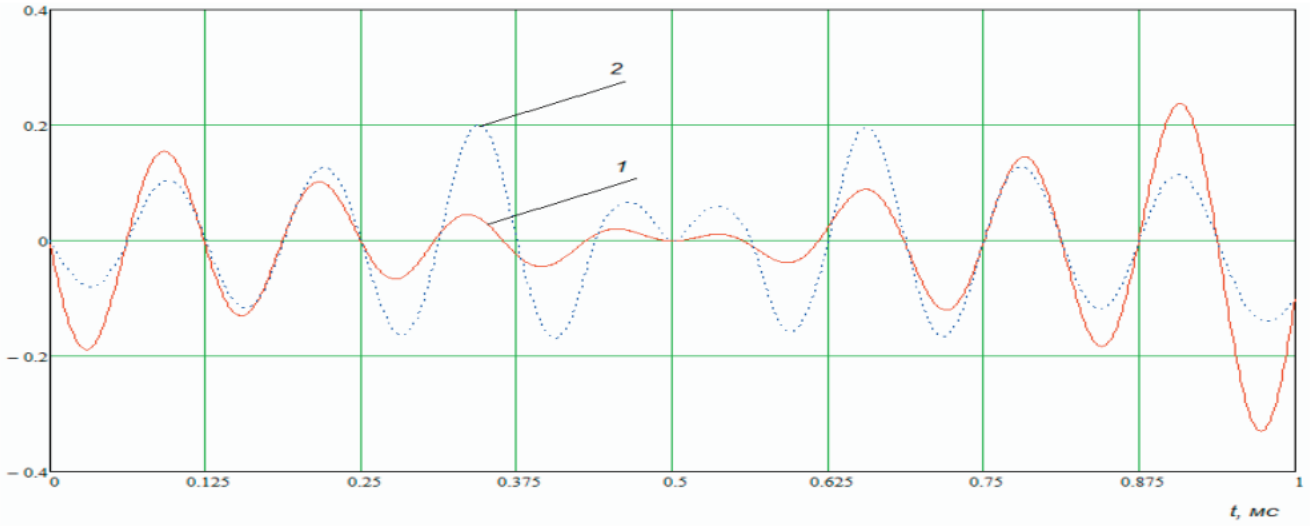


Рисунок 1.6 – Відхилення відновленого сигналу від первинного

Розглянемо довільний проміжок $[t_i,\ t_{i+1}]$, де $i = 0, N-1$. Розіб'ємо кожен з цих інтервалів на n частин. Точки розбиття кожної частини позначимо через t_{ij}, де $i = 0, N$, $i = 0, n$. Таким чином, на кожному з проміжків $[t_i,\ t_{i+1}]$, $i = 0, N-1$ задана власна сітка розбиття Δ_i: $t_i = t_{i0} < t_{i1} < ... < t_{iN} = t_{i+1}$, з кроком розбиття $\Delta t_i = 1/64$.

Відповідно формулі (6) роботи [11] визначимо середньоквадратичну помилку апроксимації ε_n на кожному із часткових проміжків проміжок $[t_i,\ t_{i+1}]$, де $i = 0, N-1$, значення ε_n зведемо до таблиці 1.1.

Обчислимо середньоквадратичне відхилення (СКВ) різниці відновленого и вихідного сигналів:

$$\sigma_n = \sqrt{\frac{1}{k}\sum_{i,j}\left(\varphi(t_{ij}) - \varepsilon_n\right)^2},\quad i = 0, N,\quad j = 1, n. \tag{1.9}$$

де $\varphi(t_{ij})$ – значення функції, що апроксимує вихідний сигнал в точках розбиття кожного часткового проміжка $[t_i,\ t_{i+1}]$, $i = 0, N-1$;

ε_n – середня величина відхилення значення відновленого сигналу на кожному частковому проміжку проміжка $[t_i,\ t_{i+1}]$, $i = 0, N-1$;

t_{ij} – точки розбиття кожного часткового проміжка $[t_i,\ t_{i+1}]$, $i = 0,\ N-1$, $j = 0, n$;

$k = n$ – кількість точок розбиття на кожному частковому інтервалі $[t_i,\ t_{i+1}]$, $i = 0, N - 1$.

Результати значень СКВ різниці відновленого та вихідного сигналу поруч Котельникова та кубічним сплайном для кожного часткового проміжку $[t_i,\ t_{i+1}]$, $i = 0, N - 1$ приведені в таблиці 1.1.

Таблиця 1.1 – Значення СКВ відновленого та вихідного сигналів

№	Проміжок	Межі проміжка, мкс	ε_n ряду Котельникова	ε_n сплайну	σ_n ряду Котельникова	σ_n сплайну
1	$[t_0,\ t_1]$	[0,000 0,063]	0,119	0,05	0,059	0,025
2	$[t_1,\ t_2]$	[0,063, 0,125]	0,098	0,066	0,048	0,032
3	$[t_2,\ t_3]$	[0,125, 0,188]	0,083	0,074	0,041	0,036
4	$[t_3,\ t_5]$	[0,188, 0,250]	0,066	0,082	0,031	0,038
5	$[t_4,\ t_6]$	[0,250, 0,313]	0,039	0,103	0,023	0,051
6	$[t_5,\ t_7]$	[0,313, 0,375]	0,023	0,128	0,02	0,061
7	$[t_6,\ t_8]$	[0,375, 0,438]	0,027	0,106	0,017	0,053
8	$[t_7,\ t_9]$	[0,438, 0,500]	0,012	0,041	0,0072	0,022
9	$[t_8,\ t_9]$	[0,500, 0,563]	0,0048	0,036	0,0049	0,02
10	$[t_9,\ t_{10}]$	[0,563, 0,625]	0,022	0,099	0,015	0,049
11	$[t_{10},\ t_{11}]$	[0,625, 0,688]	0,059	0,121	0,026	0,062
12	$[t_{11},\ t_{12}]$	[0,688, 0,750]	0,079	0,107	0,035	0,05
13	$[t_{12},\ t_{13}]$	[0,750, 0,813]	0,091	0,081	0,046	0,04
14	$[t_{13},\ t_{14}]$	[0,813, 0,875]	0,117	0,076	0,056	0,036
15	$[t_{14},\ t_{15}]$	[0,875, 0,938]	0,149	0,072	0,075	0,037
16	$[t_{15},\ t_{16}]$	[0,938, 1,000]	0,225	0,102	0,092	0,038

Аналіз отриманих результатів показує, що на проміжках інтерполяції «з плавною зміною амплітуди» (проміжки $[t_0,\ t_1]$, $[t_1,\ t_2]$, $[t_2,\ t_3]$, $[t_{12},\ t_{13}]$, $[t_{13},\ t_{14}]$, $[t_{14}, t_{15}]$, $[t_{15},\ t_{16}]$), де форма сигналу безперервно змінюється без «підвищеної крутизни амплітуди» найкращі результати дає сплайн-інтерполяція. А на ділянках з «підвищеною крутістю амплітуди» (проміжки $[t_3,\ t_4]$, $[t_4,\ t_5]$, $[t_5,\ t_6]$, $[t_6,\ t_7]$, $[t_7,\ t_8]$, $[t_8,\ t_9]$, $[t_9,\ t_{10}]$, $[t_{10},\ t_{11}]$, $[t_{11},\ t_{12}]$) – інтерполяційний ряд Котельникова.

Таким чином, для відновлення безперервних сигналів із підвищеною крутизною амплітуди і швидким зростанням значень функції, необхідно

розглянути можливість використання іншого математичного апарату, який дозволить зменшити помилку апроксимації і буде ефективнішим для відновлення заданого виду сигналів.

Одним із варіантів підвищення точності апроксимації сигналів є використання вейвлет-функцій [14], які мають вигляд коротких обмежених часу хвиль, що змінюються в масштабі і переміщуються по осі часу, завдяки чому ці «короткі хвилі» здатні відображати локальні зміни в сигналі.

1.4 Висновки за розділом

Проведено порівняння двох способів відновлення безперервних сигналів за їх дискретними відліками за допомогою ряду Котельникова та сплайн-функцій для сигналів з «підвищеною крутизною амплітуди».

Отримані оцінки похибки апроксимації сигналу рядом Котельникова і кубічним сплайном дозволяють стверджувати, що при невеликій частоті дискретизації, у разі коли амплітуда сигналу має плавну зміну краще використовувати сплайн-інтерполяцію, а для сигналів з різким зростанням амплітуди краще використовувати інтерполяційний ряд Котельникова. Інакше помилка інтерполяції може досягати 6,2% та 9,2% у першому та у другому випадках відповідно

Проведене дослідження показує наявність можливості використання іншого математичного апарату, який дозволить відновлювати сигнали, що характеризуються наявністю швидких осциляцій з більшою точністю, ніж ряд Котельникова чи кубічні сплайни.

2 ВИБІР СПОСОБУ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МОДУЛЯ ВІДНОВЛЕННЯ ТРАФІКУ

2.1 Аналіз методів практичної реалізації математичних моделей

На сьогоднішній день основним інструментом для розробки та дослідження сучасної вузлів комп’ютерної та інфокомунікаційної апаратури, у тому числі і апаратури для відновлення інформаційних сигналів, є персональні комп'ютери. Заміна фізичних вузлів їх віртуальними математичними еквівалентами дозволяє суттєво скоротити матеріальні та часові витрати на розробку радіотехнічних та комп’ютерних пристроїв.

На сьогоднішній день існують два основних напрямки в моделюванні електронних та комп’ютерних пристроїв, кожен з яких має свої обмеження, переваги та недоліки, але які в цілому взаємно доповнюють один одного при вирішенні практичних завдань.

Перший напрямок заснований на синтезі складних електричних кіл за допомогою найпростіших універсальних віртуальних електронних компонентів. Для цього існує певна велика кількість програмного забезпечення, прикладом якого є Electronic Workbench пакету Multisim компанії National Instruments [15], Proteus компанії Labcenter Electronics [16] або LTSpice компанії Analog Devices [17], більшість з яких у якості математичного апарату використовують програму симуляції аналогових схем і цифрової логіки, описаною мовою SPICE (Personal Simulation Program with Integrated Circuit Emphasis, PSpice) [18].

Другий напрямок заснований на розробці математичної моделі, яка відображає специфіку конкретного пристрою або класу пристроїв і дозволяє безпосередньо визначати параметри, які при використанні універсальних моделей можна визначити тільки непрямим шляхом. Прикладом подібного програмного забезпечення є онлайн-симулятори eDesignSuite, розроблений компанією ST Microelectronics [19], SpeedFit Design Simulator, розроблений компанією Wolfspeed [20], та Elite Power Simulator виробництва компанії On Semicondactor [21, 22].

Для моделювання за допомогою першого способу достатньо придбати відповідне програмне забезпечення та вивчити технічну документацію до нього, а от при використанні другого способу виникає завдання перетворення математичної моделі, яка написана високорівневою мовою математики, в набір низькорівневих інструкцій процесору персонального комп'ютера. При побудові різних характеристик це завдання ускладнюється необхідністю додаткового опису також правил побудови цих характеристик (алгоритму проведення досліджень).

Незважаючи на широкий вибір інструментів для вирішення цього завдання, існуючі способи мають ряд істотних недоліків, основними з яких є: низька швидкість розрахунку, наявність спеціальних знань у галузі програмування та складність інтеграції з іншими системами. У той же час сучасне програмне забезпечення, що працює під управлінням операційних систем Windows, широко використовує бібліотеки, що динамічно зв’язуються (Dynamic Link Library – DLL). При цьому інформація про використання цього способу для вирішення задач моделювання у відомих публікаціях практично відсутня.

Найпопулярнішим на сьогоднішній день способом реалізації математичних розрахунків є використання універсальних обчислювальних програм, таких як MATLAB, виробництва компанії MathWorks [23], MathCAD, виробництва компанії Mathsoft [24], або LabVIEW виробництва компанії National Instruments [25]. Проте у деяких випадках, наприклад, при необхідності інтеграції математичних моделей в інтегровані середовища розробки (Integrated Development Environment, IDE), використання універсальних обчислювальних програм пов’язано із певними складнощами.

При використанні універсальних обчислювальних програм математична модель та правила проведення дослідження описуються внутрішньою високорівневою мовою обчислювальної програми, синтаксис якого близький до формальних мов математики та логіки (рис. 2.1). Далі вихідний текст перетворюється на внутрішній алгоритм розрахунку з використанням готових заготовок інструкцій, які є невід'ємною частиною програмного продукту. Результати обчислень переважно виводяться у файл із вихідним текстом моделі.

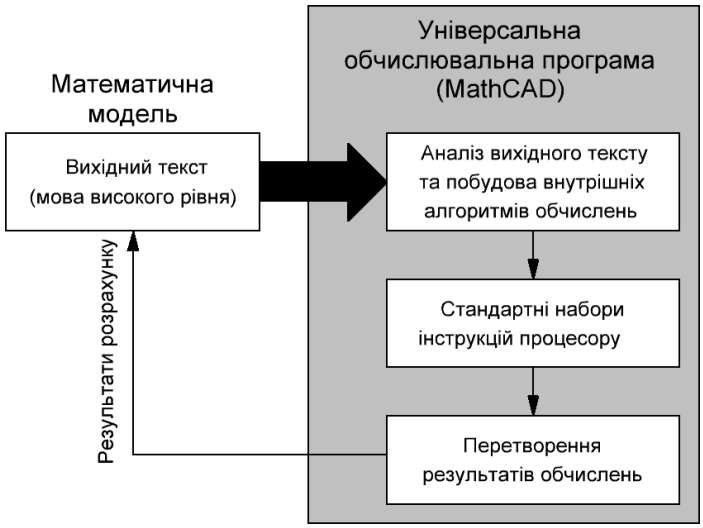


Рисунок 2.1 – Перетворення математичної моделі на набір інструкцій процесору при використанні універсальних обчислювальних програм

При розробці спеціалізованого програмного забезпечення математична модель та правила проведення дослідження описуються мовою програмування (переважно це одна з мов високого рівня: C/C++, Delphi, Fortran тощо) та разом з вихідними текстами технічної частини програми перетворюються компілятором на набір інструкцій, що представляє самостійну програму (рис. 2.2). Ця програма може бути як просту програму для розрахунку та дослідження конкретного радіотехнічного пристрою, так і складну САПР.

Головним недоліком першого способу є складність інтеграції отриманого модуля з іншими обчислювальними програмами або системами автоматизованого проектування, а також залежність від проектування (програми-оболонки). Проста інтеграція можлива лише у випадку, коли система, у якому інтегрується отриманий модуль, створена у тому ж самому середовищі розробки. В іншому випадку необхідно конвертувати отриманий модуль або використовувати технологію ActiveX [26], що не завжди зручно, а в деяких випадках і неможливо. Також до недоліків першого способу слід віднести низьку швидкість обчислень у деяких системах (наприклад, MathCAD), що накладає обмеження при проведенні великого об'єму обчислень.

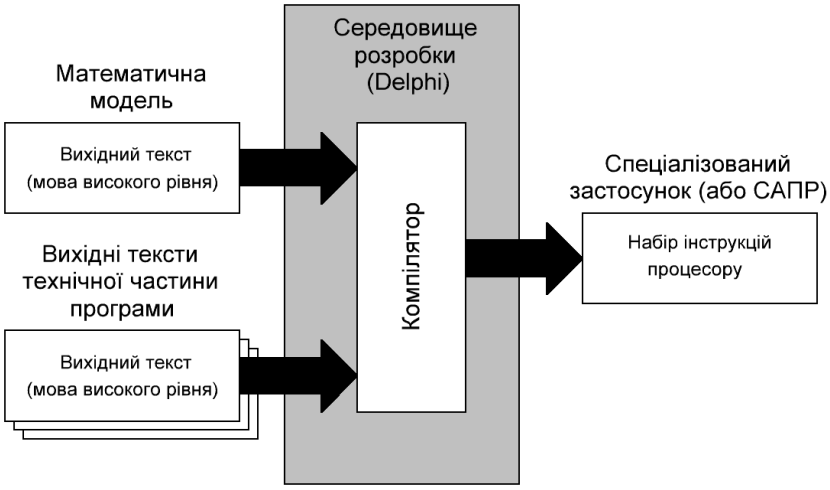


Рисунок 2.2 – Перетворення математичної моделі на набір інструкцій процесору при створенні самостійних застосунків

Основними недоліками другого способу є складність створення готового програмного продукту. Для створення програмного забезпечення, що задовольняє сучасним вимогам, потрібна наявність спеціальних знань у галузі програмування. Загальний обсяг технічних модулів може перевищувати обсяг модуля з описом математичної моделі у сотні разів [26]. Для цього способу характерна обмеженість проведених досліджень, кількість і якість яких після створення програмного забезпечення змінити вже неможливо.

При використанні технології бібліотек, що динамічно-зв’язуються, математична модель описується однією з мов програмування високого рівня і після компіляції перетворюється на набір інструкцій процесору (рис. 2.3). Бібліотека містить у собі мінімальну кількість технічного коду, більшість якого генерується автоматично середовищем розробки. Це значно скорочує час, що витрачається створення DLL, і вимагає мінімум спеціальних знань у сфері програмування. Слід зазначити, що час, що витрачається створення бібліотеки, приблизно дорівнює часу, що витрачається створення файлу в універсальних обчислювальних програмах. Оскільки бібліотека містить готові інструкції процесору, то швидкість обчислення практично дорівнює швидкості обчислень при реалізації у вигляді самостійного продукту.

Рисунок 2.3 – Перетворення математичної моделі на набір інструкцій процесору при використанні технології динамічно-зв’язаних бібліотек

Основним недоліком оформлення математичної моделі у вигляді динамічно-зв’язаних бібліотек, неможливість її самостійного використання. Використання бібліотеки можливе лише за наявності спеціальної програми, в якій реалізовано модуль сполучення з бібліотекою. Однак цей недолік є також і перевагою, оскільки до однієї бібліотеки одночасно можуть звертатися кілька програм, а одна програма, що має один модуль сполучення, може одночасно звертатися до декількох бібліотек (рис. 2.4).

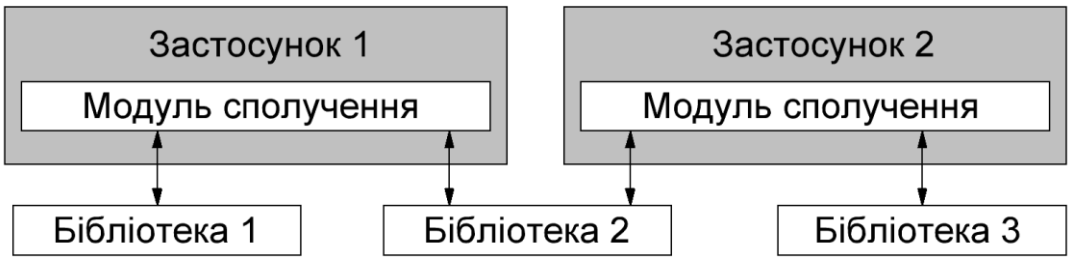


Рисунок 2.4 – Варіанти використання кількох бібліотек, які мають однакову структуру

З точки зору головної програми, DLL є її невід'ємною частиною. Слід зазначити, що технологія DLL широко використовується у програмуванні, а час, що витрачається на створення модуля сполучення з DLL, невеликий і часто зводиться лише до опису ресурсів, що знаходяться в DLL [26].

Порівняльна оцінка розглянутих методів наведена в таблиці 2.1. Вочевидь, що використання DLL характеризується необхідною якістю та меншими витратами часу, ніж при використанні інших способів.

Таблиця 2.1 – Порівняльний аналіз способів реалізації математичної моделі

Критерій оцінювання	**Універсальна обчислювальна система**	**Самостійний програмний застосунок**	**Динамічно-зв'язана бібліотека**
Швидкість реалізації математичної моделі	Висока (+)	Низька (–)	Висока (+)
Рівень знань в галузі програмування	Не потребує (+)	Високий (–)	Базовий (+)
Швидкість проведення обчислень	Низька (–)	Висока (+)	Висока (+)
Складність інтеграції з іншими програмними застосунками	Складна (–)	Середня (±)	Проста (+)
Можливість самостійного використання	Складна (–)	Є самостійним програмним застосунком (+)	Неможливо (–)

Подібний підхід вже реалізовувався при дослідженнях характеристик вузлів інфокомунікаційного обладнання, зокрема за допомогою спеціалізованого застосунку TechProgect [27, 28], однак існую і інший спосіб реалізації математичних моделей, придатних для практичного використання, наприклад використання віртуальних лабораторій [29, 30].

2.2 Опис універсальної віртуальної лабораторії Labs

Віртуальна лабораторія Labs складається із ядра та лабораторних макетів (рис. 2.5). Ядро програми містить користувацький інтерфейс, до складу якого входить інтерактивна схема із функцією масштабування, органи керування процесом моделювання та параметрами досліджуваної схеми, віртуальні вимірювальні прилади та усі додаткові програмні модулі, що необхідні для проведення досліджень.

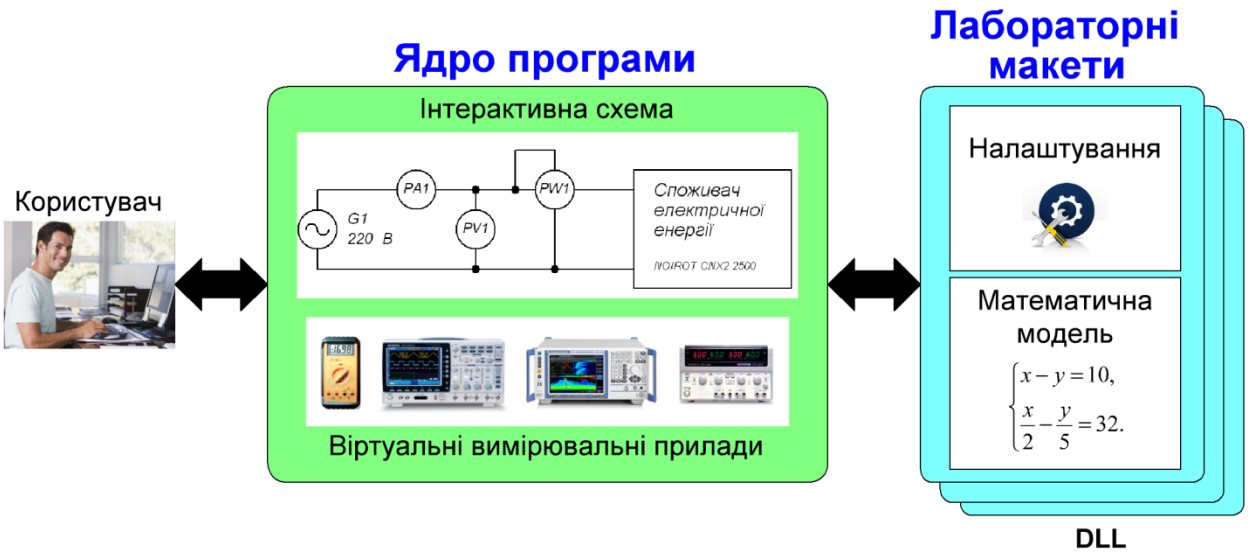


Рисунок 2.5 – Принцип побудови віртуальної лабораторії

Інформацію про параметри досліджуваної схеми (параметри компонен-тів, перелік та налаштування вимірювальних приладів, параметри органів оперативного регулювання) а також математична модель, за допомогою якої відбувається обчислення роботи схеми розташовані у файлах лабораторних макетів, що виконані за технологією динамічно-зв’язаних бібліотек (Dynamic Link Library – DLL).

Після підключення лабораторного макету (файлу DLL) до ядра він фактично стає його невід’ємною частиною, що дозволяє досягти максимальної уніфікації програмного забезпечення та максимальної швидкості проведення обчислень.

Завдяки використанню технології DLL та продуманої системи налашту-вань віртуальна лабораторія є гнучкою у використанні, а створення нових лабораторних макетів (створення нових DLL) займає мінімум часу та потребує лише базових знань у області програмування. Крім того, оскільки лабораторний макет фактично є самостійним програмним забезпеченням, а технологію DLL підтримують майже усі виробники систем автоматизованого проектування для створення програмного забезпечення, то нові лабораторні макети можуть

розроблюватися самостійно на будь-якій мові програмування, та будь якій системі розробки програмного забезпечення, що підтримує дану технологію.

Усі лабораторні макети виконані у єдиному стилі (рис. 2.6). Інтерфейс лабораторних макетів є інтуїтивно зрозумілим і потребує мінімум часу на його вивчення. Зміст та призначення органів керування макетом та віртуальними вимірювальними приладами максимально наближені до реальних аналогів.

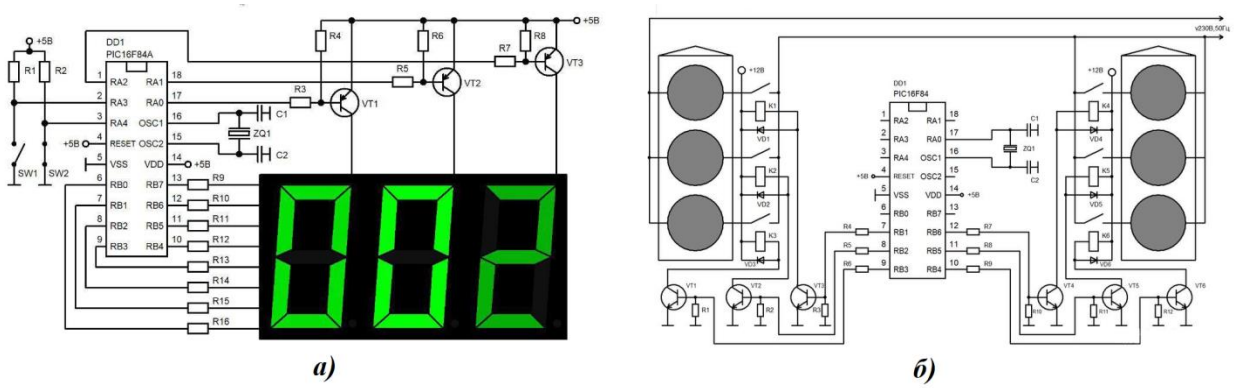


Рисунок 2.6 – Зовнішній вигляд лабораторних макетів, призначеного для програмування мікроконтролерів

Лабораторні макети не потребують значних обчислювальних ресурсів і автоматично підстроюються під параметри конкретного комп'ютера. Вони ма-ють можливість індивідуальної настройки параметрів елементів (для запобігання повторюваності результатів вимірів у різних бригад студентів).

Завдяки використанню технології DLL може бути досягнута висока швидкість обчислень, що у деяких випадках може перевищувати швидкість протікання досліджуваних процесів у реальному часі. При цьому може забезпечуватися максимальна відповідність форми та характеру поведінки досліджуваних сигналів, наприклад, на екрані віртуального осцилографа (рис. 2.7) його реальному аналогу.

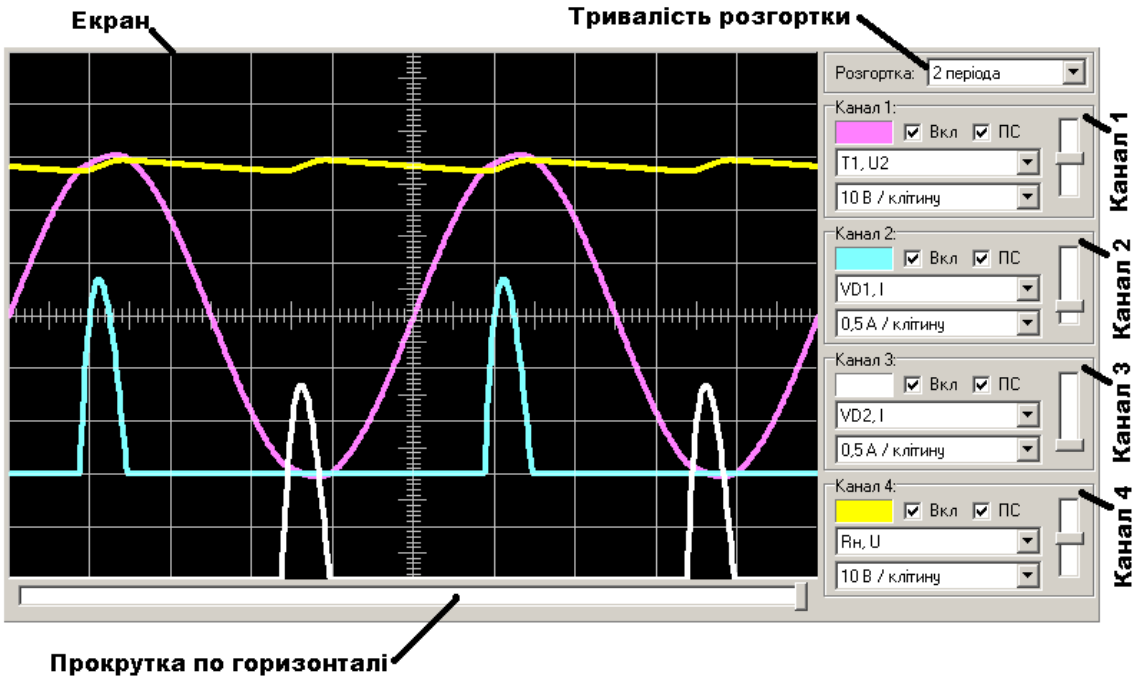


Рисунок 2.7 – Зовнішній вигляд віртуального осцилографа

Головними особливостями віртуальної лабораторії є:

– можливість дослідження широкого кола процесів у різних галузях техніки;

– простота створення нових лабораторних макетів;

– кожен лабораторний макет може містити кілька досліджуваних схем;

– єдиний стиль інтерфейсу для проведення досліджень;

– підтримка кількох мов інтерфейсу;

– широкий набір контрольно-вимірювальних приладів із можливістю гнучкого налаштування;

– висока швидкість та достовірність результатів моделювання;

– можливість зміни параметрів досліджуваної схеми без зупинки моделювання із максимально достовірним відображенням усіх перехідних процесів.

Віртуальний лабораторний практикум, порівняно з аналогічними практикумами має наступні переваги:

– лабораторний практикум містить віртуальні лабораторні макети, які набагато зручніші при встановленні, не потребують спеціального трифазного живлення та подальшого обслуговування;

– підтримка кількох мов інтерфейсу дозволяє використовувати лабораторний практикум для підготовки іноземних студентів;

– лабораторні макети є самостійними програмами і не потребують додаткового програмного забезпечення;

– адаптація програмного забезпечення для використовування мінімуму обчислювальних ресурсів не потребує потужних комп'ютерів;

– інтуїтивно зрозумілий інтерфейс та виконання лабораторних макетів у єдиному стилі мінімізує час освоєння правил роботи з макета-ми;

– велика швидкість моделювання без погіршення його якості зменшує час обчислення перехідних процесів при проведені досліджень – це дозволяє зменшити у цілому час на проведення вимірів;

– один лабораторний макет може містити декілька досліджуваних схем і придатний для проведення декількох лабораторних робіт;

– можливість індивідуального встановлення параметрів елементів для кожного макета запобігає повторюваності результатів вимірів у різних бригад.

Головною сферою застосування віртуальної лабораторії є використання у закладах усіх рівнів освіти для проведення лабораторних робіт та вивчення широкого кола процесів, дослідження яких за допомогою фізичних установок пов’язано із значними матеріальними витратами або має нижчу якість порівняно із дослідженням за допомогою математичних моделей. У першу чергу це теоретичні дисципліни, пов’язані із електричними процесами, спостерігання яких у реальному житті пов’язано із можливістю ураження електричним струмом, споживанням значної кількості енергії, або, через високу швидкість протікання процесів, потребує для дослідження дорогого вимірювального обладнання.

Основними дисциплінами, у яких можна застосувати дану лабораторію є:

– фізика (електричні явища);

– електротехніка;

– теорія електричних кіл;

– теорія електричного зв’язку;

– теорія автоматизованого керування;

– електроживлення та електропостачання;

– енергозберігаючі технології;

– аналогова та цифрова схемотехніка;

– мікроконтролерна техніка;

– пристрої автоматизації;

– робототехніка;

– пристрої Інтернету речей;

– інтелектуальні пристрої.

Віртуальну лабораторію можна використовувати у навчальних закладах практично усіх рівнів освіти:

– школах;

– коледжах;

– закладах вищої освіти;

– центрах підвищення кваліфікації.

Окрім використання для навчання, віртуальну лабораторію можна також використовувати у науковій роботі та під час проектування широкого спектру технічних пристроїв:

– перевірка достовірності математичних моделей;

– розробка та тестування програмного забезпечення;

– проведення наукових досліджень.

2.3 Висновки за розділом

Результати аналізу існуючих методів практичної реалізації математичних моделей показав, що використання технології динамічно-зв’язаних бібліотек має певну низку переваг, головною з яких є можливість подальшої інтеграції

розробленого програмного забезпечення з іншими програмними застосунками, у том числі і з системами автоматизованого проєктування (САПР).

Таким чином, у якості подальшого напряму практичної реалізації розробленої моделі обираємо технологію динамічно-зв’язаних бібліотек, з використання у якості інтерфейсної частини програмного застосунку віртуальної лабораторії «Labs».

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВІДНОВЛЕННЯ ТРАФІКУ ЗА ДОПОМОГОЮ РЯДУ КОТЕЛЬНИКОВА

3.1 Інтерфейсна частина макету

Лабораторний макет складається із двох файлів:

– файлу ядра Labs.exe;

– файлу лабораторного макета Math01.dll (динамічно-зв’язаної бібліотеки).

Для запуску лабораторного макета (макет працює тільки під керування операційних систем Windows) необхідно двічі клацнути лівою кнопкою миші по файлу Labs.exe і у вікні, що відкриється, обрати лабораторний макет «Ряд Котельникова» (рис. 3.1).

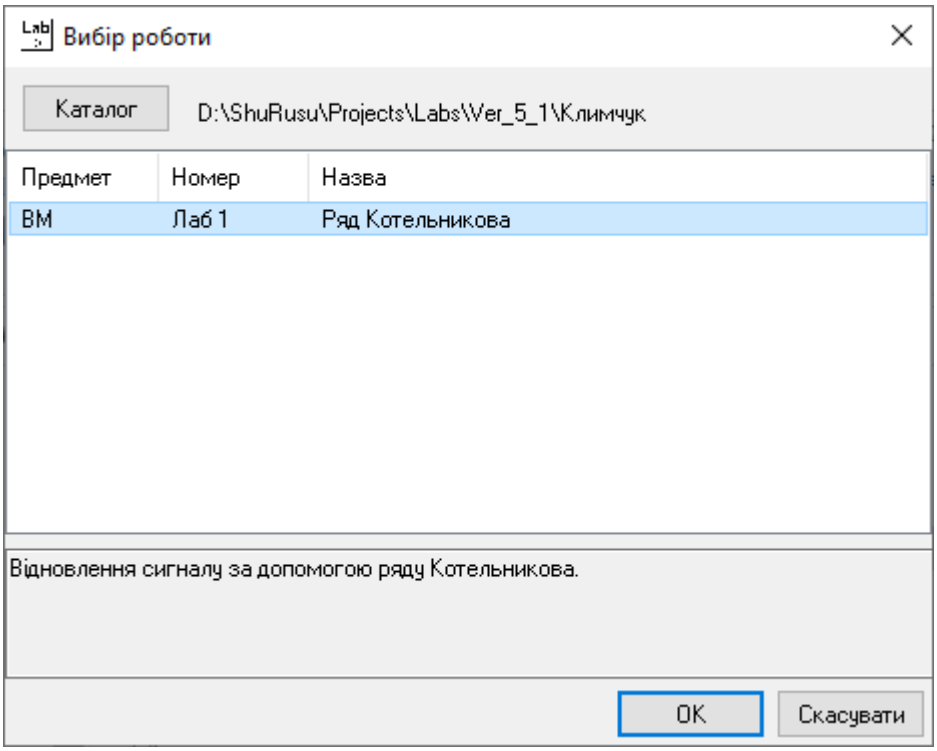


Рисунок 3.1 – Вікно вибору лабораторного макету

Після цього відкриється лабораторний макет, призначений для дослідження методів відновлення трафіку за допомогою ряду Котельникова (рис. 3.2).

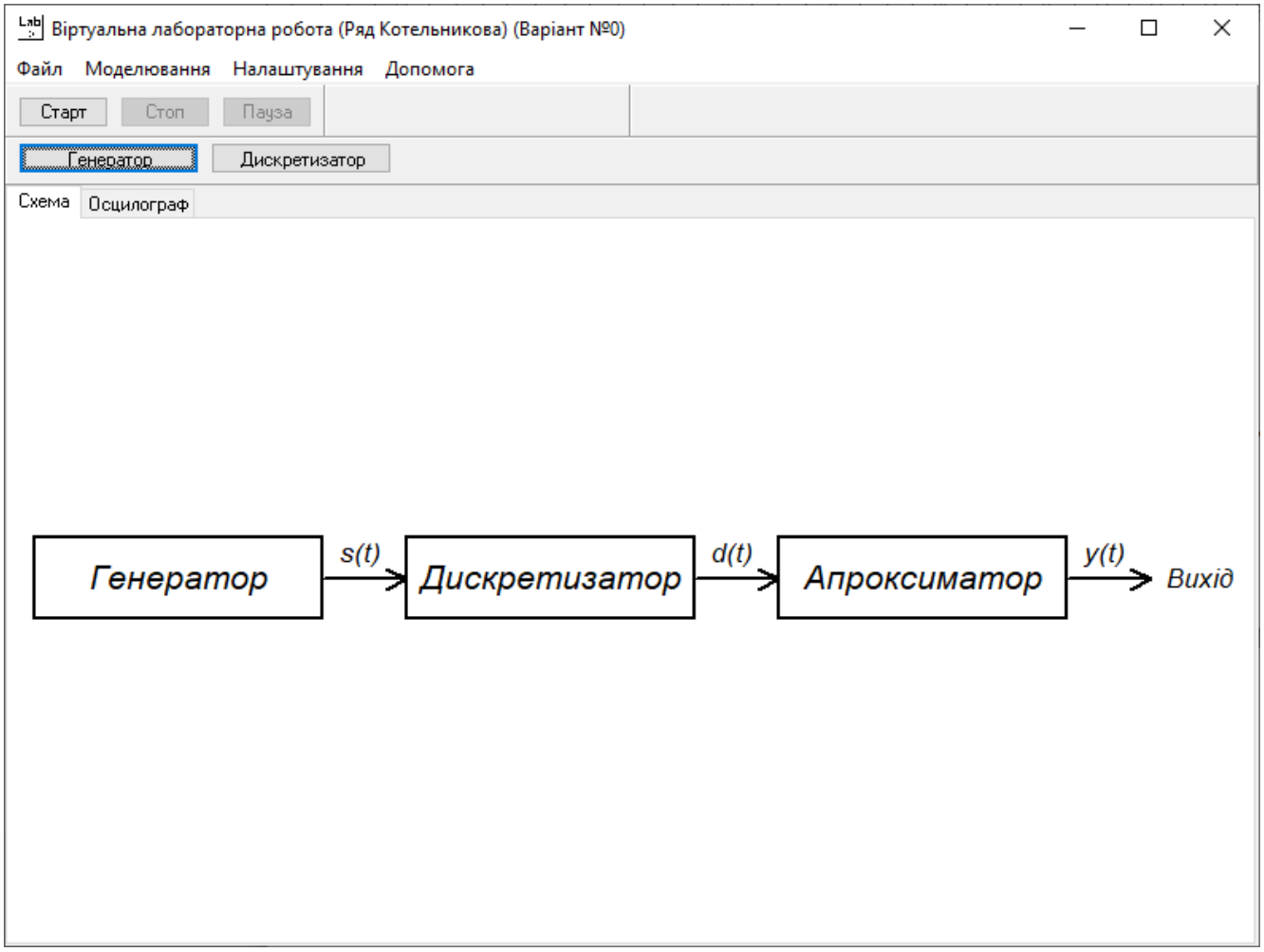


Рисунок 3.2 – Зовнішній вигляд лабораторного макету

Лабораторний макет складається із трьох функціональних вузлів: генератору тестового сигналу *s(t)* (в макеті він позначений як Генератор), дискретизатора, призначеного для формування відліків *d(t)*, та апроксиматора, що по відлікам забезпечує відновлення первинного сигналу.

Генератор тестового сигналу складається із двох синусоїдальних генераторів, кожен з яких може генерувати синусоїдальну напругу з амплітудою від 0 В до 1 В та частотою від 1 Гц до 10 кГц. Користувач може самостійно встановити дані параметри. Для цього достатньо натиснути на кнопку «Генератор», після цього відкриється діалогове вікно редагування параметрів генератора (рис. 3.3).

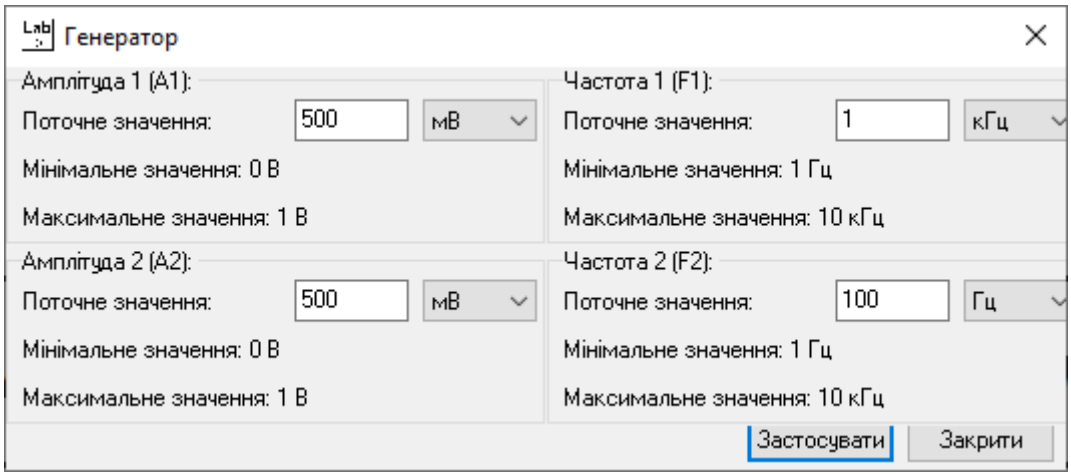


Рисунок 3.3 – Вікно редагування параметрів генератора

Вихідна напруга генератора *s(t)* э сумою двох синусоїдальних сигналів, яка обчислюється за формулою:

$$s(t) = A_1 \sin 2\pi f_1 t + A_2 \sin 2\pi f_2 t; \qquad (3.1)$$

де t – поточний час моделювання; A_1, A_2, f_1, f_2 – відповідно, амплітуди та частоти першого та другого синусоїдального генераторів.

Один із варіантів вихідного сигналу генератора *s(t)* показаний на рис. 3.4.

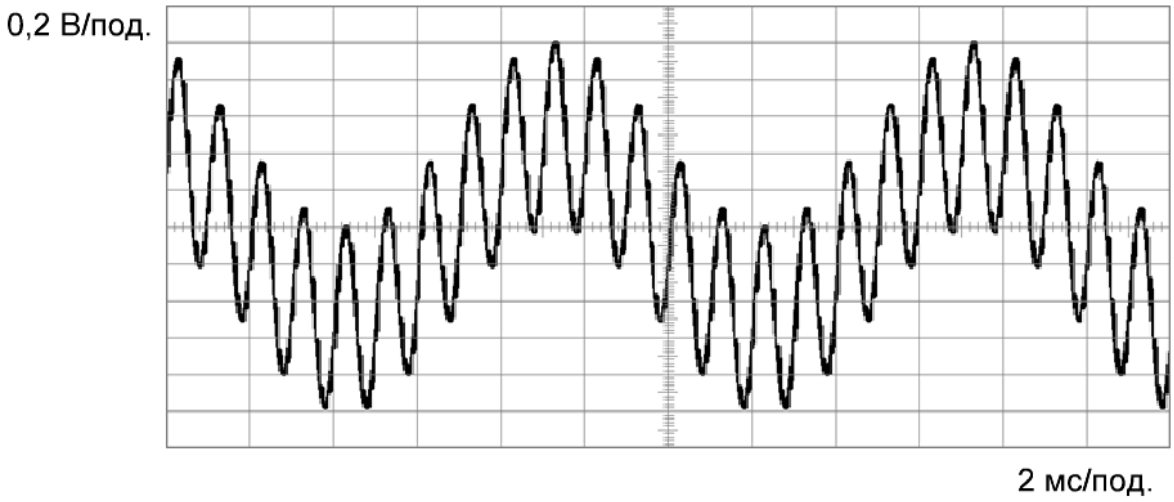


Рисунок 3.4 – Форма вихідного сигналу генератора
(A_1 = 0,5 В, f_1 = 1 кГц, A_2 = 0,5 В, f_2 = 100 Гц)

Дискретизатор на основі сигналу *s(t)* формує відліки *d(t)* з певною частотою дискретизації $f_д$. Сигнал *d(t)* описується формулою:

$$d(t)=\begin{cases} s(t), & \text{якщо } \left\{\dfrac{t}{T_д}\right\}=0 \\ 0, & \text{якщо } \left\{\dfrac{t}{T_д}\right\}\neq 0 \end{cases}; \tag{3.2}$$

де $T_д = 1/f_д$ – період дискретизації.

Частота дискретизації $f_д$ може змінюватися користувачем в діапазоні від 1 Гц до 10 кГц. Для зміни частоти дискретизації необхідно натиснути на кнопку «Дискретизатор», після цього відкриється діалогове вікно редагування параметрів дискретизатора (рис. 3.5).

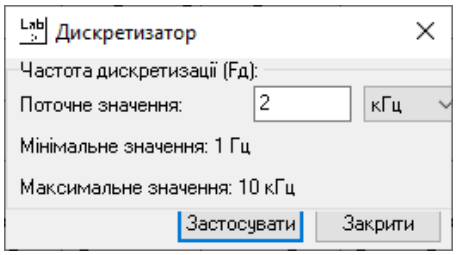


Рисунок 3.5 – Вікно редагування параметрів дискретизатора

Один із варіантів вихідного сигналу дискретизатора *d(t)* показаний на рис. 3.6.

Вхідним сигналом для апроксиматора є вихідний сигнал дискретизатора *d(t)*, який виконує відновлення вхідного сигналу за формулою:

$$y(t)=\sum_{k=-5}^{5} d_k \frac{\sin\left(\dfrac{\pi}{T_д}\left[t-\left(N_s+k\right)T_д\right]\right)}{\dfrac{\pi}{T_д}\left[t-\left(N_s+k\right)T_д\right]}; \tag{3.3}$$

де N_s – абсолютний (від початку моделювання) номер відліку.

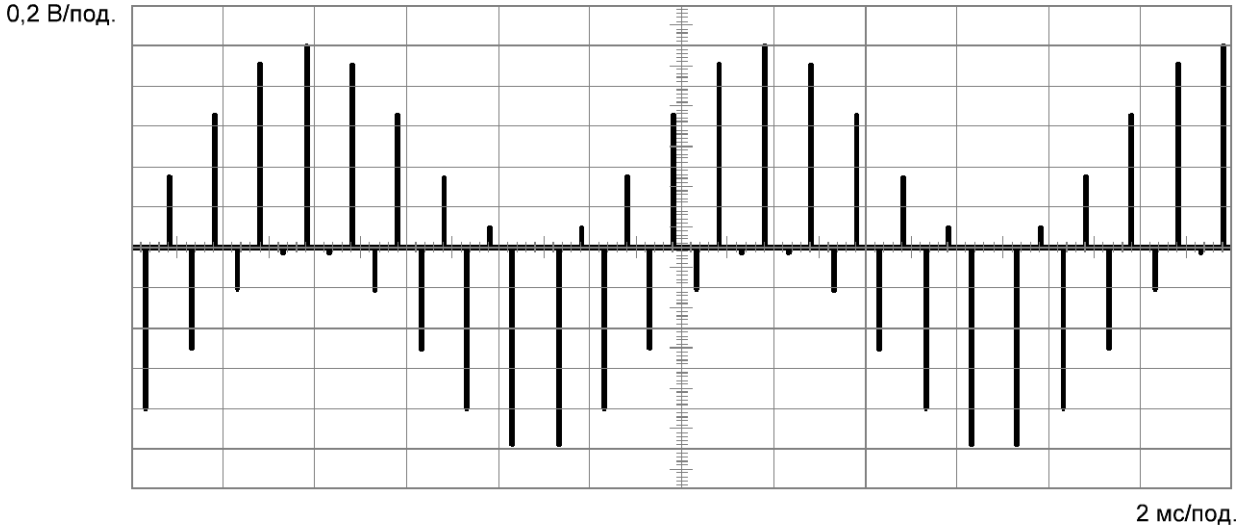


Рисунок 3.6 – Форма сигналу на виході дискретизатора
(A_1 = 0,5 В, f_1 = 1 кГц, A_2 = 0,5 В, f_2 = 100 Гц, $f_д$ = 2 кГц)

Як видно із формули (3.3) у відновленні сигналу приймають участь 11 відліків дискретизованого сигналу $d_k(t)$ (k = –5…5).

Один із варіантів сигналу *y(t)* на виході апроскиматора показаний на рис. 3.7.

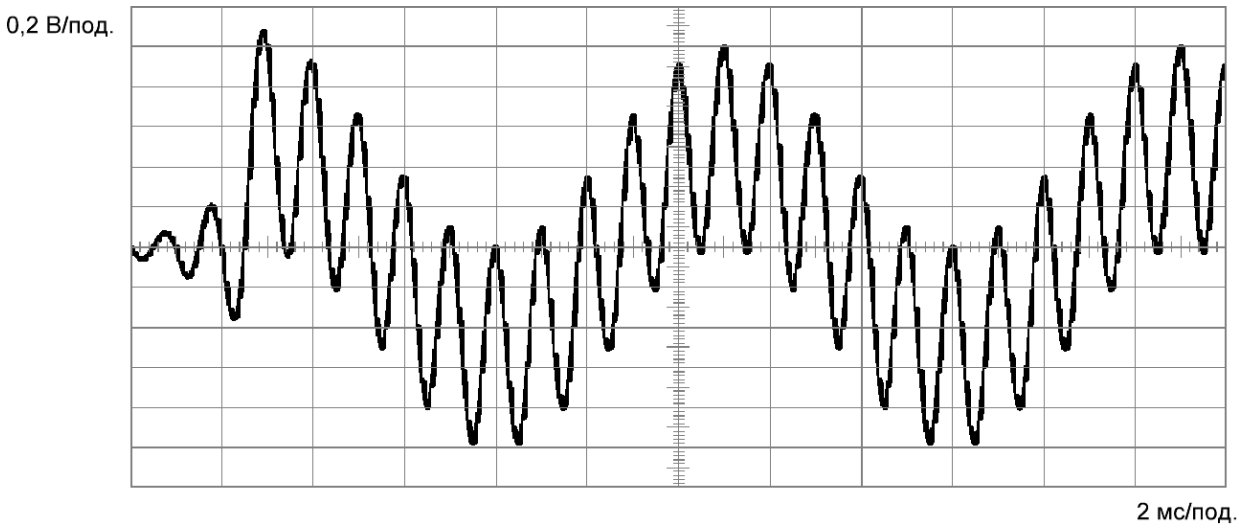


Рисунок 3.7 – Форма сигналу на виході апроксиматора
(A_1 = 0,5 В, f_1 = 1 кГц, A_2 = 0,5 В, f_2 = 100 Гц, $f_д$ = 2 кГц)

Сигнали *s(t)*, *d(t)* та *y(t)* можна спостерігати за допомогою інтегрованого у ядро програми Labs багатоканального осцилографу. У даному макеті осцилограф має три незалежні канали, кожен з яких можна підключити до буть якого із сигналів (рис. 3.8). За замовчуванням перший канал осцилографа підключається до сигналу s(t), другий – до сигналу *d(t)*, третій – до сигналу *y(t)*.

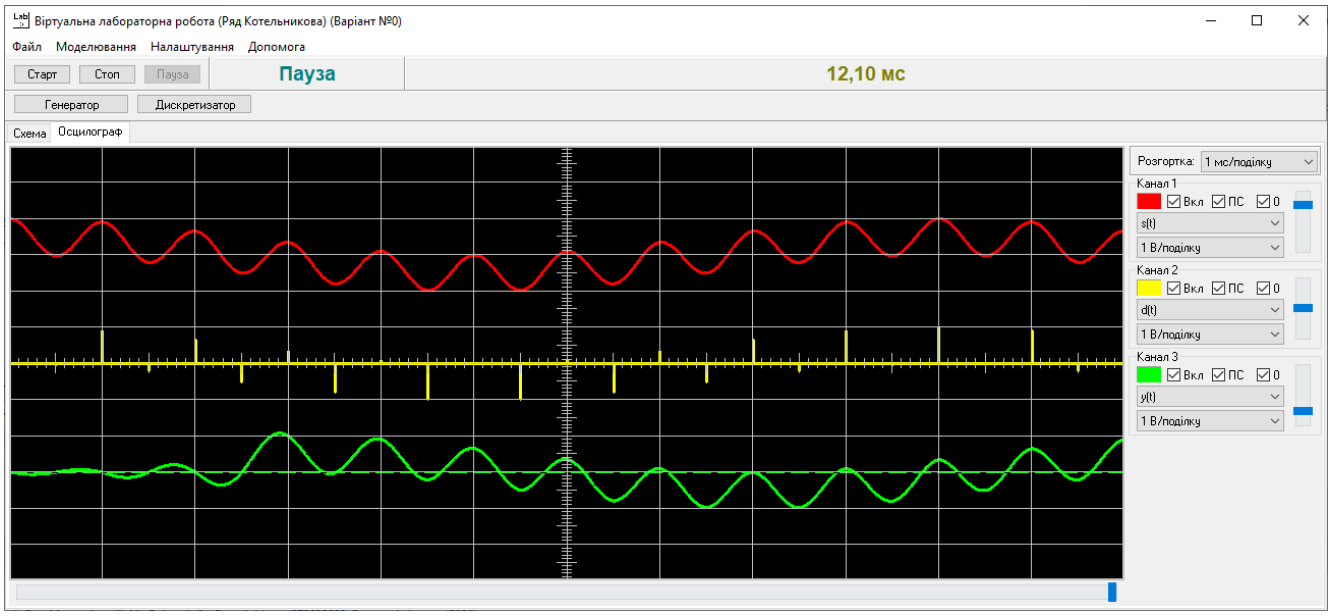


Рисунок 3.8 – Зовнішній вигляд осцилографа

Кожен із трьох каналів осцилографа має можливість:

– вибору точки підключення (*s(t)*, *d(t)* або *y(t)*);

– вибору масштабу відображення сигналу;

– вибору кольору променю;

– зміщення променю по вертикалі;

– відключення променю;

– видалення постійної складової сигналу, що відображається;

– демонстрації лінії, що відповідає нульовому рівню сигналу.

Крім того осцилограф забезпечує зміну масштабу по горизонталі та групового горизонтального зміщення усіх трьох променів.

Для початку моделювання необхідно натиснути кнопку «Старт», після чого у верхній частині лабораторного макета з’явиться напис «Моделювання» разом із

значенням промодельованого часу. Призупинити моделювання можна натиснувши на кнопку «Пауза», а натискання на кнопку «Стоп» призведе до повної зупинки моделювання. Різниця між паузою та зупинкою моделювання полягає у тому, що після паузи моделювання буде продовжено із тими самими значеннями змінних, у той час як після зупинки моделювання та подальшого його поновлення усі внутрішні змінні програми (окрім налаштувань користувача), у тому числі і час моделювання будуть ініціалізовані початковими значеннями.

3.2 Програмна частина макету

Більша частина програмного коду у тому числі інтерфейсна частина, заготовки для редагування компонентів, віртуальний осцилограф та внутрішні механізми вже реалізовані в ядрі програми Labs. Створення програмного модуля зводилось до налаштувань параметрів ядра та створення математичної моделі, призначеної для відновлення сигналів за допомогою ряду Котельникова.

Лабораторний макет створений в інтегрованому середовищі розробки Delphi 11.4. Вихідний код лабораторного макеті наведений у Додатку Б. Перелік підпрограм лабораторного макету наведений у таблиці 3.1.

Таблиця 3.1 – Підпрограми макету

Назва підпрограми	**Призначення**
SetLayoutGlobals	Ініціалізація глобальних змінних макету
GetLayoutInfo	Налаштування опису макету (назва, предмет тощо)
SetSchemInfo	Налаштування параметрів досліджуваної схеми
SetComponentInfo	Налаштування елементів досліджуваної схеми
SetComponentParamInfo	Налаштування параметрів елементів досліджуваної схеми
SetOscillographInfo	Налаштування осцилографу
SetOscillographChannelInfo	Налаштування каналів осцилографу
SetOscillographChannelPointInfo	Налаштування точок підключення осцилографу до елементів схеми
PaintSchem_0	Підпрограма у якій створюється графічне зображення конкретної досліджуваної схеми
PaintSchem	Підпрограма для малювання графічної частини схеми
ResetSchem	Перезавантаження схеми
CalcNextIteration	Математична модель

Основними програмними об’єктами є генератор, дикретизатор та апроксиматор. Ці програмні об’єкти реалізовані у вигляді глобальних змінних, відповідно, G1, D1, та R1, що складаються з кількох полів (змінні типу «запис»). Призначення полів наведено у таблицях 3.2 – 3.4.

Таблиця 3.2 – Опис полів глобальної змінної G1 (генератор)

Поле (змінна)	**Тип**	**Призначення**
Info	PComponentInfo	Покажчик на структуру, де зберігаються зовнішні параметри об’єкта
A1	PDouble	Покажчик на значення амплітуди першого синусоїдального генератора
A2	PDouble	Покажчик на значення амплітуди другого синусоїдального генератора
F1	PDouble	Покажчик на значення частоти першого синусоїдального генератора
F2	PDouble	Покажчик на значення частоти другого синусоїдального генератора
U	Double	Вихідна напруга генератора

Таблиця 3.3 – Опис полів глобальної змінної D1 (дискретизатор)

Поле (змінна)	**Тип**	**Призначення**
Info	PComponentInfo	Покажчик на структуру, де зберігаються зовнішні параметри об’єкта
Fs	PDouble	Покажчик на значення частоти дискретизації
LastSampleTime	Double	Час формування останнього відліку
U	Double	Вихідна напруга дискретизатора

Таблиця 3.4 – Опис полів глобальної змінної R1 (апроксиматор)

Поле (змінна)	**Тип**	**Призначення**
Info	PComponentInfo	Покажчик на структуру, де зберігаються зовнішні параметри об’єкта
SampleCount	Double	Кількість відліків
Us	array of Double	Значення відліків, що приймають участь у відновлені сигналу
U	Double	Вихідна напруга апроксиматора

3.3 Загальний алгоритм моделювання

Спрощений алгоритм проведення моделювання показаний на рис. 3.9. Після натискання на кнопку «Старт» (блок 1) відбувається початкова ініціалізація

змінних математичної моделі (блоки 2, 3). На даному етапі ініціалізуються як внутрішні змінні ядра (блок 2), так і глобальні змінні лабораторного макету (блок 3). Для цього у лабораторного макеті використовується спеціальна підпрограма ResetSchem (блок 3).

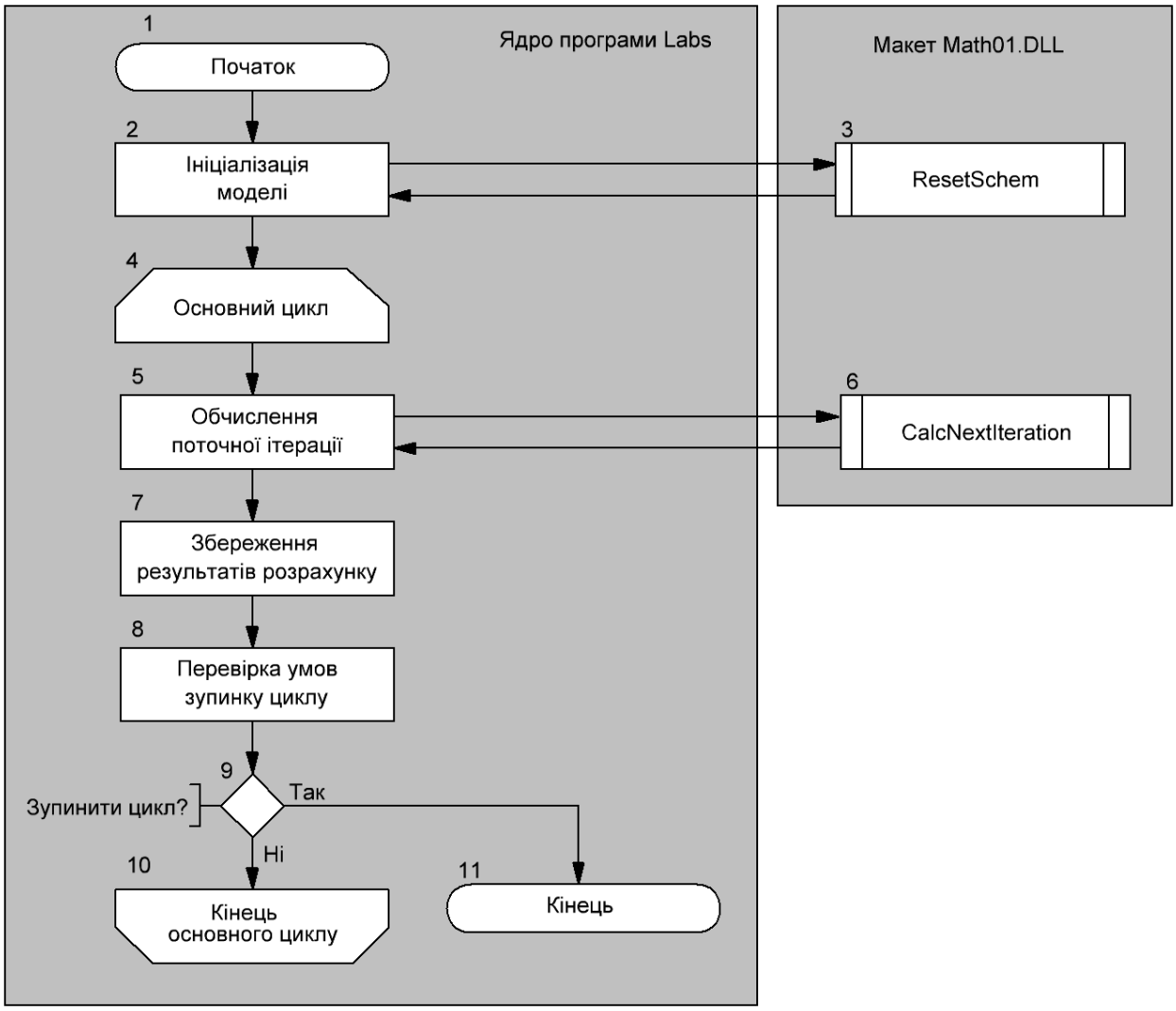


Рисунок 3.9 – Спрощений алгоритм проведення моделювання

Після ініціалізації починає виконуватися нескінченний цикл, обмежений блоками 2 та 10. Для того, щоб користувач під час виконання розрахунків мав можливість втручатися у процес моделювання цей цикл виконується в окремому потоці. На кожній ітерації циклу спочатку обчислюється поточна ітерація розрахунку (блоки 5, 6), причому головні обчислення відбуваються у підпрограмі CalcNextIteration (блок 6), розташовані й у файлі лабораторного макету.

Обчислення поточної ітерації полягає у встановленні значень певних глобальних змінних, розташованих переважно у файлі макету. Після цього у блоці 7 відбувається збереження результатів розрахунку у контрольно вимірювальних приладах лабораторного макету (осцилографі, мультіметрах, аналізаторах спектр тощо) для подальшого відображення в інтерфейсній частині лабораторії.

Заключним етапом основного циклу є перевірка результатів розрахунку на предмет подальшого їх продовження. Припинення розрахунків можу виникнути як за бажанням користувача (натиснена кнопка «Стоп»), або у випадку коли моделювання закінчилося (для цього у підпрограму CalcNextIteration передається спеціальна змінна StopSumilation), або у виникненні непередбачуваної помилки, наприклад, виявлення некоректної комбінації параметрів моделювання.

Перевірка необхідності подальшого моделювання виконується у блоці 9, після якого виконується або наступна ітерація основного циклу (блок 10), або моделювання завершується (блок 11).

3.4 Алгоритм розрахунку

Спрощений алгоритм розрахунку (алгоритм програми CalcNextIteration) наведено на рис. 3.10. Після входу у підпрограму (блок 1) на основі поточного часу моделювання (змінна CurTime) відбувається обчислення вихідної напруги генератора сигналів G1 (блок 2). Після цього у блоці 3 виконується обчислення часу, який минув після останньої дискретизації (підпрограма CalcNextIteration викликається ядром із інтервалом приблизно 0,1 мкс умовного часу). Якщо цей час дорівнює або перевищує тривалість періоду дискретизації (перевірка відбувається у блоці 4), то відбувається формування нового відліку, а якщо ні, то вихідна напруга дискретизатора дорівнює нулю, і програма одразу переходить до блоків відновлення сигналу.

Формування вихідної напруги дискретизатора (копіювання вихідної напруги генератора) відбувається у блоці 5. Після цього у блоках 6 – 8 відбувається циклічне зміщення відліків у внутрішньому буфері апроксиматора.

Кінцевим кроком роботи дискретизатора є запис поточного відліку в останню комірку буферу апроксиматора (блок 9).

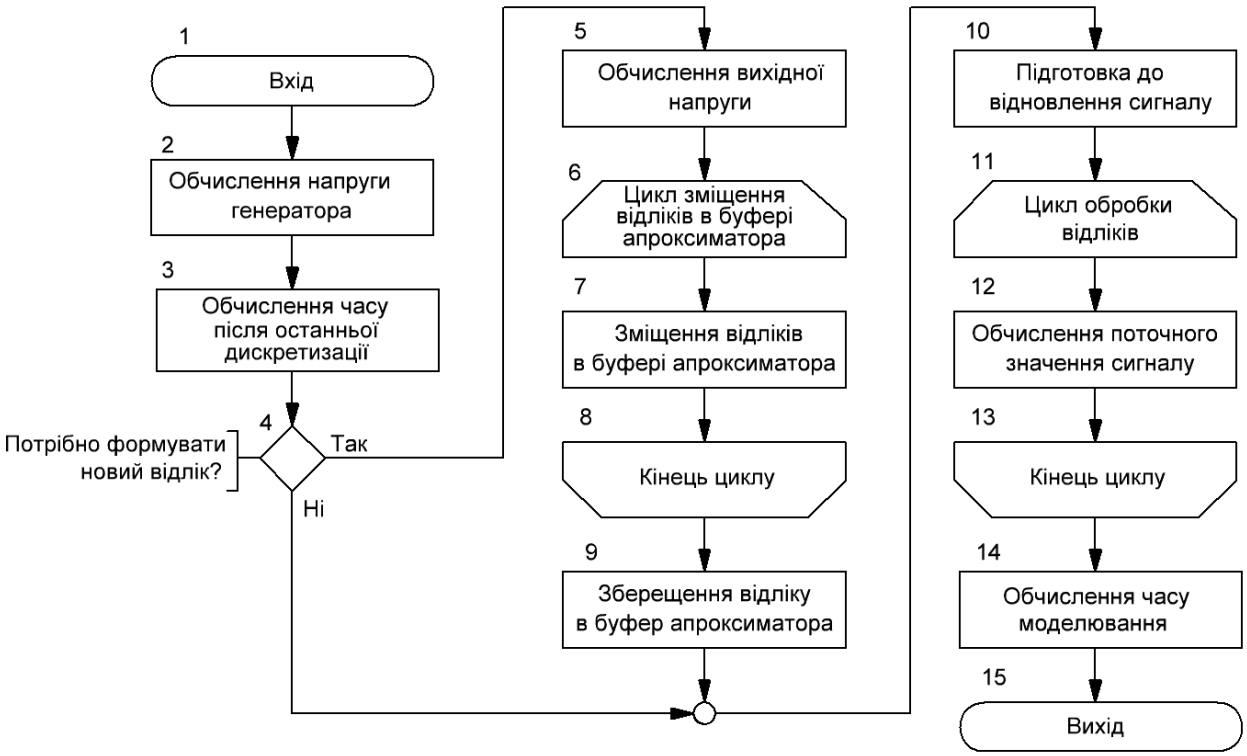


Рисунок 3.10 – Спрощений алгоритм підпрограми CalcNextIteration

Відновлення сигналу відбувається у блоках 10 – 13. На цьому етапі оброблюються значення відліків у внутрішньому буфері та обчислення вихідної напруги апроксиматора. Перед початком обробки у блоці 10 виконуються підготовчі операції (обнулення підсумкової суми).

Кінцевим етапом розрахунку є зміна абсолютного часу моделювання, яка виконується у блоці 14.

3.5 Аналіз результатів моделювання

Результати відновлення тестового сигналу приведені на рис. 3.11. Як видно з рисунку, при частоті дискретизації $f_д$ = 2 кГц, що вдвічі перевищує максимальну частоту генератора (f_1 = 1 кГц) відбувається якісне відновлення первинного сигналу як за амплітудою, так і за формою. При цьому перші 4 мс відновлений

сигнал значно відрізняється від первинного, оскільки у цей час в буфері апроксиматора ще недостатньо відліків для його відновлення. Крім того, спостерігається відставання у часі між первинним та відновленим сигналом на величину приблизно 3 мс, що є цілком природним, оскільки для якісного відновлення сигналу апроксиматор повинен мати певну інформацію як про попередні, так і про наступні відліки.

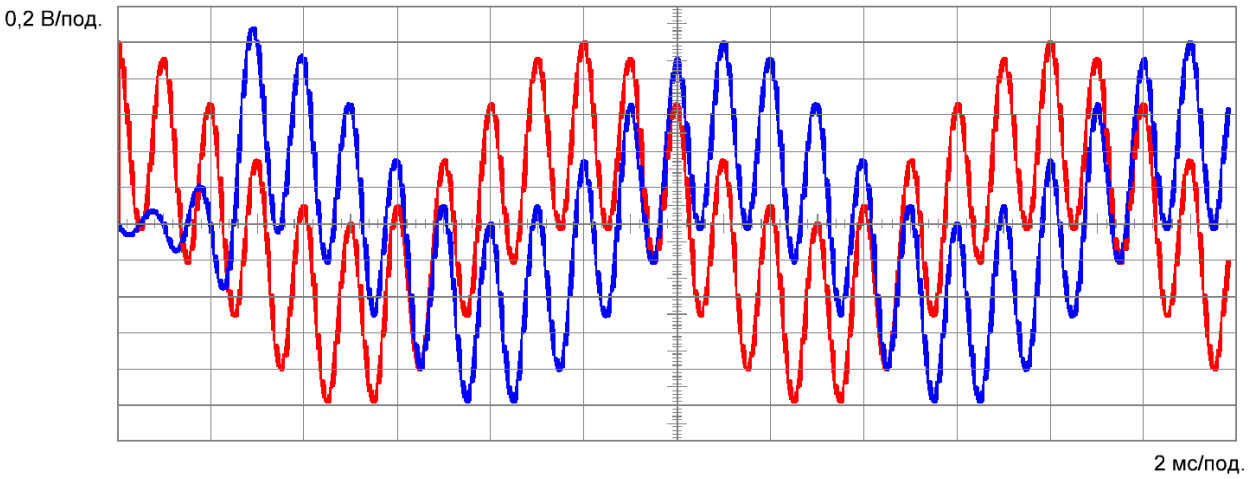


Рисунок 3.11 – Результати відновлення сигналу
(A_1 = 0,5 В, f_1 = 1 кГц, A_2 = 0,5 В, f_2 = 100 Гц, $f_д$ = 2 кГц),
первинний сигнал – червона лінія, відновлений сигнал – синя лінія

Для перевірки працездатності створеного програмного забезпечення проведемо відновлення сигналу за умов, що не відповідають теоремі Котельника. Для цього зменшим частоту дискретизації до значення $f_д$ = 1,5 кГц, що менше ніж подвійна максимальна частота сигналу (f_1 = 1 кГц) (рис. 3.12). Як видно з рисунку, у цьому випадку відновлений сигнал значно відрізняється від сигналу, сформованого генератором. Спостерігаються як частотні, так і фазові спотворення первинного сигналу, що свідчить про те, що частота дискретизації недостатня для повноцінного відтворення сигналу.

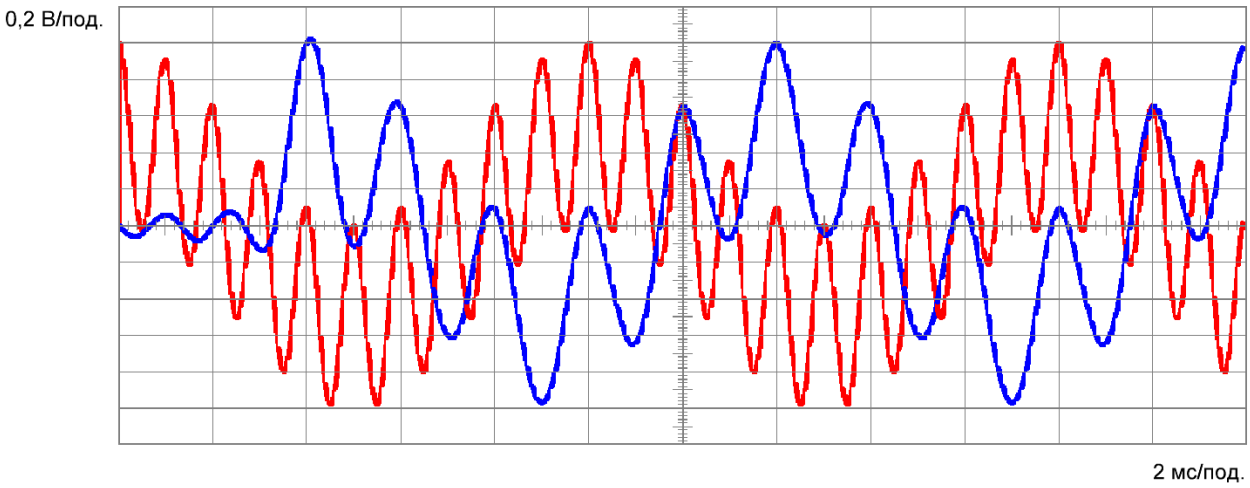


Рисунок 3.12 – Результати відновлення сигналу
(A_1 = 0,5 В, f_1 = 1 кГц, A_2 = 0,5 В, f_2 = 100 Гц, $f_д$ = 1,5 кГц),
первинний сигнал – червона лінія, відновлений сигнал – синя лінія

Збільшення частоти дискретизації до $f_д$ = 10 кГц призводить до зменшення фазового зсуву піж первинним та поновленим сигналами (приблизно до 0,2 мс) (рис. 3.13) Крім того у даному випадку зменшується час, необхідний для виходу апроксиматора на робочий режим, однак істотної зміни якості відновлення сигналу при цьому не спостерігається.

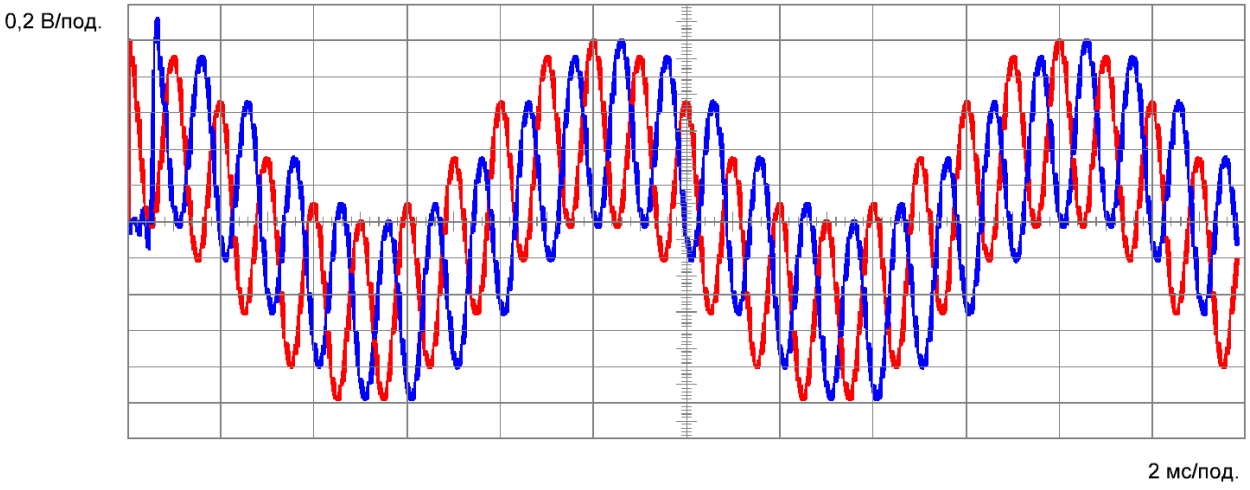


Рисунок 3.13 – Результати відновлення сигналу
(A_1 = 0,5 В, f_1 = 1 кГц, A_2 = 0,5 В, f_2 = 100 Гц, $f_д$ = 10 кГц),
первинний сигнал – червона лінія, відновлений сигнал – синя лінія

Слід зауважити, що умова $f_д \geq 2f_{max}$ повинна виконуватися безумовно, оскільки навіть невелике зменшення частоти дискретизації (наприклад, до $f_д$ = 1,9 кГц) вже призводить до помітних спотворень сигналу (рис. 3.14). Як видно з рисунка, в області негативних значень амплітуда відновленого сигналу на 0,2 В менша за амплітуду первинного сигналу, у той час як в області позитивних значень вони практично однакові. Таким чином, у відновленому сигналі з'являється додаткова постійна складова, що свідчить про наявність як амплітудних так і частотних спотворень.

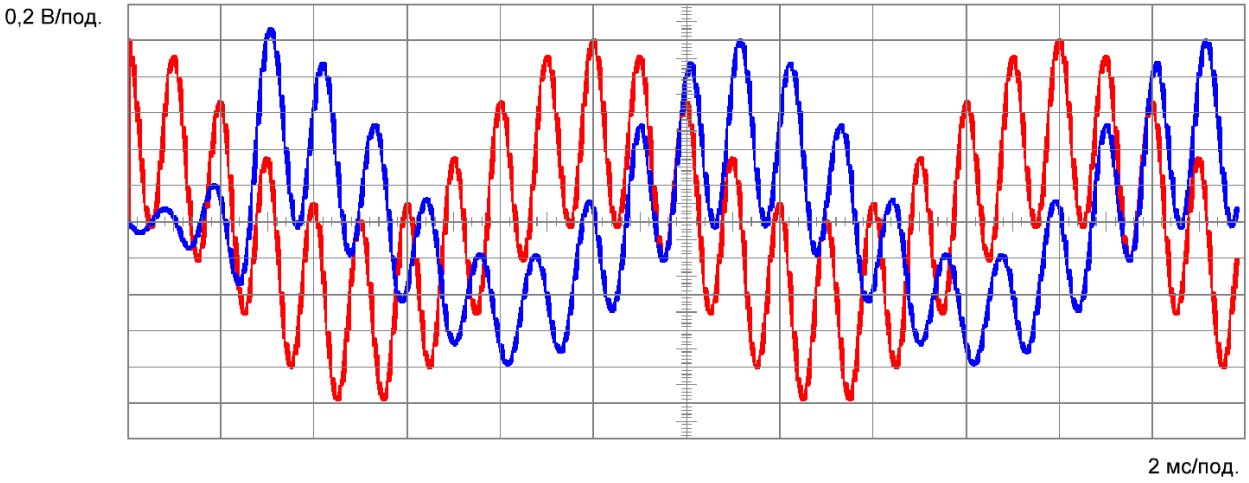


Рисунок 3.14 – Результати відновлення сигналу
(A_1 = 0,5 В, f_1 = 1 кГц, A_2 = 0,5 В, f_2 = 100 Гц, $f_д$ = 1,9 кГц),
первинний сигнал – червона лінія, відновлений сигнал – синя лінія

3.6 Висновки за розділом

Розроблене програмне забезпечення показало повну відповідність результатів моделювання теоретичним розрахункам. Результати моделювання показали, що первинний сигнал можна повністю відновити за допомогою ряду Котельникова за умови, що частота дискретизації буде щонайменше удвічі більша за максимальну частоту первинного сигналу.

ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

1. Проведено аналіз методів відновлення інформаційного трафіку, показано доцільність відновлення трафіку різними методами апроксимації, а саме:

– за допомогою ряду Котельникова;

– за допомогою сплайн-апроксимацій;

– за допомогою вейвлет-функцій.

2. Проведено аналіз методів практичної реалізації математичних моделей, обґрунтовано використання технології динамічно-зв’язаних бібліотек та універсальної віртуальної лабораторії.

3. Створено новий програмний модуль для досліджень методів відновлення трафіку за допомогою ряду Котельникова. Проведено тестування його працездатності, яке підвердило усі теоретичні розрахунки.

4. Результати роботи доцільно рекомендувати для практичного застосування у наступних напрямках:

– при проєктуванні пристроїв для передачі інформації, інфокомунікаційної техніки та інших застосунків, в яких використовується відновлення інфокомунікаційного трафіку;

– під час підготовки фахівців у галузях інформаційних технологій, інфокомунікації та радіотехніки, у тому числі і при вивченні методів передавання інформації за допомогою цифрових систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Donovan J. Building the Network of the Future: Getting Smarter, Faster, and More Flexible with a Software Centric Approach. Donovan John, Prabhu Krish (eds.). Chapman and Hall/CRC, 2017.

2. Ioan-Sorin Comşa, Ramona Trestian Next-Generation Wireless Networks Meet Advanced Machine Learning Applications. IGI Global. 2019. 356p. ISBN:9781522574590, 152257459X.

3. Sheng Zhong, Hong Zhong, Xinyi Huang, Panlong Yang, Jin Shi, Lei Xie, Kun Wang Security and Privacy for Next-Generation Wireless Networks. Springer International Publishing. 2019. 183p. ISBN:9783030011512, 3030011518.

4. Gan Zheng, Ioannis Krikidis Advanced Relay Technologies in Next Generation Wireless Communications. Institution of Engineering and Technology. 2016. 536p. ISBN:9781785610035, 1785610031.

5. Strelkovskaya I., Solovskaya I., Strelkovska J., «Spline-Approximation and SplineExtrapolation Methods in Telecommunication Problems», Current Trends in Communication and Information Technologies. IPF 2020. Lecture Notes in Networks and Systems, vol 212, Springer, Cham, 2021, P. 3-22. URL: https://doi.org/10.1007/978-3-030-76343-5_1.

6. Strelkovskaya I.V., Bukhan D.Yu., «Restoration of continuous signals based on the Kalman-Bucy filter and cubic splines», Radio engineering: All-Ukr. Inter. scien. tech. col., Vol. 156, P. 61-63.

7. Strelkovskaya I., Solovskaya I., Severin N., Paskalenko S., «Spline approximation based restoration for self-similar traffic», Eastern-European Journal of Enterprise Technologies, 3/4 (87), P. 45-50. URL: https://doi.org/10.15587/1729-4061.2017.102999.

8. Стрелковская И.В., Лысюк Е.В, Золотухин Р.В. Сравнительный анализ востановления непрерывных сигналов рядом Котельникова и сплайн-функциями. *Восточно-европейский журнал передовых технологий*. 2013, Вып. 2/9. С. 12-15.

9. Стрелковская И.В., Лысюк Е.В., Золотухин Р.В Восстановление непрерывных сигналов на базе вейвлет-функций. *Первая Международная научно-практическая конференция «Проблемы инфокомуникаций. Наука и технологии» (PICS&T-2013): Сборник научных трудов, Харьков, 9-11 октября 2013*. ХНУРЭ, С.223-225.

10. Стрелковская И.В., Омельченко О.П., Зубенко М.Г. Визначення характеристик якості QoS трафіку ТМЗК за допомогою апроксимації Вейвлет-функцією Хаара. *Сьома міжнародна наук.-практ. конф. «Інфокомунікації – сучасність та майбутнє»: Збірник тез., 26-27 жовтня 2017 р., Ч.І: тези доп.* ОНАЗ ім. О.С. Попова, Одеса, 2017. С. 72-74.

11. Стрелковская И.В., Бухан Д.Ю. Восстановление непрерывных сигналов на основе ряда Котельникова и кубических сплайнов. *Радиотехника: Всеукр. межвед. науч.-техн. сб*. Харьков: ХНУРЭ, 2007. № 4. С. 181-185.

12. Стрелковская И.В., Лысюк Е.В., Золотухин Р.В Сравнительный анализ восстановления непрерывных сигналов рядом Котельникова и сплайн-функциями. Восточно-Европейский журнал передовых технологий. 2013. Т.2, № 9(62). С.12-15 https://doi.org/10.15587/1729-4061.2013.12437

13. Marks, II, R.J. "Chapters 5–8". Handbook of Fourier Analysis and Its Applications. Oxford University Press, 2009. ISBN 978-0-19-804430-7

14. Геранін, В. О. Теорія вейвлетів з елементами фрактального аналізу [Текст] / В. О. Геранін, Л. Д. Писаренко, Я. Я. Рушицький.: Науково-методичне видання. – Київ: ВПФ УкрІНТЕІ, 2002. 364 с.

15. Farzin Asadi Essential Circuit Analysis using NI Multisim and MATLAB. *Springer Cham* 2022. 769p. ISBN: 978-3-030-89849-6

16. Farzin Asadi Essential Circuit Analysis Using Proteus. *Springer Singapore* 2023. 661. DOI: 10.1007/978-981-19-4353-9

17. Colin May Passive Circuit Analysis with LTspice®: An Interactive Approach. *Springer International Publishing* 2021. 763p. ISBN 3030383067, 9783030383060

18. Dr A Chrispin Jiji Analog Integrated Circuits with PSPICE. *BFC Publications*, 2021. 153p. ISBN: 9789355090591, 9355090595.

19. Онлайн-симулятор електронних кіл eDesignSuite. URL: https://www.st.com/content/st_com/en/support/resources/edesign.html

20. SpeedFit Design Simulator. URL: https://www.wolfspeed.com/tools-and-support/power/speedfit/

21. Abhishek Jadhav Simulation Tools for Power Electronics Applications. *Power electronics news*. 2023. URL: https://www.powerelectronicsnews.com/simulation-tools-for-power-electronics-applications/

22. Elite Power Simulator. URL: https://www.onsemi.com/design/tools-software/elite-power-simulator

23. Пакет прикладних програм для числового аналізу MATLAB. URL: https://www.mathworks.com/products/matlab.html

24. Система комп'ютерної алгебри Mathcad. URL: https://mathcad.com.ua/

25. Платформа для візуальної мови програмування LabVIEW. URL: https://www.ni.com/en-us/shop/labview.html

26. Primož Gabrijelčič Mastering Delphi Programming: A Complete Reference Guide. *Packt Publishing*, 2019. 674p. ISBN: 9781838983918, 1838983910

27. Русу А.П. Использование динамически подключаемых библиотек для моделирования электрических процессов радиотехнических устройств. *Наукові праці ОНАЗ ім. О.С. Попова*. 2010. №1. С. 143–147.

28. Русу А.П. Динамически подключаемая библиотека для расчета характеристик импульсных преобразователей постоянного напряжения с ШИМ-регулированием. *Материалы 64-ой НТК профессорско-преподавательского состава научных работников, аспирантов и студентов (г. Одесса, 1 – 4 декабря 2009 г.)*

29. Русу О. П. Універсальна віртуальна лабораторія для традиційної та дистанційної форм навчання / О. П. Русу // Гуманітарний і інноваційний ракурс професійної майстерності: пошуки молодих вчених : матер. VIII Всеукр. наук.-практич. конф. студентів, аспірантів та молодих учених, (18 листоп. 2022 р., м. Одеса). – Львів – Торунь : Liha-Pres, 2022. – С. 385-388.

30. Русу О. П. Віртуальна лабораторія для програмування мікроконтролерів / О. П. Русу, А. С. Маркітан, Н. Р. Султанзаде // Чорноморські наукові студії : IX Всеукраїнської мультидисциплінарної конференції, (12 трав. 2023 року, м. Одеса). – Львів – Торунь : Liha-Pres, 2023. – С. 214-216.

ДОДАТОК А

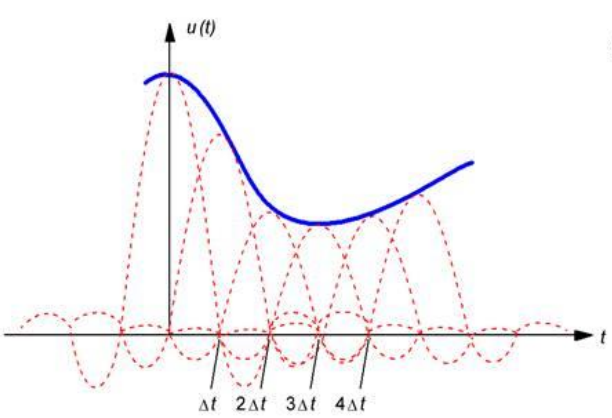
ПЕРЕЛІК КОПІЙ ДЕМОНСТРАЦІЙНОГО МАТЕРІАЛУ

Слайд 1 – Титульний слайд

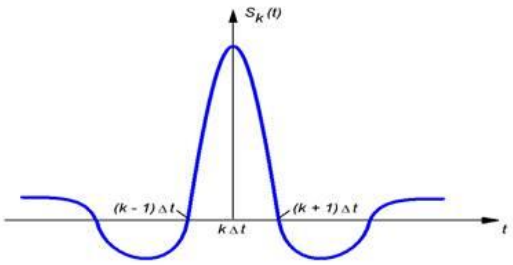
Слайд 2 – Основні характеристики роботи

Слайд 3 – Принцип відтворення сигналу за допомогою ряду Котельникова

Принцип відтворення сигналу за допомогою ряду Котельникова



$$u(t) = \sum_{k=-\infty}^{\infty} u(k\Delta t)\frac{\sin \omega_{\max}(t - k\Delta t)}{\omega_{\max}(t - k\Delta t)}$$



3

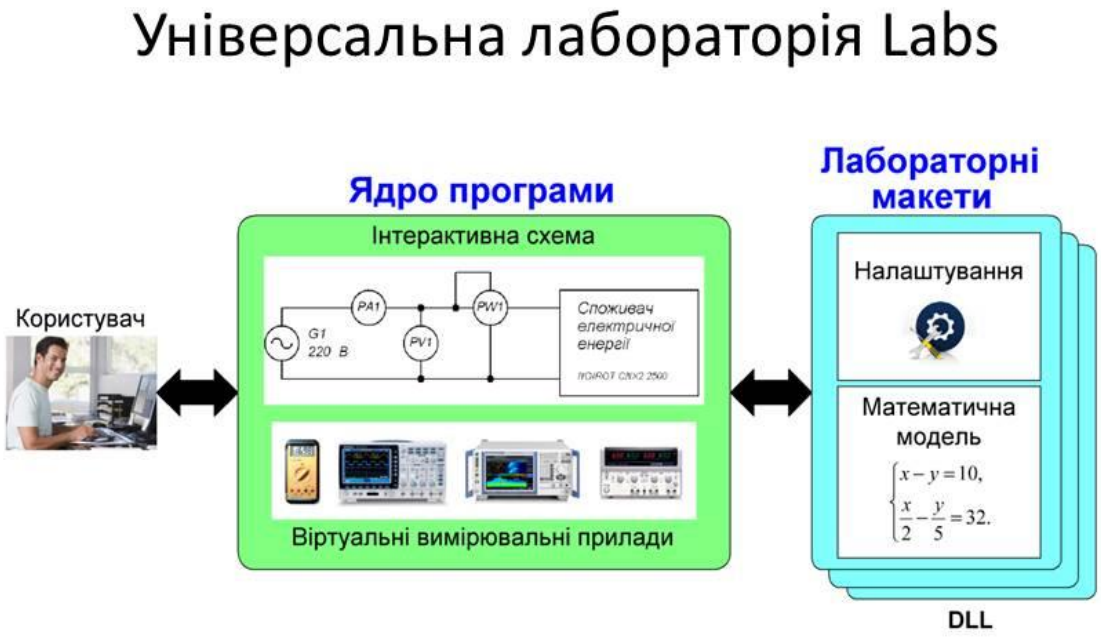
Слайд 4 – Порівняльний аналіз способів реалізації математичної моделі

Порівняльний аналіз способів реалізації математичної моделі

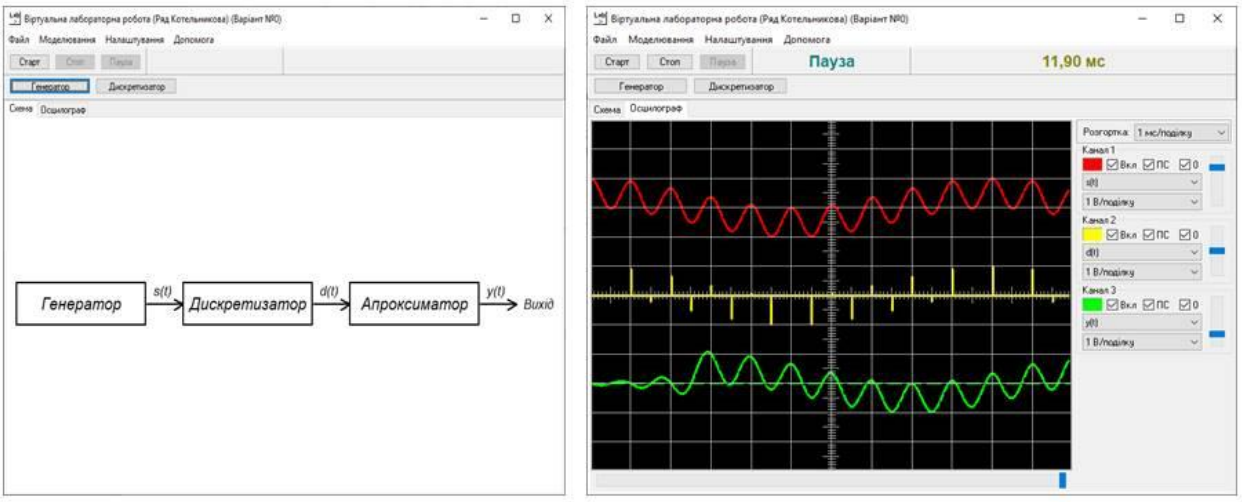
Критерій оцінювання	Універсальна обчислювальна система	Самостійний програмний застосунок	Динамічно-зв'язана бібліотека
Швидкість реалізації математичної моделі	Висока (+)	Низька (–)	Висока (+)
Рівень знань в галузі програмування	Не потребує (+)	Високий (–)	Базовий (+)
Швидкість проведення обчислень	Низька (–)	Висока (+)	Висока (+)
Складність інтеграції з іншими програмними застосунками	Складна (–)	Середня (±)	Проста (+)
Можливість самостійного використання	Складна (–)	Є самостійним програмним застосунком (+)	Неможливо (–)

4

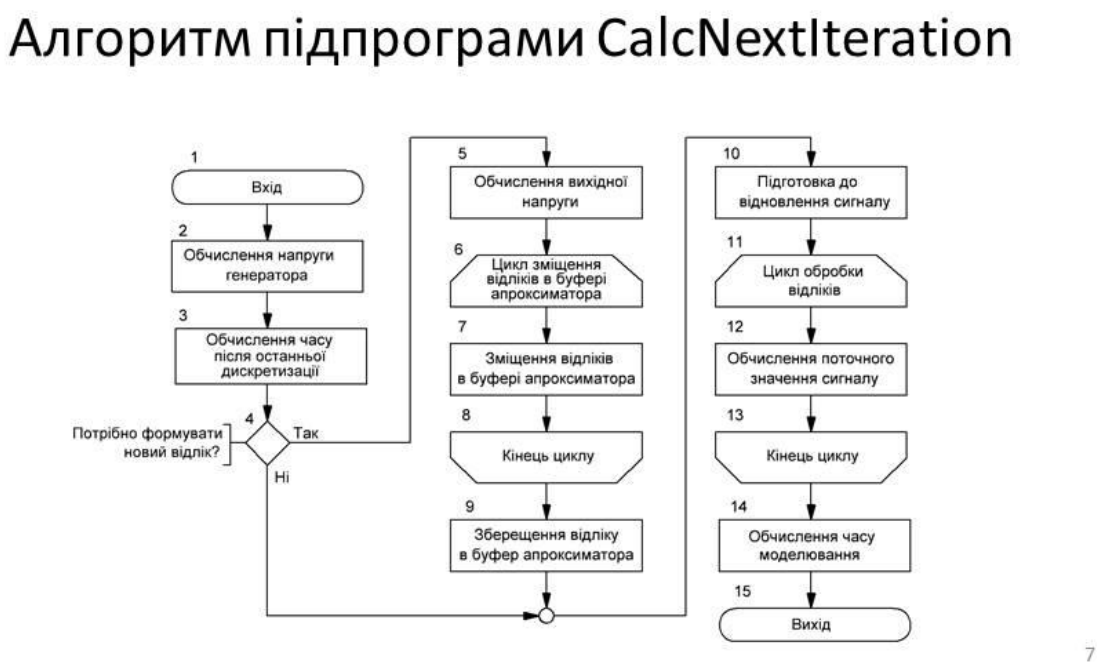
Слайд 5 – Універсальна лабораторія Labs



Слайд 6 – Інтерфейс програмного забезпечення



Слайд 7 – Алгоритм підпрограми CalcNextIteration



Слайд 8 – Результати роботи програми

Слайд 9 – Висновки та рекомендації

Висновки та рекомендації

1. Проведено аналіз методів відновлення інформаційного трафіку, показано доцільність відновлення трафіку різними методами апроксимації, а саме: за допомогою ряду Котельникова, за допомогою сплайн-апроксимацій та за допомогою вейвлет-функцій

2. Проведено аналіз методів практичної реалізації математичних моделей, обґрунтовано використання технології динамічно-зв'язаних бібліотек та універсальної віртуальної лабораторії

3. Створено новий програмний модуль для досліджень методів відновлення трафіку за допомогою ряду Котельникова. Проведено тестування його працездатності, яке підвердило усі теоретичні розрахунки

4. Результати роботи доцільно рекомендувати для практичного застосування у наступних напрямках:
– при проєктуванні пристроїв для передачі інформації, інфокомунікаційної техніки та інших застосунків, в яких використовується відновлення інфокомунікаційного трафіку;
– під час підготовки фахівців у галузях інформаційних технологій, інфокомунікації та радіотехніки, у тому числі і при вивченні методів передавання інформації за допомогою цифрових систем

9

Слайд 10 – Заключний слайд

Дякую за увагу!

Климчук Олексій Сергійович

10

ДОДАТОК Б

ВИХІДНИЙ КОД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

```
library Math01;

uses
  VCL.Graphics,
  SysUtils,
  Classes,
  VCL.Dialogs,
  VCL.Forms,
  Math,
  fmEditSwitch in '..\Common\fmEditSwitch.pas' {frmEditSwitch},
  uDLLTools in '..\Common\uDLLTools.pas',
  uPaintComponents in '..\Common\uPaintComponents.pas',
  uComClasses in '..\Common\uComClasses.pas',
  uComTools in '..\Common\uComTools.pas',
  fmEditComponent in '..\Common\fmEditComponent.pas' {frmEditComponent};

{$R *.res}

const
  SCHEM_0 = 0;
  BUFFER_SIZE = 1000;
  MAX_FREQUENCY = 10000;
  DEBUG_MODE = False;
  N_S = 5;

var
  G1: record
    Info: PComponentInfo;
    A1: PDouble;
    F1: PDouble;
    A2: PDouble;
    F2: PDouble;
    U: Double;
  end;

  D1: record
    Info: PComponentInfo;
    Fs: PDouble;
    LastSampleTime: Double;
    U: Double;
  end;

  R1: record
    Info: PComponentInfo;
    SampleCount: PDouble;
    Us: array [-N_S..N_S] of Double;
    U: Double;
  end;

  SampleNumber: Integer;

// ############################################################################
//                       Налаштування параметрів
// ############################################################################

procedure SetLayoutGlobals(LG: PLayoutGlobals); stdcall;
begin
  Application.Handle := LG.ApplicationHandle;
```

```
  gVariantNumber := LG.VariantNumber;
  gLanguage := LG.Language;
  gGraphicParams := LG.GraphicParams;
  LG.DebugMode := DEBUG_MODE;
end;

// ############################################################################

function GetLayoutInfo(LayoutInfo: PLayoutInfo;
  aLanguage: TLanguage): Boolean; stdcall;
begin
  case aLanguage of
    lgEng: LayoutInfo.Subject := 'Math';
    lgRus: LayoutInfo.Subject := 'ВМ';
    else LayoutInfo.Subject := 'ВМ';
  end;

  case aLanguage of
    lgEng: LayoutInfo.Caption := 'Lab 1';
    lgRus: LayoutInfo.Caption := 'Лаб 1';
    else LayoutInfo.Caption := 'Лаб 1';
  end;

  case aLanguage of
    lgEng: LayoutInfo.FullCaption := 'Kotelnikov series';
    lgRus: LayoutInfo.FullCaption := 'Ряд Котельникова';
    else LayoutInfo.FullCaption := 'Ряд Котельникова';
  end;

  case aLanguage of
    lgEng: LayoutInfo.Decription := 'Signal restoration using Kotelnikov series.';
    lgRus: LayoutInfo.Decription := 'Восстановление сигнала с помощью ряда
Котельникоа.';
    else LayoutInfo.Decription := 'Выдновлення сигналу за допомогою ряду
Котельникова.';
  end;

  LayoutInfo.SchemIndexDefault := SCHEM_0;

  Result := True;
end;

// ############################################################################
//                       Налаштування параметрів схеми
// ############################################################################

function SetSchemInfo(SchemInfo: PSchemInfo; const SchemIndex: Integer): Boolean;
stdcall;
begin
  Result := False;

  case SchemIndex of
    SCHEM_0: begin
      Result := True;

      case gLanguage of
        lgEng: SchemInfo.Caption := 'Kotelnikov series';
        lgRus: SchemInfo.Caption := 'Ряд Котельникова';
        else SchemInfo.Caption := 'Ряд Котельникова';
      end;

      SchemInfo.Width := 30;
      SchemInfo.Height := 2;
```

```
      SchemInfo.UpdateInterval := 0;
      SchemInfo.UpdateOnStop := False;
    end;
  end;
end;

// ############################################################################
//                            Налаштування компонентів
// ############################################################################

function SetComponentInfo(Component: PComponentInfo; const SchemIndex,
  ComponentIndex: Integer): Boolean; stdcall;
begin
  Result := False;

  case ComponentIndex of
    0: begin
      Result := True;
      G1.Info := Component;

      case gLanguage of
        lgEng: Component.Caption := 'Generator';
        lgRus: Component.Caption := 'Генератор';
        else Component.Caption := 'Генератор';
      end;
    end;

    1: begin
      Result := True;
      G1.Info := Component;

      case gLanguage of
        lgEng: Component.Caption := 'Sampler';
        lgRus: Component.Caption := 'Дискретизатор';
        else Component.Caption := 'Дискретизатор';
      end;
    end;
  end;
end;

// ############################################################################
//                    Налаштування параметрів компонентів
// ############################################################################

function SetComponentParamInfo(Param: PComponentParamInfo;
  const SchemIndex, ComponentIndex, ParamIndex: Integer): Boolean; stdcall;
begin
  Result := False;

   case ComponentIndex of
    // G1
    0: case ParamIndex of
      0: begin
        Result := True;
        G1.A1 := @Param.Value;

        case gLanguage of
          lgEng: Param.FullCaption := 'Amplitude 1';
          lgRus: Param.FullCaption := 'Амплитуда 1';
          else Param.FullCaption := 'Амплітуда 1';
        end;

        case gLanguage of
```

```
    lgEng: Param.ShortCaption := 'A1';
    lgRus: Param.ShortCaption := 'A1';
    else Param.ShortCaption := 'A1';
  end;

  case gLanguage of
    lgEng: Param.UnitCaption := 'V';
    lgRus: Param.UnitCaption := 'В';
    else Param.UnitCaption := 'В';
  end;

  Param.Minimum := 0;
  Param.Maximum := 1;
  Param.Value := 0.5;
end;

1: begin
  Result := True;
  G1.F1 := @Param.Value;

  case gLanguage of
    lgEng: Param.FullCaption := 'Frequency 1';
    lgRus: Param.FullCaption := 'Частота 1';
    else Param.FullCaption := 'Частота 1';
  end;

  case gLanguage of
    lgEng: Param.ShortCaption := 'F1';
    lgRus: Param.ShortCaption := 'F1';
    else Param.ShortCaption := 'F1';
  end;

  case gLanguage of
    lgEng: Param.UnitCaption := 'Гц';
    lgRus: Param.UnitCaption := 'Гц';
    else Param.UnitCaption := 'Гц';
  end;

  Param.Minimum := 1;
  Param.Maximum := 10000;
  Param.Value := 1000;
end;

2: begin
  Result := True;
  G1.A2 := @Param.Value;

  case gLanguage of
    lgEng: Param.FullCaption := 'Amplitude 2';
    lgRus: Param.FullCaption := 'Амплитуда 2';
    else Param.FullCaption := 'Амплітуда 2';
  end;

  case gLanguage of
    lgEng: Param.ShortCaption := 'A2';
    lgRus: Param.ShortCaption := 'A2';
    else Param.ShortCaption := 'A2';
  end;

  case gLanguage of
    lgEng: Param.UnitCaption := 'V';
    lgRus: Param.UnitCaption := 'В';
    else Param.UnitCaption := 'В';
```

```
    end;

    Param.Minimum := 0;
    Param.Maximum := 1;
    Param.Value := 0.5;
  end;

  3: begin
    Result := True;
    G1.F2 := @Param.Value;

    case gLanguage of
      lgEng: Param.FullCaption := 'Frequency 2';
      lgRus: Param.FullCaption := 'Частота 2';
      else Param.FullCaption := 'Частота 2';
    end;

    case gLanguage of
      lgEng: Param.ShortCaption := 'F2';
      lgRus: Param.ShortCaption := 'F2';
      else Param.ShortCaption := 'F2';
    end;

    case gLanguage of
      lgEng: Param.UnitCaption := 'Гц';
      lgRus: Param.UnitCaption := 'Гц';
      else Param.UnitCaption := 'Гц';
    end;

    Param.Minimum := 1;
    Param.Maximum := 10000;
    Param.Value := 100;
  end;
end;

// D1
1: case ParamIndex of
  0: begin
    Result := True;
    D1.Fs := @Param.Value;

    case gLanguage of
      lgEng: Param.FullCaption := 'Sampling frequency';
      lgRus: Param.FullCaption := 'Частота дискретизации';
      else Param.FullCaption := 'Частота дискретизації';
    end;

    case gLanguage of
      lgEng: Param.ShortCaption := 'Fs';
      lgRus: Param.ShortCaption := 'Fд';
      else Param.ShortCaption := 'Fд';
    end;

    case gLanguage of
      lgEng: Param.UnitCaption := 'Гц';
      lgRus: Param.UnitCaption := 'Гц';
      else Param.UnitCaption := 'Гц';
    end;

    Param.Minimum := 1;
    Param.Maximum := 10000;
    Param.Value := 2000;
  end;
```

```
    end;

    // R1
    2: case ParamIndex of
      0: begin
        Result := True;
        D1.Fs := @Param.Value;

        case gLanguage of
          lgEng: Param.FullCaption := 'Samples count';
          lgRus: Param.FullCaption := 'Количество семплов';
          else Param.FullCaption := 'Кількість семплів';
        end;

        case gLanguage of
          lgEng: Param.ShortCaption := 'Ns';
          lgRus: Param.ShortCaption := 'Ns';
          else Param.ShortCaption := 'Ns';
        end;

        case gLanguage of
          lgEng: Param.UnitCaption := '';
          lgRus: Param.UnitCaption := '';
          else Param.UnitCaption := '';
        end;

        Param.Minimum := 1;
        Param.Maximum := 10;
        Param.Value := 3;
      end;
    end;
  end;
end;

// ############################################################################
//                        Налаштування осцилографу
// ############################################################################

function SetOscillographInfo(OscillographInfo: POscillographInfo;
  const SchemIndex: Integer): Boolean; stdcall;
begin
  Result := True;
  OscillographInfo.GridVisible := True;
  OscillographInfo.GridColor := clSilver;
  OscillographInfo.GridCountX := 12;
  OscillographInfo.GridCountY := 12;
  OscillographInfo.BackgroundColor := clBlack;
  OscillographInfo.EditBackgroundColor := False;
  OscillographInfo.XScale := 1e-3;
  OscillographInfo.BufferSize := BUFFER_SIZE;
end;

// ############################################################################
//                        Налаштування каналів осцилографа
// ############################################################################

function SetOscillographChannelInfo(ChannelInfo: POscillographChannelInfo;
  const SchemIndex, ChannelIndex: Integer): Boolean; stdcall;
begin
  case ChannelIndex of
    0: begin
      Result := True;
      ChannelInfo.Visible := True;
```

```
      ChannelInfo.DCOn := True;
      ChannelInfo.Color := clRed;
      ChannelInfo.Width := 3;
      ChannelInfo.Position := -30;
      ChannelInfo.YScale := 1;
      ChannelInfo.PointIndexDefault := 0;
      ChannelInfo.ZeroLineVisible := True;
    end;

    1: begin
      Result := True;
      ChannelInfo.Visible := True;
      ChannelInfo.DCOn := True;
      ChannelInfo.Color := clYellow;
      ChannelInfo.Width := 3;
      ChannelInfo.Position := 0;
      ChannelInfo.YScale := 1;
      ChannelInfo.PointIndexDefault := 1;
      ChannelInfo.ZeroLineVisible := True;
    end;

    2: begin
      Result := True;
      ChannelInfo.Visible := True;
      ChannelInfo.DCOn := True;
      ChannelInfo.Color := clLime;
      ChannelInfo.Width := 3;
      ChannelInfo.Position := 30;
      ChannelInfo.YScale := 1;
      ChannelInfo.PointIndexDefault := 2;
      ChannelInfo.ZeroLineVisible := True;
    end;
  end;
end;

// ############################################################################
//              Налаштування контрольних точок осцилографу
// ############################################################################

function SetOscillographChannelPointInfo(ControlPoint:
   POscillographChannelPointInfo;
  const SchemIndex, ChannelIndex, PointIndex: Integer): Boolean; stdcall;
begin
  Result := False;

  if not Assigned(ControlPoint)
    then Exit;

  case PointIndex of
    0: begin
      Result := True;

      case gLanguage of
        lgEng:  ControlPoint.Caption := 's(t)';
        lgRus: ControlPoint.Caption := 's(t)';
        else  ControlPoint.Caption := 's(t)';
      end;

      case gLanguage of
        lgEng: ControlPoint.UnitCaption := 'V';
        lgRus: ControlPoint.UnitCaption := 'В';
        else ControlPoint.UnitCaption := 'В';
      end;
```

```
      ControlPoint.Source := @G1.U;
    end;

    1: begin
      Result := True;

      case gLanguage of
        lgEng:  ControlPoint.Caption := 'd(t)';
        lgRus: ControlPoint.Caption := 'd(t)';
        else  ControlPoint.Caption := 'd(t)';
      end;

      case gLanguage of
        lgEng: ControlPoint.UnitCaption := 'V';
        lgRus: ControlPoint.UnitCaption := 'В';
        else ControlPoint.UnitCaption := 'В';
      end;

      ControlPoint.Source := @D1.U;
    end;

    2: begin
      Result := True;

      case gLanguage of
        lgEng:  ControlPoint.Caption := 'y(t)';
        lgRus: ControlPoint.Caption := 'y(t)';
        else  ControlPoint.Caption := 'y(t)';
      end;

      case gLanguage of
        lgEng: ControlPoint.UnitCaption := 'V';
        lgRus: ControlPoint.UnitCaption := 'В';
        else ControlPoint.UnitCaption := 'В';
      end;

      ControlPoint.Source := @R1.U;
    end;
  end;
end;

// ############################################################################
//                              Графіка
// ############################################################################

procedure PaintSchem_0(C: TCanvas);
const
  LINE_WIDGHT_DEFAULT = 2;
var
  S: String;
begin
  SetPaintMode(C, False, LINE_WIDGHT_DEFAULT);

  // генератор
  C.Rectangle(TX(0), TY(0), TX(7), TY(2));

  case gLanguage of
    lgEng: S := 'Generator';
    lgRus: S := 'Генератор';
    else S := 'Генератор';
  end;
```

```
  PaintText(C, S, False, 0.2, 0.5, 6.6, 1, haCenter, vaCenter);

  C.Polyline([TP(7, 1), TP(9,  1)]);
  C.Polyline([TP(8.5, 0.75), TP(9,  1), TP(8.5, 1.25)]);

  PaintText(C, 's(t)', False, 7.4, 0, 1, 0.8, haLeft, vaCenter);

  // дискретизатор
  C.Rectangle(TX(9), TY(0), TX(16), TY(2));

  case gLanguage of
    lgEng: S := 'Sampler';
    lgRus: S := 'Дискретизатор';
    else S := 'Дискретизатор';
  end;

  PaintText(C, S, False, 9.2, 0.5, 6.6, 1, haCenter, vaCenter);

  C.Polyline([TP(16, 1), TP(18,  1)]);
  C.Polyline([TP(17.5, 0.75), TP(18,  1), TP(17.5, 1.25)]);

  PaintText(C, 'd(t)', False, 16.4, 0, 1, 0.8, haLeft, vaCenter);

  // апроксиматор
  C.Rectangle(TX(18), TY(0), TX(25), TY(2));

  case gLanguage of
    lgEng: S := 'Approximator';
    lgRus: S := 'Аппроксиматор';
    else S := 'Апроксиматор';
  end;

  PaintText(C, S, False, 18.2, 0.5, 6.6, 1, haCenter, vaCenter);

  C.Polyline([TP(25, 1), TP(27,  1)]);
  C.Polyline([TP(26.5, 0.75), TP(27,  1), TP(26.5, 1.25)]);

  PaintText(C, 'y(t)', False, 25.4, 0, 1, 0.8, haLeft, vaCenter);

  case gLanguage of
    lgEng: S := 'Output';
    lgRus: S := 'Выход';
    else S := 'Вихід';
  end;

  PaintText(C, S, False, 27.4, 0.6, 6.6, 0.8, haLeft, vaCenter);
end;

// ############################################################################

procedure PaintSchem(const SchemIndex: Integer; const InSimulation: Boolean);
stdcall;
var
  C: TCanvas;
begin
  if not Assigned(gGraphicParams)
    then Exit;

  C := TCanvas.Create;

  C.Brush.Color := clWhite;
  C.Brush.Style := bsSolid;
```

```
  C.Pen.Color := clBlack;
  C.Pen.Width := 2;

  C.Font.Name := 'Arial';
  C.Font.Style := [fsItalic];

  try
    C.Handle := gGraphicParams.Handle;

    case SchemIndex of
      0: PaintSchem_0(C);
    end;
  finally
    C.Handle := 0;
    C.Free;
  end;
end;

// ############################################################################
//                            Математична модель
// ############################################################################

procedure ResetSchem(); stdcall;
var
  k: Integer;
begin
  G1.U := 0;

  D1.U := 0;
  D1.LastSampleTime := -MaxDouble;

  for k := Low(R1.Us) to High(R1.Us) do
    R1.Us[k] := 0;

  SampleNumber := 0;
end;

procedure CalcNextIteration(SchemIndex: Integer; var CurTime: Double;
  var ResetSimulation, StopSumilation, UpdateSchem: Boolean); stdcall;
const
  DELTA_T = 1 / (MAX_FREQUENCY * 1000);
var
  DT: Double;
  k: Integer;
  X: Double;
begin
  // генератор сигналів
  G1.U := G1.A1^ * Cos(2 * pi * Frac(G1.F1^ * CurTime))
    + G1.A2^ * Cos(2 * pi * Frac(G1.F2^ * CurTime));

  // дискретизатор
  DT := CurTime - D1.LastSampleTime;
  if DT >= (1 / D1.Fs^) then
  begin
    D1.LastSampleTime := CurTime;
    D1.U := G1.U;

    for k := Low(R1.Us) to High(R1.Us) - 1 do
      R1.Us[k] := R1.Us[k + 1];

    R1.Us[High(R1.Us)] := G1.U;

    Inc(SampleNumber);
```

```
  end else
  begin
    D1.U := 0;
  end;

  R1.U := 0;
  for k := Low(R1.Us) to High(R1.Us) do
  begin
    X := Pi * D1.Fs^ * (CurTime - (k + SampleNumber) / (D1.Fs^));

    if X <> 0 then
      R1.U := R1.U + R1.Us[k] * Sin(X) / X
    else
      R1.U := R1.U + R1.Us[k];
  end;

  // час
  CurTime := CurTime + DELTA_T;
end;

exports
  SetLayoutGlobals,
  GetLayoutInfo,
  SetSchemInfo,
  SetComponentInfo,
  SetComponentParamInfo,
  SetOscillographInfo,
  SetOscillographChannelInfo,
  SetOscillographChannelPointInfo,
  PaintSchem,
  ResetSchem,
  CalcNextIteration;

begin
end.
```

ДОДАТОК В

ТЕЗИ ДОПОВІДІ НА ІХ ВСЕУКРАЇНСЬКІЙ НАУКОВО-ПРАКТИЧНІЙ КОНФЕРЕНЦІЇ СТУДЕНТІВ, АСПІРАНТІВ ТА МОЛОДИХ УЧЕНИХ «ГУМАНІТАРНИЙ І ІННОВАЦІЙНИЙ РАКУРС ПРОФЕСІЙНОЇ МАЙСТЕРНОСТІ: ПОШУКИ МОЛОДИХ ВЧЕНИХ»

15 грудня 2023 року м. Одеса, Україна

ПРОГРАМНА РЕАЛІЗАЦІЯ ВІДНОВЛЕННЯ ТРАФІКУ ЗА ДОПОМОГОЮ РЯДУ КОТЕЛЬНИКОВА

Стрелковська І.В.
доктор технічних наук, професор
декан факультету кібербезпеки, програмної інженерії та комп'ютерних наук
Міжнародного гуманітарного університету
м. Одеса, Україна

Русу О.П.
кандидат технічних наук,
доцент кафедри комп'ютерних наук
Міжнародного гуманітарного університету
м. Одеса, Україна

Климчук О.С.
здобувач вищої освіти другого (магістерського рівня)
спеціальності 121 Інженерія програмного забезпечення
Міжнародного гуманітарного університету
м. Одеса, Україна

Постійний розвиток інфокомунікаційних систем призводить до стрімкої зміни технологій та регулярного підвищення вимог до рівня якості послуг, що надаються. Тому необхідно постійно шукати нові шляхи підвищення якості передавання інформації та періодично оновлювати відомі і перевірені рішення. Одними з основних завдань, які вирішуються в сучасних інфокомунікаційних системах та мережах, є аналіз та синтез інформаційних сигналів, яким притаманні

як повна детермінованість, так і раптові непередбачувані «сплески» і швидкі осциляції. Відтворити сигнал з високою точністю на приймальній стороні можна за допомогою сучасних методів апроксимації, наприклад, сплайн-апроксимації (лінійної, квадратичної чи кубічної), або вейвлет-функцій на основі вейвлету Котельникова-Шеннона [1].

Використання вейвлет- та сплайн-функцій дозволяє досягти більшої точності відтворення сигналу, проте класичні методи відновлення сигналу на основі ряду Котельникова ще не втратили своєї актуальності і продовжують використовуватись в інфокомунікаційних системах. Однак, незважаючи на широку розповсюдженість методу відновлення сигналу за допомогою ряду Котельникова, кількість відомих програмних модулів, що підтримують швидке інтегрування в існуюче програмне забезпечення, наразі обмежена. Це і обумовило мету цієї роботи, яка полягає у створенні програмного модуля для відновлення трафіку за допомогою ряду Котельникова.

Розроблений програмний модуль, призначений для використання у віртуальній лабораторії [2], складається з генератора тестового сигналу *s(t)*, дискретизатора, призначеного для формування відліків *d(t)*, та апроксиматора, що забезпечує відновлення первинного сигналу (Рис. 1).

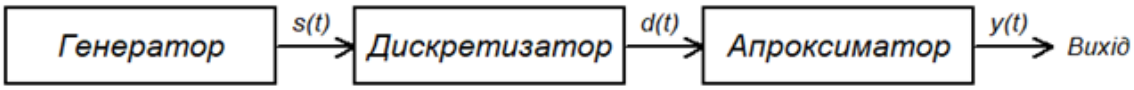


Рисунок 1 – Структурна схема програмного модуля

Спрощений алгоритм відновлення сигналу, реалізований у програмному забезпеченні, наведено на рисунку 2. Після входу у підпрограму розрахунку (блок 1) відбувається обчислення вихідної напруги генератора (блок 2). Після цього у блоці 3 виконується обчислення часу, який минув після останньої дискретизації, а у блоці 4 – його перевірка Якщо цей час перевищує тривалість періоду дискретизації, то формується новий відлік, а якщо ні, то вихідна напруга дискретизатора дорівнює нулю.

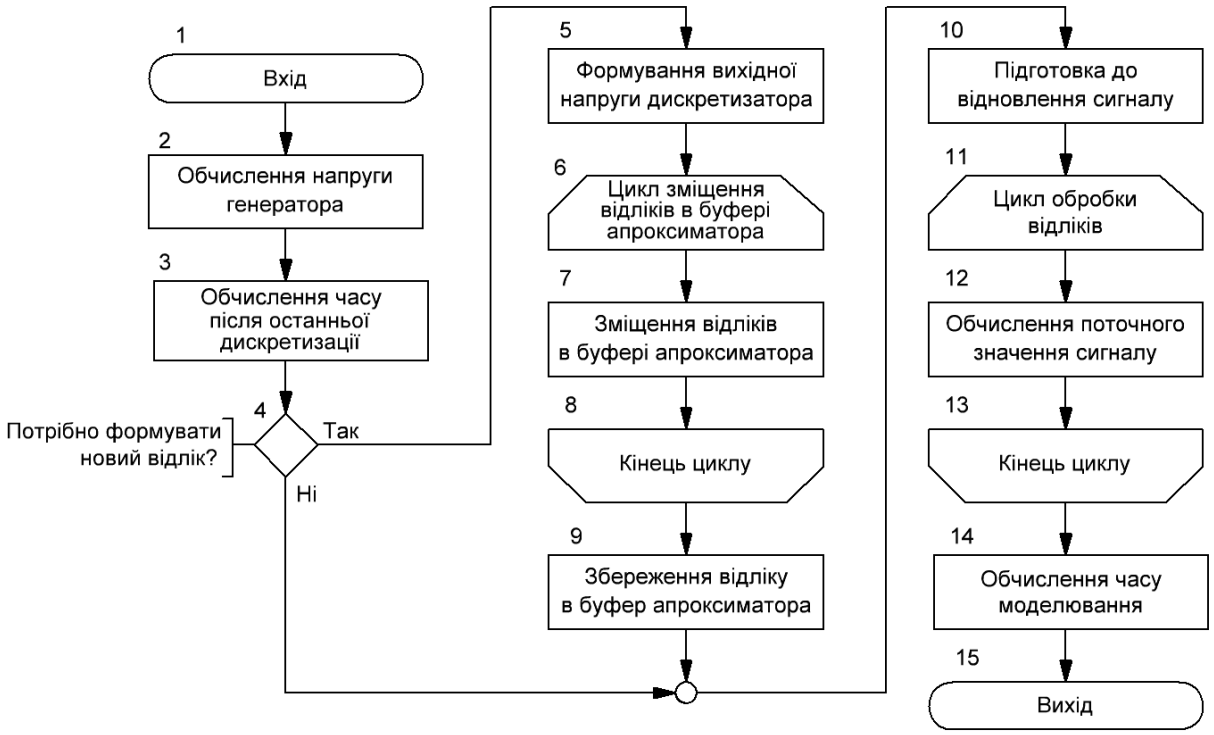


Рисунок 2 – Спрощений алгоритм підпрограми відновлення сигналу за допомогою ряду Котельникова

Вихідна напруга дискретизатора формується у блоці 5. Після цього у блоках 6 – 8 відбувається циклічне зміщення відліків в буфері апроксиматора. Кінцевим етапом роботи дискретизатора є запис поточного відліку в останню комірку буферу апроксиматора (блок 9).

Результати відновлення тестового сигналу наведено на рисунку 3. Як видно з рисунку, при частоті дискретизації $f_д$ = 2 кГц, що вдвічі перевищує максимальну частоту генератора (f_{max} = 1 кГц), відбувається якісне відновлення первинного сигналу як за амплітудою, так і за формою. При цьому, перші 4 мс відновлений сигнал значно відрізняється від первинного, оскільки у цей час в буфері апроксиматора ще недостатньо відліків для його відновлення. Крім того, спостерігається відставання у часі між первинним та відновленим сигналом на величину приблизно 3 мс, що є цілком природним, оскільки для якісного відновлення сигналу апроксиматор повинен мати певну інформацію як про попередні, так і про наступні відліки.

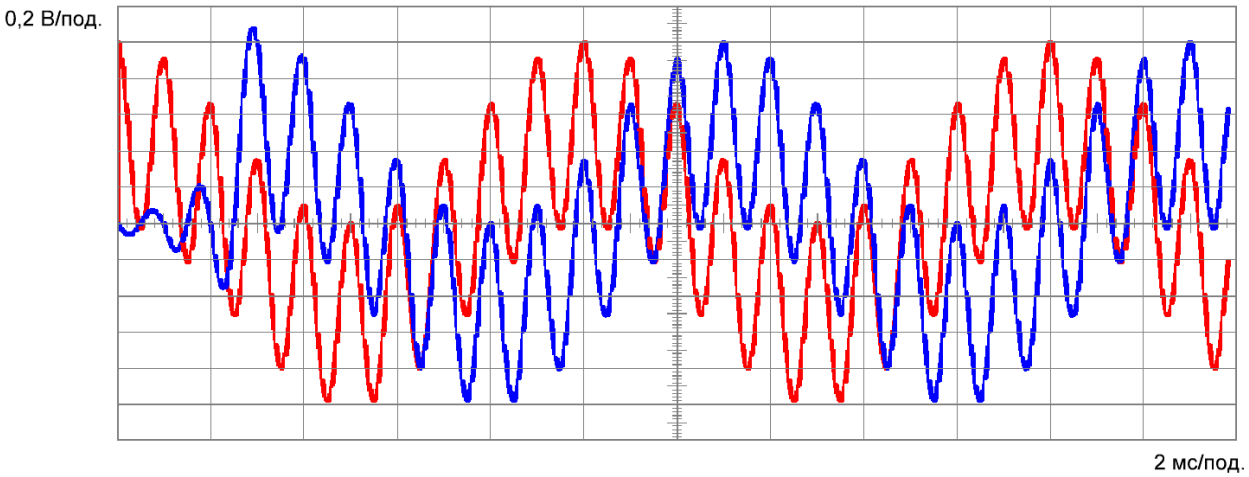


Рисунок 3 – Результати роботи програмного модуля (f_{max} = 1 кГц, $f_д$ = 2 кГц), первинний сигнал – червона лінія, відновлений сигнал – синя лінія

Висновки. Розроблене програмне забезпечення показало повну відповідність результатів моделювання теоретичним розрахункам. Результати моделювання показали, що первинний сигнал можна повністю відновити за допомогою ряду Котельникова за умови, що частота дискретизації $f_д$ буде щонайменше удвічі більша за максимальну частоту первинного сигналу f_{max}.

Література:

1. Strelkovskaya I., Solovskaya I., Strelkovska J., «Spline-Approximation and SplineExtrapolation Methods in Telecommunication Problems», Current Trends in Communication and Information Technologies. IPF 2020. Lecture Notes in Networks and Systems, vol 212, Springer, Cham, 2021, P. 3-22. URL: https://doi.org/10.1007/978-3-030-76343-5_1.

2. Русу О.П. Універсальна віртуальна лабораторія для традиційної та дистанційної форм навчання. Гуманітарний і інноваційний ракурс професійної майстерності: пошуки молодих вчених: матер. VIII Всеукр. наук.-практич. конф. студентів, аспірантів та молодих учених, (18 листоп. 2022 р., м. Одеса). – Львів – Торунь : Liha-Pres, 2022. – С. 385-388.