

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВИЩИЙ ДЕРЖАВНИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ІНЖЕНЕРНО-ТЕХНІЧНИЙ ФАКУЛЬТЕТ
КАФЕДРА КОМП'ЮТЕРНИХ СИСТЕМ ТА МЕРЕЖ**

**МЕТОДИЧНІ ВКАЗІВКИ І ЗАВДАННЯ
ДЛЯ ЛАБОРАТОРНИХ РОБІТ З КУРСУ
КОМП'ЮТЕРНІ ЗАСОБИ ОБРОБКИ СИГНАЛІВ
для студентів 4-го курсу інженерно-технічного факультету
спеціальності 123 «Комп'ютерна інженерія»**

Ужгород – 2021

Методичні вказівки і завдання для лабораторних робіт з курсу «Комп'ютерні засоби обробки сигналів» для студентів 4-го курсу інженерно-технічного факультету спеціальності 123 «Комп'ютерна інженерія».

Укладачі: Ваврук Є.Я. – канд. техн. наук, доцент кафедри комп'ютерних систем та мереж ДВНЗ «УжНУ»;
Тютюнникова Г. С. – старший викладач кафедри комп'ютерних систем та мереж ДВНЗ «УжНУ».

Рецензент: Глебена М.І., канд. фіз.-мат. наук, доцент кафедри системного аналізу та теорії оптимізації ДВНЗ «УжНУ».

Відповідальний за випуск – Горват П.П., канд. фіз.-мат. наук, доцент, завідувач кафедри комп'ютерних систем та мереж ДВНЗ «УжНУ».

Дані методичні вказівки розглянуто та схвалено на засіданні кафедри комп'ютерних систем та мереж, протокол № 11 від 20.05.2021 року та методичної комісії інженерно-технічного факультету протокол №4 від 24.05.2021 року.

ВСТУП

Метою викладання дисципліни «Комп'ютерні засоби обробки сигналів» є вивчення:

- основного математичного забезпечення, що застосовується при розробці комп'ютерних засобів реєстрації, обробки та відображення цифрових сигналів і зображень;
- основних методів побудови швидких алгоритмів цифрової обробки сигналів та зображень (ЦОСЗ) і дослідження обчислювальної складності (складності реалізації) алгоритмів;
- основних принципів побудови програмованих проблемно-орієнтованих однокристальних комп'ютерів ЦОСЗ;
- основних принципів побудови спеціалізованих великих інтегральних схем ЦОСЗ;
- принципів та основних структурних рішень побудови програмно-апаратних комплексів ЦОСЗ.

Вивчення дисципліни «Комп'ютерні засоби обробки сигналів» дає студентам необхідну теоретичну і практичну підготовку для того, щоб знати теоретичні основи стиску зображень; принципи побудови сучасних систем розпізнавання мови; принципи побудови і функціонування систем і процесорів обробки сигналів та зображень; засоби автоматизованого проектування та аналізу характеристик вузлів ЦОСЗ.

Дані методичні вказівки ознайомлять студентів з різними типами цифрових фільтрів і застосуванням їх на практиці; ознайомлять з методами та засобами фільтрації сигналів та зображень. Виконання студентами лабораторних робіт з курсу «Комп'ютерні засоби обробки сигналів» дозволяє закріпити теоретичні знання, надає можливість набути практичних навичок роботи в середовищі MATLAB 6.0. та вміння розробляти оптимальні алгоритми і процедури обробки сигналів та зображень, проектувати відповідні системи і окремі функціональні вузли на різній елементній базі; експлуатувати, проводити ремонтні та профілактичні роботи існуючих систем та процесорів ЦОСЗ.

Кожній лабораторній роботі має передувати самостійна підготовка студентів, у процесі якої вони вивчають теоретичні відомості, що стосуються виконуваної роботи. При оформленні студент повинен сформулювати мету і порядок виконання лабораторної роботи і відповісти на контрольні питання викладача.

Перед початком наступного заняття в лабораторії студент зобов'язаний подати викладачеві повністю оформлений звіт з попередньої лабораторної роботи. Звіт повинен містити всі необхідні схеми, формули, таблиці, діаграми, графіки, одержані у процесі експериментального дослідження схем, а також підсумкові висновки.

Лабораторна робота №1

Тема. Аналіз обчислювальної похибки при виконанні базових операцій алгоритмів цифрової обробки сигналів. Обчислення математичних функцій.

Мета роботи. Дослідити шляхи виникнення обчислювальної похибки та її вплив на точність обчислень. Проаналізувати величину похибки при обчисленні деяких математичних функцій.

Загальні теоретичні відомості

При реалізації обчислень на процесорах обробки сигналів чи НВІС, які характеризуються обмеженою розрядністю і роботою в форматі фіксованої крапки необхідно враховувати ефекти, які викликані, насамперед, наближеним представленням формули обчислень і кінцевою розрядністю використовуваних регістрів. До таких ефектів відносяться:

- а) шум аналогово-цифрового перетворення;
- б) некорельований шум заокруглення;
- в) похибки, які викликані квантуванням коефіцієнтів.

Враховуючи методи представлення чисел, способи квантування, які використовуються для скорочення розрядності чисел до необхідної величини, а також особливості структурної схеми обчислень, в кожному конкретному випадку можна оцінити, як перераховані ефекти впливають на результат обчислень.

Квантування в цифрових пристроях

При квантуванні використовують два стандартних способи: відкидання і заокруглення. Розглянемо їх особливості стосовно різних систем числення і похибки, які виникають при цьому. Припускається, що всі значення чисел по модулю менші від 1.0 ($|X| < 1.0$).

Відкидання. Відкидаються всі молодші розряди, що стоять після найменшого розряду, який зберігається. Тоді значення похибки для доповняльного коду задовольняє нерівність:

$$-2^{-b} \leq X_{вдк} - X \leq 0,$$

де b - число розрядів, що зберігаються; $X_{вдк}$ - відкинута значення X .

Для чисел, які представлені в прямому і оберненому кодах для від'ємних значень справедлива нерівність:

$$0 \leq X_{вдк} - X < 2^{-b}, \quad X < 0.$$

Важливим є те, що похибка відкидання лежить між значеннями нуля і числа, що пропорційне $\pm 2^{-b}$.

Заокруглення. При заокругленні вихідне число X замінюється найближчим до нього b -розрядним числом. Тоді похибка заокруглення задовольняє нерівність:

$$-2^{-b} / 2 \leq X_{ок} - X \leq 2^{-b} / 2$$

для всіх трьох методів представлення чисел (додаткового, прямого і оберненого коду).

Шум аналогово-цифрового перетворення

В залежності від методу квантування вхідної послідовності шум квантування може мати різний амплітудний розподіл. При найменшому кроці квантування Q похибка квантування $e(n)$ лежить в границях:

$$\begin{aligned} -Q/2 \leq e(n) \leq Q/2 & \text{ - для випадку заокруглення;} \\ 0 \leq e(n) \leq Q & \text{ - для випадку відкидання.} \end{aligned}$$

В даному випадку розподіл сигналу похибки є рівномірним. При цьому середнє значення похибки рівне нулю при заокругленні і $Q/2$ при відкиданні, а її дисперсія в обох випадках рівне $Q^2/12$. Як аналогію аналогово-цифрового перетворення в нашому випадку необхідно розглядати представлення вхідного (тестового) масиву чисел в заданій розрядній сітці b ; тоді Q дорівнюватиме b .

Некорельований шум заокруглення

В цифровій обробці використовуються операції множення, додавання і зсуву. Їх виконання приводить до необхідності розширення розрядної сітки. Наприклад, перемноження двох b -розрядних чисел приводить до $2b$ -розрядного результату, подальше перемноження може привести до нескінченного збільшення розрядної сітки. Для подолання ефекту застосовують квантування результатів множення до вихідної b -розрядної сітки з заокругленням або відкиданням молодших розрядів. При цьому виникає шум заокруглення.

При додаванні, в загальному випадку розширення розрядної сітки не виникає, але в деяких випадках може виникнути переповнення. Для подолання цього ефекту застосовують зсув результатів вправо і його квантування. Для квантування результатів множення і додавання застосовують заокруглення або відкидання, в залежності від вимог реалізації. Похибки, що виникають при цьому будуть мати випадковий характер.

Квантування коефіцієнтів

Постійні коефіцієнти, які використовуються при обчисленні базових операцій алгоритмів ЦОСЗ також представляються у фіксованому розрядному просторі. Загального підходу до їх квантування нема. Тому застосовується така оптимізація, щоб максимум зваженої різниці ідеальних і реальних обчислень був мінімальним. При цьому необхідно розглянути схему обчислень стосовно чутливості до розрядності коефіцієнтів. Для цього, необхідно, змінюючи спосіб квантування коефіцієнтів (відкидання, заокруглення), добитися найменшого розходження між ідеальною і розрахованою функцією.

Базові операції алгоритмів цифрової обробки сигналів та зображень

До основних базових операцій, які застосовуються в ЦОСЗ відносяться: множення, додавання і зсув. Оскільки, при виконанні цих операцій необхідно залишитися в заданій розрядній сітці розглянемо прийоми їх моделювання на мові Pascal.

<Розрядна сітка: $b = 8$.>

Множення

```
{x, k, y - 8-ми розрядні слова зі знаком;}  
{work - 16-ти розрядне слово зі знаком;}  
work := k * x; {множення}  
work := work + $80; {моделювання заокруглення}  
y := work shr 8; {квантування до 8-ми розрядів}
```

Додавання.

```
{x, k, y - 8-ми розрядні слова зі знаком;}  
{work - 16-ти розрядне слово зі знаком;}  
work := k + x; {сумування}  
work := work + 1; {моделювання заокруглення}  
y := work shr 1; {квантування до 8-ми розрядів}
```

Для іншої розрядної сітки (наприклад $b = 16$) застосовуються такі ж прийоми, з врахуванням довжини слова і відповідних коректуючи констант.

Порядок виконання роботи

1. Записати в аналітичному вигляді формулу математичної функції, що є оптимальною за складністю обчислень.
2. Розробити схему обчислень, блок схему виконання алгоритму, для режиму з фіксованою крапкою, враховуючи спосіб реалізації (ПОС, НВІС). Рекомендується враховувати 3-5 членів ряду.
3. Проаналізувати можливі шляхи виникнення похибок обчислень, методи їх подолання.
4. Створити тестову послідовність для досліджуваної функції і провести розрахунок еталонного зразку в режимі з рухомою крапкою на довільній мові програмування.

5. Провести обчислення математичної функції в режимі з фіксованою крапкою в бітовому просторі 8, 16 біт на довільній мові програмування для вибраної кількості членів ряду (див.п.2).

6. Провести порівняння результатів отриманих в п.4 і п.5 і пояснити причину їх розбіжності.

Зміст звіту до лабораторної роботи

1. Титульний аркуш.
2. Завдання на лабораторну роботу.
3. Теоретичний матеріал стосовно поставленого завдання.
4. Лістинг програми на довільній мові програмування.
5. Результати роботи – у відповідності з табл.1 і графіки абсолютної (Δ) і відносної (ε) похибки.
6. Висновки

Таблиця 1 – Результати роботи

Значення аргумента Δx	Значення $F(\Delta x)$			Значення Δ		Значення ε	
	F_e	F_{p16}	F_{p8}	Δ_{16}	Δ_8	ε_{16}	ε_8
-0,95							
...							
...							
0							
...							
...							
+0,95							

Примітка: 1. Абсолютна похибка визначається як $\Delta = |F_e - F_{pi}|$, де F_e – значення досліджуваної функції з рухомою крапкою, F_{pi} – значення досліджуваної функції з фіксованою крапкою для відповідних бітових просторів.

2. Відносна похибка визначається як $\varepsilon = (\Delta / F_e) * 100\%$.

Контрольні запитання

1. Які ефекти впливають на появу похибки при обчисленнях?
2. Які способи квантування Ви знаєте?
3. Назвіть відомі Вам критерії оцінки похибок.

Варіанти завдань до лабораторної роботи

Вар.	Функція	Формула розкладу	Додаткові дані
1	$(1 \pm x)^m$ ($ x \leq 1$)	$1 + \sum_{n=1}^{\infty} (\pm 1)^n \frac{m(m-1)\dots(m-n+1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.05$ $m = 1/4$
2	$\sin x$ ($ x < \infty$)	$\sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}$	$x \in]0., \pi/2[$ $\Delta x = 0.01$
3	$\cos x$ ($ x < \infty$)	$\sum_{n=1}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$	$x \in]0., \pi/2[$ $\Delta x = 0.02$
4	$(1 \pm x)^m$ ($ x \leq 1$)	$1 + \sum_{n=1}^{\infty} (\pm 1)^n \frac{m(m-1)\dots(m-n+1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.04$ $m = 1/2$
5	e^x ($ x < \infty$)	$\sum_{n=0}^{\infty} \frac{x^n}{n!}$	$x \in]0., 1.[$ $\Delta x = 0.01$
6	a^x ($ x < \infty$)	$\sum_{n=0}^{\infty} \frac{(x \ln a)^n}{n!}$	$x \in]0., 1.[$ $\Delta x = 0.1$ $a = 3$
7	$(1 \pm x)^m$ ($ x \leq 1$)	$1 + \sum_{n=1}^{\infty} (\pm 1)^n \frac{m(m-1)\dots(m-n+1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.1$ $m = 3/2$
8	a^x ($ x < \infty$)	$\sum_{n=0}^{\infty} \frac{(x \ln a)^n}{n!}$	$x \in]0., 1.[$ $\Delta x = 0.02$ $a = 3$
9	$\ln x$ ($x > 0$)	$2 \sum_{n=1}^{\infty} \frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}$	$x \in]0., 10.[$ $\Delta x = 0.1$
10	$(1 \pm x)^m$ ($ x \leq 1$)	$1 + \sum_{n=1}^{\infty} (\pm 1)^n \frac{m(m-1)\dots(m-n+1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.02$ $m = 5/2$
11	$\ln(1+x)$ ($-1 < x \leq 1$)	$\sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^n}{n}$	$x \in]-1., 1.[$ $\Delta x = 0.02$
12	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.05$ $m = 1/2$
13	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.01$ $m = 1/3$
14	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.02$ $m = 1/4$
15	$\ln(1-x)$ ($-1 \leq x < 1$)	$-\sum_{n=1}^{\infty} \frac{x^n}{n}$	$x \in]-1., 1.[$ $\Delta x = 0.1$
16	$\ln \frac{1+x}{1-x}$ ($ x < 1$)	$2 \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1}$	$x \in]-1., 1.[$ $\Delta x = 0.02$

Вар.	Функція	Формула розкладу	Додаткові дані
17	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.025$ $m = 1$
18	$\arcsin x$ ($ x \leq 1$)	$x + \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot 5 \dots (2n-1)}{2 \cdot 4 \cdot 6 \dots (2n)(2n+1)} x^{2n+1}$	$x \in]-1., 1.[$ $\Delta x = 0.02$
19	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.02$ $m = 3/2$
20	$\arccos x$ ($ x \leq 1$)	$\frac{\pi}{2} - x - \sum_{n=1}^{\infty} \frac{1 \cdot 3 \cdot 5 \dots (2n-1)}{2 \cdot 4 \cdot 6 \dots (2n)(2n+1)} x^{2n+1}$	$x \in]-1., 1.[$ $\Delta x = 0.01$
21	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.1$ $m = 2$
22	$\operatorname{arctg} x$ ($ x \leq 1$)	$\sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1}$	$x \in]-1., 1.[$ $\Delta x = 0.01$
23	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.05$ $m = 5/2$
24	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.02$ $m = 3$
25	$\operatorname{arctg} x$ ($ x \geq 1$)	$\pm \frac{\pi}{2} + \sum_{n=0}^{\infty} (-1)^{n+1} \frac{1}{(2n+1) x^{2n+1}}$ «+» при $x > 1$, «-» при $x < -1$	$x \in]-1., 1.[$ $\Delta x = 0.1$
26	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.01$ $m = 4$
27	$\ln \frac{1+x}{1-x}$ ($ x < 1$)	$2 \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1}$	$x \in]-1., 1.[$ $\Delta x = 0.025$
28	$\ln x$ ($x > 0$)	$2 \sum_{n=1}^{\infty} \frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}$	$x \in]0., 10.[$ $\Delta x = 0.4$
29	$(1 \pm x)^{-m}$ ($ x < 1$)	$1 + \sum_{n=1}^{\infty} (\mp 1)^n \frac{m(m+1)\dots(m+n-1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.2$ $m = 5$
30	$\sin x$ ($ x < \infty$)	$\sum_{n=1}^{\infty} (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!}$	$x \in]0., \pi/2[$ $\Delta x = 0.01$

Приклад виконання

Завдання

Вар.	Функція	Формула розкладу	Додаткові дані
31	$(1 \pm x)^m$ ($ x \leq 1$)	$1 + \sum_{n=1}^{\infty} (\pm 1)^n \frac{m(m-1)\dots(m-n+1)}{n!} x^n$	$x \in]-1., 1.[$ $\Delta x = 0.02$ $m = 1/3$

Згідно поставленого завдання, запишемо аналітичний вид математичної функцій, що буде обчислюватися, враховуючи 5 членів ряду:

$$(1 \pm x)^{1/3} = \sqrt[3]{1 \pm x} = 1 + \frac{1/3}{1!} x^1 + \frac{1/3 \cdot (1/3 - 1)}{2!} x^2 + \frac{1/3 \cdot (1/3 - 1) \cdot (1/3 - 2)}{3!} x^3 + \frac{1/3 \cdot (1/3 - 1) \cdot (1/3 - 2) \cdot (1/3 - 3)}{4!} x^4 + \frac{1/3 \cdot (1/3 - 1) \cdot (1/3 - 2) \cdot (1/3 - 3) \cdot (1/3 - 4)}{5!} x^5$$

Обчисливши коефіцієнти при степенях, отримуємо:

$$\sqrt[3]{1 \pm x} = 1 + 0.3333x^1 - 0.1111x^2 + 0.0617x^3 - 0.0411x^4 + 0.0302x^5.$$

Зауважимо, що точність представлення констант залежить від розрядної сітки, що буде використовуватися при обчисленнях.

Схема обчислень для режиму з фіксованою крапкою базується на заокругленні та відкиданні молодших розрядів при кожній операції множення та додавання, а програмно буде здійснюватися за допомогою приведення типів. Повний лістинг програми подано у Додатку.

При запуску створеної програми з'являється графічне вікно (рис.1).

N	X	Еталоне Y	8 розрядів Y	16 розрядів Y	8 Абсолютна похибка	16 Абсолютна похибка	8 Відносна похибка	16 Відносна похибка
1	-1.0000	125.149863	8.3029	8.3029329	116.8470	116.8469301	93.3657	93.3656077
2	-0.9800	113.915747	7.9361	7.9361253	105.9796	105.9796217	93.0333	93.0333378
3	-0.9600	103.517485	7.5821	7.5820652	95.9354	95.9354198	92.6756	92.6755705
4	-0.9400	93.907235	7.2405	7.2405217	86.6667	86.6667133	92.2897	92.2897083
5	-0.9200	85.039023	6.9113	6.9112604	78.1277	78.1277626	91.8728	91.8728365
6	-0.9000	76.868711	6.5940	6.5940462	70.2747	70.2746648	91.4217	91.4216772
7	-0.8800	69.353953	6.2886	6.2886478	63.0654	63.0653052	90.9327	90.9325316
8	-0.8600	62.454161	5.9948	5.9948300	56.4594	56.4593310	90.4013	90.4012320
9	-0.8400	56.130467	5.7124	5.7123585	50.4181	50.4181085	89.8231	89.8230697
10	-0.8200	50.345685	5.4410	5.4410013	44.9047	44.9046837	89.1927	89.1927157
11	-0.8000	45.064275	5.1805	5.1805237	39.8838	39.8837513	88.5043	88.5041450
12	-0.7800	40.252306	4.9307	4.9306911	35.3216	35.3216149	87.7505	87.7505376
13	-0.7600	35.877415	4.6913	4.6912724	31.1861	31.1861426	86.9240	86.9241627
14	-0.7400	31.908773	4.4620	4.4620319	27.4468	27.4467411	86.0165	86.0162849
15	-0.7200	28.317047	4.2427	4.2427361	24.0743	24.0743109	85.0170	85.0170249
16	-0.7000	25.074362	4.0332	4.0331515	21.0412	21.0412105	83.9152	83.9152378
17	-0.6800	22.154264	3.8330	3.8330448	18.3213	18.3212192	82.6988	82.6988880
18	-0.6600	19.531682	3.6422	3.6421819	15.8895	15.8895001	81.3524	81.3524411
19	-0.6400	17.182892	3.4603	3.4603289	13.7226	13.7225631	79.8620	79.8617782
20	-0.6200	15.085478	3.2873	3.2872527	11.7982	11.7982253	78.2090	78.2091578
21	-0.6000	13.218294	3.1227	3.1227194	10.0956	10.0955746	76.3760	76.3757759
22	-0.5800	11.561431	2.9665	2.9664947	8.5949	8.5949363	74.3411	74.3414574
23	-0.5600	10.096173	2.8183	2.8183459	7.2779	7.2778271	72.0857	72.0850079
24	-0.5400	8.804967	2.6780	2.6780389	6.1270	6.1269281	69.5857	69.5849070
25	-0.5200	7.671379	2.5453	2.5453395	5.1261	5.1260395	66.8211	66.8203135
26	-0.5000	6.680059	2.4200	2.4200147	4.2601	4.2600443	63.7734	63.7725550
27	-0.4800	5.816706	2.3018	2.3018306	3.5149	3.5148754	60.4277	60.4272487
28	-0.4600	5.068027	2.1906	2.1905534	2.8774	2.8774736	56.7755	56.7769982
29	-0.4400	4.421701	2.0859	2.0859494	2.3358	2.3357516	52.8258	52.8247297
30	-0.4200	3.866344	1.9878	1.9877850	1.8785	1.8785590	48.5860	48.5874770
31	-0.4000	3.391467	1.8958	1.8958265	1.4957	1.4956405	44.1019	44.1001048
32	-0.3800	2.987442	1.8098	1.8098401	1.1776	1.1776019	39.4183	39.4184021
33	-0.3600	2.645463	1.7296	1.7295922	0.9159	0.9158708	34.6215	34.6204351
34	-0.3400	2.357510	1.6548	1.6548492	0.7027	0.7026608	29.8069	29.8052097
35	-0.3200	2.116310	1.5854	1.5853771	0.5309	0.5309329	25.0861	25.0876715
36	-0.3000	1.915301	1.5209	1.5209426	0.3944	0.3943584	20.5921	20.5898916
37	-0.2800	1.748594	1.4613	1.4613117	0.2873	0.2872823	16.4303	16.4293312
38	-0.2600	1.610936	1.4063	1.4062508	0.2046	0.2046852	12.7007	12.7059796
39	-0.2400	1.497672	1.3555	1.3555263	0.1422	0.1421457	9.4947	9.4911102
40	-0.2200	1.404707	1.3089	1.3089043	0.0958	0.0958027	6.8199	6.8201198

Рисунок 1 – Вікно програми з фрагментом обчислених значень.

Задавати та змінювати вхідні параметри дозволяють відповідні поля та кнопки, забезпечені графічним інтерфейсом. При цьому числові дані з'являються у відповідних стовбцях форми після

натискання кнопки «Обчислити». На рис. 1 відображено перші 40 значень вихідного результату для заданого варіанту.

Після розрахунку табличних значень, графічне відображення можна отримати, використавши довільний стандартний програмний засіб, перенісши значення відповідних полів. Графіки абсолютної та відносної похибок для 8 та 16 розрядів, приведені на рисунках 2 - 3 та 4 - 5, відповідно.

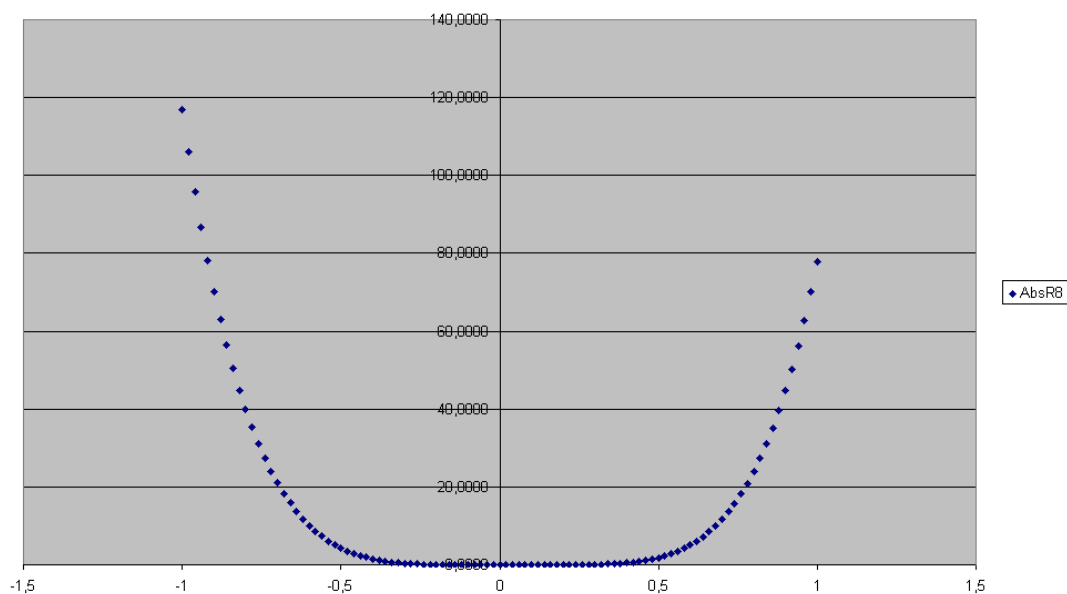


Рисунок 2 – Графік абсолютної похибки для 8 розрядної сітки

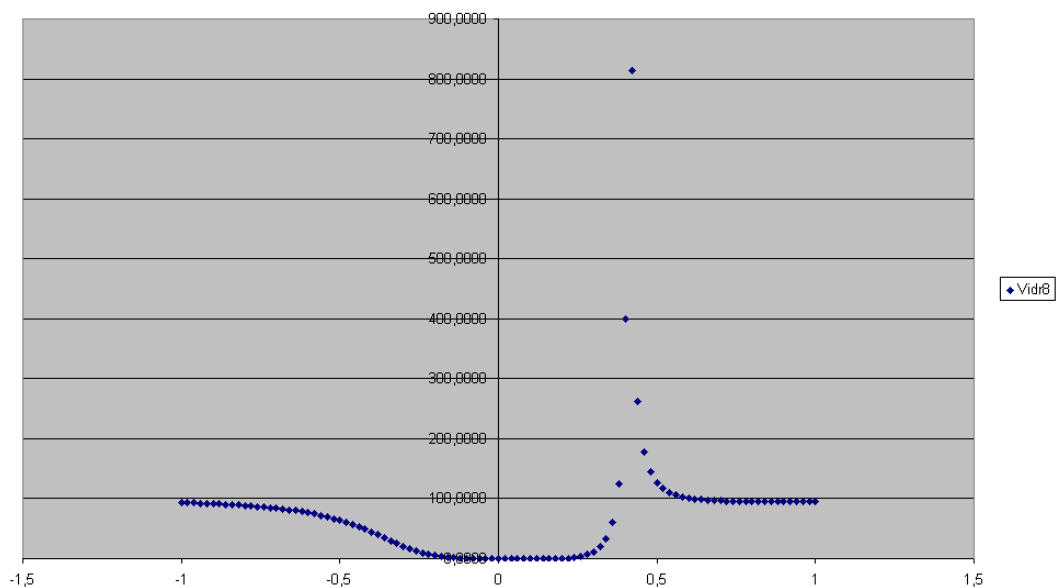


Рисунок 3 – Графік відносної похибки для 8 розрядної сітки

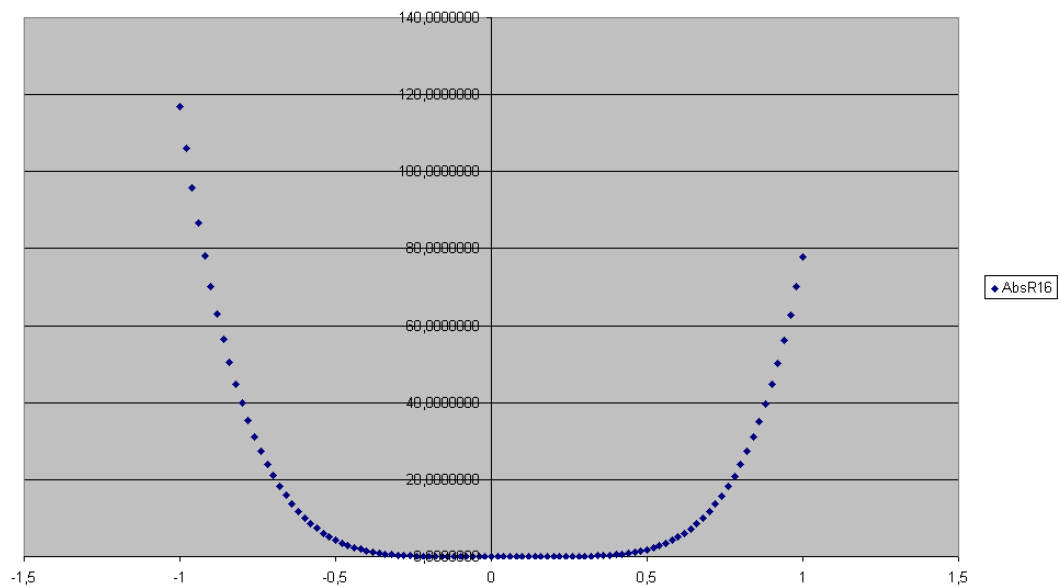


Рисунок 4 – Графік абсолютної похибки для 16 розрядної сітки

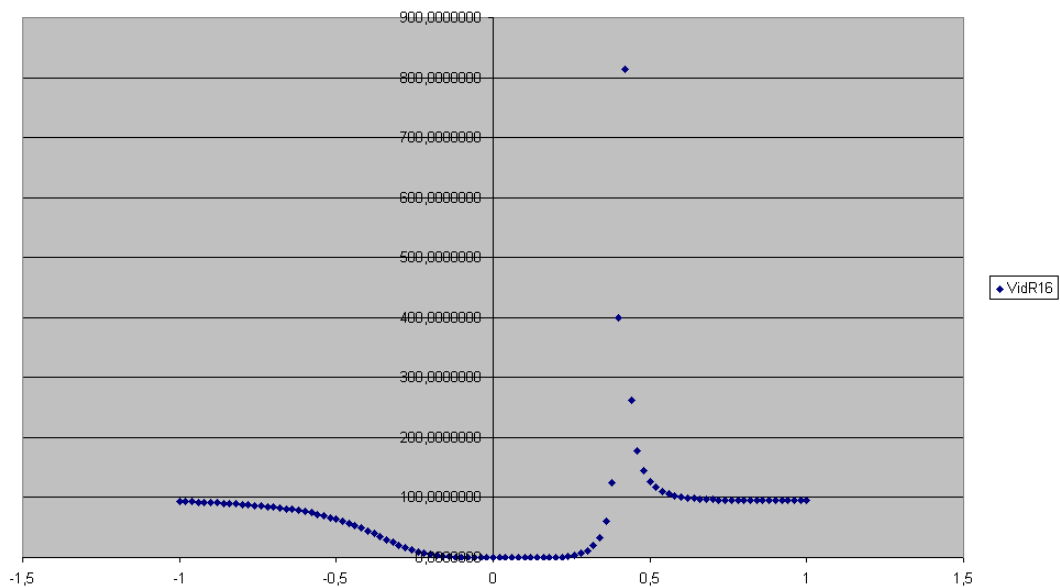


Рисунок 5 – Графік відносної похибки для 16 розрядної сітки

Висновок: в лабораторній роботі досліджено шляхи виникнення обчислювальної похибки та її вплив на точність обчислень. Проаналізувавши похибки, отримані при обчисленні заданої математичної функції, можна стверджувати, що для 16 розрядної сітки значення стрімкіше прямують до нуля, особливо в околі нульового аргументу, тоді як для 8 розрядного представлення збіг до нуля є менш вираженим, що свідчить про значну втрату точності обчислень.

Текст програми

```
// LABA 1Dlg.cpp : implementation file
```

```
#include "stdafx.h"
#include "LABA 1.h"
#include "LABA 1Dlg.h"
#include "XML WRITE/MyXmlWriter.h"
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#ifdef _DEBUG
#define new DEBUG_NEW
#endif

double vidsiktu_4_rozrjada(double a)
{
    int i =(int) a*10000;
    return (i/10000);
}

CLABA1Dlg::CLABA1Dlg(CWnd* pParent
/*=NULL*/)
: CDialog(CLABA1Dlg::IDD, pParent)
{
    m_hIcon = AfxGetApp()-
>LoadIcon(IDR_MAINFRAME);
}

void
CLABA1Dlg::DoDataExchange(CDataExchange*
pDX)
{
    CDialog::DoDataExchange(pDX);
    DDX_Control(pDX, IDC_LIST1, List);
}

BEGIN_MESSAGE_MAP(CLABA1Dlg, CDialog)
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    //}AFX_MSG_MAP
    ON_BN_CLICKED(IDC_BUTTON1,
&CLABA1Dlg::OnBnClickedButton1)
    ON_BN_CLICKED(IDC_BUTTON2,
&CLABA1Dlg::OnBnClickedButton2)
END_MESSAGE_MAP()

// CLABA1Dlg message handlers

BOOL CLABA1Dlg::OnInitDialog()
{
    CDialog::OnInitDialog();
    // Set the icon for this dialog. The framework does this
    //automatically
    // when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE); // Set big icon
    SetIcon(m_hIcon, FALSE); // Set small icon
    //ShowWindow(SW_MINIMIZE);
    // TODO: Add extra initialization here
    List.InsertColumn(0,_T("N"),LVCFMT_LEFT,35,20);
    List.InsertColumn(1,_T("X"),LVCFMT_LEFT,50,20);
    List.InsertColumn(2,_T("Еталоне Y"),
LVCFMT_LEFT,80,20);
```

```
List.InsertColumn(3,_T("8 розрядів Y"),
VCFMT_LEFT,0,20);
List.InsertColumn(4,_T("16 розрядів Y"),
LVCFMT_LEFT,130,20);
List.InsertColumn(5,_T("8 Абсолютна похибка"),
LVCFMT_LEFT,130,20);
List.InsertColumn(6,_T("16 Абсолютна похибка"),
LVCFMT_LEFT,130,20);
List.InsertColumn(7,_T("8 Відносна похибка"),
LVCFMT_LEFT,110,20);
List.InsertColumn(8,_T("16 Відносна похибка"),
LVCFMT_LEFT,125,20);
```

```
List.SetExtendedStyle(LVS_EX_GRIDLINES|LVS_E
X_FULLROWSELECT);
return TRUE; // return TRUE unless you set the focus
//to a control
}
```

```
// If you add a minimize button to your dialog, you will
//need the code below
// to draw the icon. For MFC applications using the
//document/view model,
// this is automatically done for you by the framework.
```

```
void CLABA1Dlg::OnPaint()
{
    if (IsIconic())
    {
        CPaintDC dc(this);
        SendMessage(WM_ICONERASEBKGND,
reinterpret_cast<LPARAM>(dc.GetSafeHdc()), 0);
        // Center icon in client rectangle
        int cxIcon =
GetSystemMetrics(SM_CXICON);
        int cyIcon =
GetSystemMetrics(SM_CYICON);
        CRect rect;
        GetClientRect(&rect);
        int x = (rect.Width() - cxIcon + 1) /
2;
        int y = (rect.Height() - cyIcon + 1) /
2;

        // Draw the icon
        dc.DrawIcon(x, y, m_hIcon);
    }
    else
    {
        CDialog::OnPaint();
    }
}
```

```
// The system calls this function to obtain the cursor to
display while the user drags
// the minimized window.
HCURSOR CLABA1Dlg::OnQueryDragIcon()
{
    return static_cast<HCURSOR>(m_hIcon);
}
```

```

void CLABA1Dlgl::Osnova (int X1, int X2, double m,
double dx)
{
    int Ozn,fio;
    char buf[255];
    char buf1[255];
    char buf3[255];
    double Xmax;
    Ozn=0;
    fio=1;
x=X1+vidsiktu_4_rozrjada(dx);
    Xmax=X2+dx;
    f1=((m*(m-1))*(m-5+1));
    itoa(X1,buf3,10);
    List.InsertItem(0,_T(" "));
    while ((x!=Xmax)&&(x<=Xmax))
    {
        List.InsertItem(Ozn,_T(" "));
        sprintf(buf1,"%1.4f",x);
        if (!strcmp(buf1,_T("-0.0000")))
        {
            x=0.0000;
            sprintf(buf1,"%1.4f",x);
        }
        List.SetItemText(Ozn,1,buf1);
        X[Ozn]=x;
        itoa(fio,buf,10);
        Obchislenja(Ozn,fio,x,m);
        VismRozradiv(Ozn,fio,x,m);
        ShisnadcjatRozradiv(Ozn,fio,x,m);
        x=x+dx;
        Ozn++;
        fio++;
    }
    index=Ozn;
    SaveData();
}

void CLABA1Dlgl::Obchislenja (int index,int
number,double X, double m)
{
    double rez;
    char buf2[255];
    char buf[255];

    rez = 1 +(-(((f1)/1)*X)+(((f1)/1*2)*(X*X))-
    (((f1)/1*2*3)*(X*X*X))+(((f1)/1*2*3*4)*(X*X*X*X
    ))-(((f1)/1*2*3*4*5)*(X*X*X*X*X)));
    sprintf(buf2,"%f",rez);
    Etalon[index]=atof(buf2);
    itoa(number,buf,10);
    List.SetItemText(index,0,buf);
    List.SetItemText(index,2,buf2);
}

void CLABA1Dlgl::VismRozradiv(int ind,int
fio,double X, double m)
{
    char buf[255];
    char buf2[255];
    char bufer[255];
    char bufer1[255];
    char bufer2[255];
    char bufer3[255];
    char bufer4[255];

```

```

    char bufer5[255];
    char bufer6[255];
    char bufer7[255];
    char bufer8[255];
    char bufer9[255];
    double rez;
    float f2; //X
    float f3;
    float f4; //X*X
    float f5;
    float f6; //X*X*X
    float f7;
    float f8; //X*X*X*X
    float f9;
    float f10; //X*X*X*X*X
    float f11;
    float f12; //1
    float f13;
    float f14; //2
    float f15;
    float f16; //3
    float f17;
    float f18; //4
    float f19;
    float f20; //5
    float f21;
    f21=X;
    sprintf(bufer,"%1.8f",f21);
    f2=atof(bufer);
    f3=f2*f2;
    sprintf(bufer1,"%1.8f",f3);
    f4=atof(bufer1);
    f5=f4*f2;
    sprintf(bufer2,"%1.8f",f5);
    f6=atof(bufer2);
    f7=f6*f2;
    sprintf(bufer3,"%1.8f",f7);
    f8=atof(bufer3);
    f9=f8*f2;
    sprintf(bufer4,"%1.8f",f9);
    f10=atof(bufer4);
    f11 = (f1/1)*f2;
    sprintf(bufer5,"%1.8f",f11);
    f12=atof(bufer5);
    f13 = (f1/1*2)*(f4);
    sprintf(bufer6,"%1.8f",f13);
    f14=atof(bufer6);
    f15 = (f1/1*2*3)*(f6);
    sprintf(bufer7,"%1.8f",f15);
    f16=atof(bufer7);
    f17 = (f1/1*2*3*4)*(f8);
    sprintf(bufer8,"%1.8f",f17);
    f18=atof(bufer8);
    f19 = (f1/1*2*3*4)*(f8);
    sprintf(bufer9,"%1.8f",f19);
    f20=atof(bufer9);
    rez = 1 +(-f12)+(f14)-(f16)+(f18)-(f20));
    itoa(fio,buf,10);
    sprintf(buf2,"%1.4f",rez);
    R8[ind]=atof(buf2);
    List.SetItemText(ind,0,buf);
    List.SetItemText(ind,3,buf2);
    VismRozradivPoh(ind, fio);
}

```

```

void CLABA1Dlgl::ShisnadcjatRozradiv(int ind,int
fio,double X, double m)
{
    char buf[255];
    char buf2[255];
    char bufer[255];
    char bufer1[255];
    char bufer2[255];
    char bufer3[255];
    char bufer4[255];
    char bufer5[255];
    char bufer6[255];
    char bufer7[255];
    char bufer8[255];
    char bufer9[255];
    double rez;
    float f2; //X
    float f3;
    float f4; //X*X
    float f5;
    float f6; //X*X*X
    float f7;
    float f8; //X*X*X*X
    float f9;
    float f10; //X*X*X*X*X
    float f11;
    float f12; //1
    float f13;
    float f14; //2
    float f15;
    float f16; //3
    float f17;
    float f18; //4
    float f19;
    float f20; //5
    float f21;
    f21=X;
    sprintf(bufer,"% 1.16f",f21);
    f2=atof(bufer);
    f3=f2*f2;
    sprintf(bufer1,"% 1.16f",f3);
    f4=atof(bufer1);
    f5=f4*f2;
    sprintf(bufer2,"% 1.16f",f5);
    f6=atof(bufer2);
    f7=f6*f2;
    sprintf(bufer3,"% 1.16f",f7);
    f8=atof(bufer3);
    f9=f8*f2;
    sprintf(bufer4,"% 1.16f",f9);
    f10=atof(bufer4);
    f11 = (f1/1)*f2;
    sprintf(bufer5,"% 1.16f",f11);
    f12=atof(bufer5);
    f13 = (f1/1*2)*(f4);
    sprintf(bufer6,"% 1.16f",f13);
    f14=atof(bufer6);
    f15 = (f1/1*2*3)*(f6);
    sprintf(bufer7,"% 1.16f",f15);
    f16=atof(bufer7);
    f17 = (f1/1*2*3*4)*(f8);
    sprintf(bufer8,"% 1.16f",f17);
    f18=atof(bufer8);
    f19 = (f1/1*2*3*4)*(f8);
    sprintf(bufer9,"% 1.16f",f19);

```

```

    f20=atof(bufer9);
    rez = 1 +(-(f12)+(f14)-(f16)+(f18)-(f20));
    itoa(fio,buf,10);
    sprintf(buf2,"% 1.7f",rez);
    R16[ind]=atof(buf2);
    List.SetItemText(ind,0,buf);
    List.SetItemText(ind,4,buf2);
    ShisnadcjatRozradivPoh(ind, fio);
}

void CLABA1Dlgl::VismRozradivPoh(int ind,int fio)
{
    char buf[255];
    char buf2[255];
    double osn;
    osn = fabs(Etalon[ind]-R8[ind]);
    itoa(fio,buf,10);
    sprintf(buf2,"% 1.4f",osn);
    AbsR8[ind]=atof(buf2);
    List.SetItemText(ind,0,buf);
    List.SetItemText(ind,5,buf2);
    VismRozradivPoh2(ind, fio,osn);
}

void CLABA1Dlgl::ShisnadcjatRozradivPoh(int ind,int
fio)
{
    char buf[255];
    char buf2[255];
    double osn;
    osn = fabs(Etalon[ind]-R16[ind]);
    itoa(fio,buf,10);
    sprintf(buf2,"% 1.7f",osn);
    AbsR16[ind]=atof(buf2);
    List.SetItemText(ind,0,buf);
    List.SetItemText(ind,6,buf2);
    ShisnadcjatRozradivPoh2(ind, fio,osn);
}

void CLABA1Dlgl::VismRozradivPoh2(int ind,int
fio,double rez)
{
    char buf[255];
    char buf2[255];
    double Riz;
    Riz=fabs((AbsR8[ind]/Etalon[ind])*100);
    itoa(fio,buf,10);
    sprintf(buf2,"% 1.4f",Riz);
    VidR8[ind]=atof(buf2);
    List.SetItemText(ind,0,buf);
    List.SetItemText(ind,7,buf2);
}

void CLABA1Dlgl::ShisnadcjatRozradivPoh2(int
ind,int fio,double rez)
{
    char buf[255];
    char buf2[255];
    double Riz;
    Riz=fabs((AbsR16[ind]/Etalon[ind])*100);
    itoa(fio,buf,10);
    sprintf(buf2,"% 1.7f",Riz);
    VidR16[ind]=atof(buf2);
    List.SetItemText(ind,0,buf);
    List.SetItemText(ind,8,buf2);
}

void CLABA1Dlgl::OnBnClickedButton1()
{

```

```

X1=GetDlgItemInt(IDC_EDIT2,0,1);
X2=GetDlgItemInt(IDC_EDIT3,0,1);
char b[255];
char c[255];
GetDlgItemText(IDC_EDIT4,(LPTSTR)
b,255);
GetDlgItemText(IDC_EDIT1,(LPTSTR)
c,255);
m = atof(b);
dx = atof(c);
Osnova(X1,X2,m,dx);
}

void CLABA1Dlg::SaveData()
{
    char buf1[255];
    char buf2[255];
    int num,ind;
    ind = 0;
    num = 1;
    MyXmlWriter MyXml("Rezultat.xml");
    MyXml.CreateTag("X");
    while
((ind<=index)&&(ind!=index))
    {

        sprintf(buf1,"% 1.4f",X[ind]);
        sprintf(buf2,"%d",num);

        MyXml.CreateChild(buf2,buf1);
                                num++;
                                ind++;
    }
    ind = 0;
    num = 1;
    MyXml.CloseLastTag();
    MyXml.CreateTag("Etalon");
    while
((ind<=index)&&(ind!=index))
    {
        sprintf(buf1,"% 1.15f",Etalon[ind]);

        sprintf(buf2,"%d",num);

        MyXml.CreateChild(buf2,buf1);
                                num++;
                                ind++;
    }
    ind = 0;
    num = 1;
    MyXml.CloseLastTag();
    MyXml.CreateTag("R8");
    while
((ind<=index)&&(ind!=index)) {
        sprintf(buf1,"% 1.6f",R8[ind]);
        sprintf(buf2,"%d",num);
        MyXml.CreateChild(buf2,buf1);
                                num++;
                                ind++;
    }
    ind = 0;
    num = 1;
    MyXml.CloseLastTag();
}

```

```

MyXml.CreateTag("R16");
while
((ind<=index)&&(ind!=index)){
    sprintf(buf1,"% 1.8f",R16[ind]);
    sprintf(buf2,"%d",num);
    MyXml.CreateChild(buf2,buf1);
                                num++;
                                ind++;
    }
    ind = 0;
    num = 1;
    MyXml.CloseLastTag();
    MyXml.CreateTag("AbsR8");
    while ((ind<=index)&&(ind!=index)) {
        sprintf(buf1,"% 1.10f",AbsR8[ind]);
        sprintf(buf2,"%d",num);
        MyXml.CreateChild(buf2,buf1);
                                num++;
                                ind++;
    }
    ind = 0;
    num = 1;
    MyXml.CloseLastTag();
    MyXml.CreateTag("AbsR16");
    while
((ind<=index)&&(ind!=index)) {
        sprintf(buf1,"% 1.10f",AbsR16[ind]);
        sprintf(buf2,"%d",num);
        MyXml.CreateChild(buf2,buf1);
                                num++;
                                ind++;
    }
    ind = 0;
    num = 1;
    MyXml.CloseLastTag();
    MyXml.CreateTag("VidR8");
    while
((ind<=index)&&(ind!=index)){
        sprintf(buf1,"% 1.10f",VidR8[ind]);
        sprintf(buf2,"%d",num);
        MyXml.CreateChild(buf2,buf1);
                                num++;
                                ind++;
    }
    ind = 0;
    num = 1;
    MyXml.CloseLastTag();
    MyXml.CreateTag("VidR16");
    while ((ind<=index)&&(ind!=index))
    {sprintf(buf1,"% 1.12f",VidR16[ind]);
        sprintf(buf2,"%d",num);
        MyXml.CreateChild(buf2,buf1);
                                num++;
                                ind++;
    }
    ind = 0;
    num = 1;
    MyXml.CloseLastTag();
    MyXml.CloseTag();
}

void CLABA1Dlg::OnBnClickedButton2()
{
    List.DeleteAllItems(); }

```

Лабораторна робота 2

Тема. Розрахунок і побудова цифрових СІХ фільтрів з частотною вибіркою. Фільтрація складених сигналів.

Мета роботи: Ознайомитись з різними типами цифрових фільтрів, навчитись розраховувати різні типи фільтрів і застосовувати їх на практиці. Дослідити використання вагових функцій при побудові частотних фільтрів з скінченною імпульсною характеристикою.

Теоретичні відомості

Фільтр — це система, що вибірково змінює форму сигналу (амплітудно-частотну або фазово-частотну характеристику). Основними цілями фільтрації є: покращання якості сигналу, виділення із сигналів інформації або розділення, об'єднаних раніше, сигналів для, наприклад, ефективного використання доступного каналу зв'язку.

Важливу роль в цифровому опрацюванні сигналів відіграють цифрові фільтри. В порівнянні з аналоговими фільтрами вони переважають у багатьох областях (стиск даних, біомедичне опрацювання сигналів, опрацювання мови, опрацювання зображень, передача даних, цифрове аудіо, телефонне ехопослаблення), так як володіють рядом переваг, частина з яких описана нижче:

- цифрові фільтри(ЦФ) мають характеристики, отримати які на аналогових фільтрах (АФ) неможливо (лінійна фазова характеристика);
- на відміну від АФ, продуктивність ЦФ не залежить від змін середовища (зміна температури, вологості, тиску). Таким чином ЦФ не потребують періодичного калібрування;
- якщо фільтр(Ф) побудований з використанням програмованого процесора, його частотна характеристика може налаштовуватись автоматично (такі процесори широко застосовуються а адаптивних фільтрах);
- один ЦФ може опрацьовувати декілька вхідних сигналів або каналів без дублювання апаратних блоків;
- фільтровані та не фільтровані дані можна зберігати для наступного використання;
- можна легко використовувати здобутки із області технологій НВІС і отримати невеликі ЦФ з пониженою потужністю споживання і більш низькою ціною;
- на практиці точність, яку можна отримати при використанні АФ обмежена. Точність цифрових фільтрів обмежена довжиною слова використаних даних;
- продуктивність ЦФ однакова для всіх пристроїв серії;
- ЦФ можуть використовуватись при дуже низьких частотах(біомедичні дослідження). Крім того, ЦФ можуть використовуватись у великому діапазоні частот, для цього достатньо просто змінювати частоту дискретизації.

Проте, в порівнянні з АФ, ЦФ мають і ряд недостатків:

- обмеження швидкості, максимальна ширина смуги сигналів, які в реальному часі спроможні опрацьовувати ЦФ, значно вужча, ніж у АФ. В задачах реального часу процес перетворення “аналоговий – цифровий – аналоговий” вводить обмеження по швидкості на продуктивність ЦФ. Найвищу частоту дискретизації, з якою може працювати фільтр, обмежує час конвертації АЦП і час встановлення сигналу ЦАП. Крім того, швидкість роботи ЦФ залежить від швидкості роботи цифрового процесора і числа арифметичних операцій, які необхідно виконати в алгоритмі фільтрації, і збільшується, коли характеристика фільтра стає більш стиснутою.

- вплив кінцевої розрядності. ЦФ піддаються впливу шуму АЦП, що виникає при квантуванні неперервного сигналу, і шуму заокруглення, який виникає при обчисленнях. При використанні рекурсивних фільтрів високих порядків накопичення шуму заокруглення може призвести до нестійкості фільтра.
- значний час розробки і реалізації. Розробка і реалізація ЦФ, особливо реалізація апаратного забезпечення можуть виконуватись набагато довше, ніж подібні процедури для АФ.

Типи цифрових фільтрів

ЦФ поділені на два великі класи: фільтри з нескінченною імпульсною характеристикою (НІХ-фільтри) і фільтри з скінченною імпульсною характеристикою (СІХ-фільтри). В спрощеній формі, як показано на рис.1, фільтр кожного типу можна представити через коефіцієнти його імпульсної характеристики $h(k)(k=0,1,\dots)$. Вхідний і вихідний сигнали фільтра зв'язані через операцію згортки, даний зв'язок наведений у виразі (1) для НІХ- фільтра, і у виразі (2) для СІХ – фільтра.

$$y(n) = \sum_{k=0}^{\infty} h(k)x(n-k) \quad (1)$$

$$y(n) = \sum_{k=0}^N h(k)x(n-k) \quad (2)$$

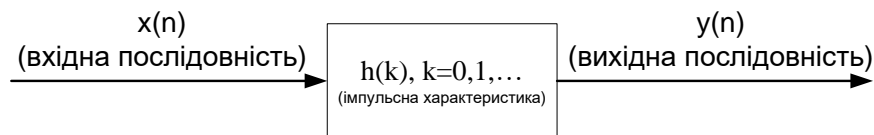


Рисунок 1 – Спрощена схема ЦФ

Для НІХ – фільтрів імпульсна характеристика має безкінечну довжину, тоді як для СІХ – фільтра вона скінченна, оскільки $h(k)$ для СІХ-фільтрів може приймати всього N значень. На практиці вивести вихід НІХ - фільтра з використанням рівності (1) неможливо, оскільки довжина імпульсного відгуку надто велика (теоретично – нескінченна). Замість цього рівняння НІХ – фільтрації переписується в рекурсивній формі

$$y(n) = b_0x(n) + b_1x(n-1) + \dots + b_Px(n-P) - a_1y(n-1) - a_2y(n-2) - \dots - a_Qy(n-Q) \quad (3)$$

де a_k і b_k - коефіцієнти фільтра. Дана рівність (значення $h(k)$ для СІХ-фільтра або a_k і b_k для НІХ- фільтра) використовується в багатьох задачах розробки фільтрів. Варто відмітити, що у виразі (3) поточна вихідна вибірка $y(n)$ являється функцією попередніх виходів, а також поточної і попередніх вхідних вибірок. Таким чином НІХ-фільтр — це в деякому вигляді система із зворотнім зв'язком. Якщо взяти всі b_k рівними нулю, то вираз (3) зводиться до рівності (2).

$$y(n) = b_0x(n) + b_1x(n-1) + \dots + b_Px(n-P) \quad (4)$$

де, P — порядок фільтра, $x(n)$ — вхідний сигнал, $y(n)$ — вихідний сигнал, а b_i — коефіцієнти фільтра.

$$y(n) = b_0x(n) + b_1x(n-1) + \dots + b_Px(n-P) - a_1y(n-1) - a_2y(n-2) - \dots - a_Qy(n-Q) \quad (5)$$

де, P — порядок вхідного сигналу, b_i — коефіцієнти вхідного сигналу, Q — порядок зворотнього зв'язку(порядок фільтра) , a_i — коефіцієнти зворотнього зв'язку , $x(n)$ — вхідний, а $y(n)$ — вихідний сигнали.

Вирази (4) та (5) відповідають рівнянням СІХ на НІХ фільтрів відповідно.

Використання цифрових фільтрів

ЦФ набули широкого використання у задачах частотної фільтрації. Розрізняють такі частотні фільтри:

- Фільтр низьких частот (ФНЧ) — фільтр, що ефективно пропускає частотний спектр сигналу нижче деякої частоти (частота зрізу), і зменшує(або послаблює) частоти сигналу вище цієї частоти. Степінь послаблення кожної частоти залежить від виду фільтра.
- Фільтр верхніх частот (ФВЧ) — фільтр, що пропускає високі частоти вхідного сигналу, при цьому послаблює частоти сигналу менші, ніж частота зрізу. Степінь послаблення залежить от конкретного виду фільтра.
- Смуговий фільтр — фільтр, який пропускає частоти, що знаходяться в потрібному діапазоні і вирізує всі решта частоти. Такі фільтри також можуть бути виготовлені комбінуванням ФНЧ і ФВЧ.
- Загороджувальний фільтр (режекторний фільтр) — фільтр, що не пропускає коливання деякого визначеного діапазону частот, і пропускає коливання з частотами, що виходять за межі цього діапазону. Загороджувальний фільтр, призначений для послаблення одної визначеної частоти, називається вузькосмуговим загороджувальним фільтром або фільтром-пробкою.

На рис.2 наведена ідеальна АЧХ описаних типів фільтрів

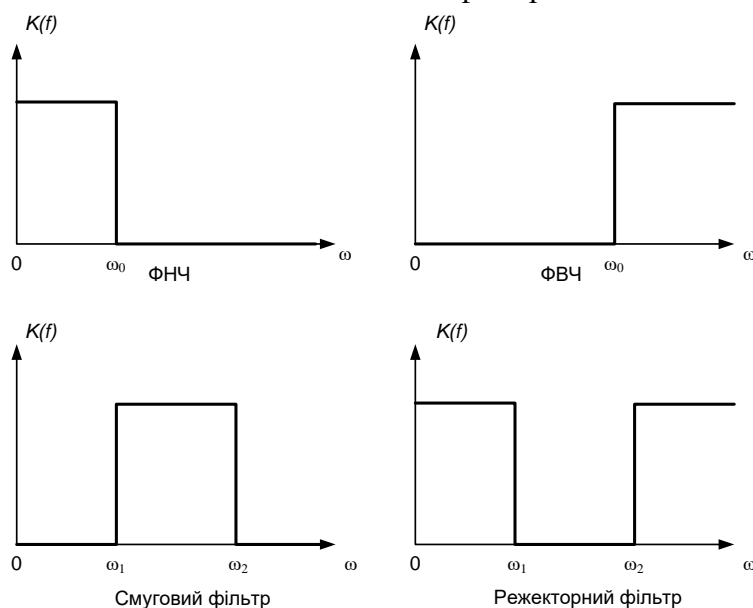


Рисунок 2 – Ідеальна АЧХ різних типів фільтрів

Методи розрахунку цифрових фільтрів

На практиці розрахунок цифрових фільтрів полягає в знаходженні коефіцієнтів відповідних СІХ або НІХ фільтра, які відповідають заданим параметрам (частота чи смуга пропускання фільтра, дискретизація вхідного сигналу, порядок фільтра, послаблення пульсацій і т.д.). Відповідно, для НІХ і СІХ фільтрів існують окремі методи розрахунку коефіцієнтів.

Розрахунок коефіцієнтів НІХ фільтрів виконують за допомогою таких методів:

- метод розміщення полюсів і нулів;
- метод інваріантного перетворення імпульсної характеристики;

- метод узгодженого z-перетворення;
- метод білінійного z-перетворення;
- метод білінійного z-перетворення і класичних аналогових фільтрів;
- метод розміщення нулів і полюсів на s-площині і т.д.
- Розрахунок коефіцієнтів СІХ фільтрів виконують за допомогою таких методів:
- метод зважування(вирізування);
- оптимізовані методи;
- метод частотної вибірки і т.д.

Кожен із запропонованих методів знайшов своє використання залежно від заданих параметрів.

Розрахунок коефіцієнтів СІХ фільтра за допомогою методу зважування (вирізування)

СІХ фільтр характеризується наступними рівностями:

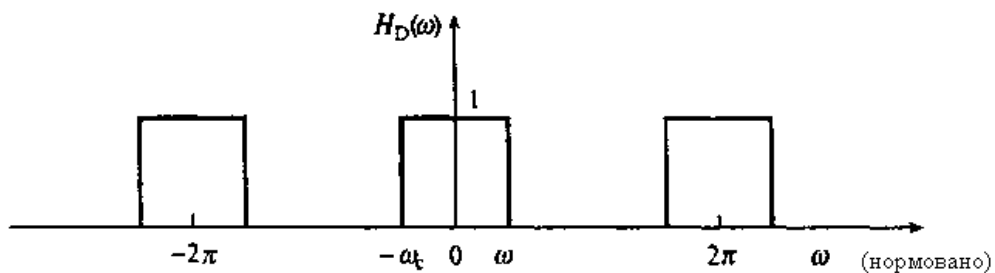
$$y(n) = \sum_{m=0}^{N-1} x(m)h(n-m),$$

$$Y(z) = \sum_{n=0}^{N-1} h(n)z^{-n}.$$

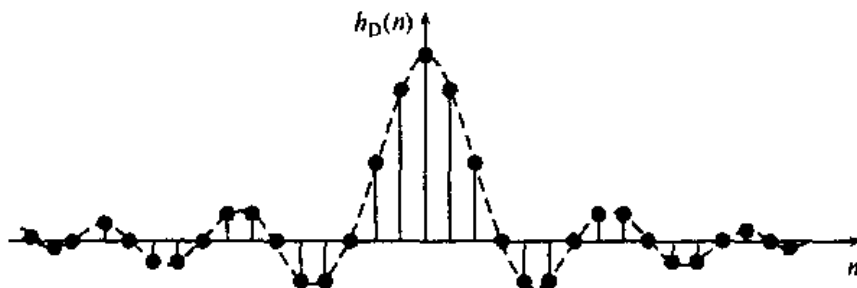
В даному методі використовується факт, що частотна характеристика фільтра $H_D(\omega)$ і відповідна імпульсна характеристика $h_D(n)$ зв'язані зворотнім перетворенням Фур'є:

$$H_D(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} h_D(n) e^{j\omega n} dn \quad (6)$$

Якщо $H_D(\omega)$ відома, $h_D(n)$ можна отримати, застосувавши перетворення Фур'є до обох частин рівності (6). Припустимо, що необхідно розробити ФНЧ. Розглянемо ідеальну імпульсну характеристику, представлену на рис.3(а)



а)



б)

Рисунок 3 – Ідеальна частотна характеристика ФНЧ (а),
Імпульсна характеристика ідеального ФНЧ (б).

Припустимо, що характеристика змінюється від $-\omega_c$ до ω_c тоді спрощуючи інтегрування отримуємо наступну імпульсну характеристику:

$$\begin{aligned}
 h_D(n) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} 1 \cdot e^{jn\omega} d\omega = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{jn\omega} d\omega \\
 &= \frac{2f_c \sin(n\omega_c)}{n\omega_c} \\
 &= 2f_c \text{ for } n=0
 \end{aligned} \tag{7}$$

Імпульсні характеристики ідеальних фільтрів верхніх частот, смугових та загороджувальних фільтрів знаходяться на основі рівності (7) і всі вони наведені в таблиці 1.

Таблиця 1 Імпульсні характеристики ідеальних фільтрів

Тип фільтра	$h_D(n), n \neq 0$	$h_D(n), n=0$
ФНЧ	$\frac{2f_c \sin(n\omega_c)}{n\omega_c}$	$2f_c$
ФВЧ	$\frac{2f_c \sin(n\omega_c)}{n\omega_c}$	$1 - 2f_c$
Смуговий фільтр	$\frac{2f_2 \sin(n\omega_2)}{n\omega_2} \frac{2f_1 \sin(n\omega_1)}{n\omega_1}$	$2(f_2 - f_1)$
Загороджувальний фільтр	$\frac{2f_2 \sin(n\omega_2)}{n\omega_2} \frac{2f_1 \sin(n\omega_1)}{n\omega_1}$	$1 - 2(f_2 - f_1)$

Оскільки імпульсна характеристика $h_D(n)$ зменшується при віддалені від точки $n = 0$, то теоретично вона іде до $n = \pm\infty$. Відповідно, отриманий фільтр не є СІХ фільтром. Для вирішення цієї ситуації пропонується обрізати ідеальну імпульсну характеристику, взявши $h_D(n) = 0$ для n , більших, ніж певне M . При цьому вводиться небажана нерівномірність і викиди — має місце так званий ефект Гобса. Чим більше коефіцієнтів залишилось, тим ближче спектр фільтра до ідеальної характеристики. Обрізання характеристики $h_D(n)$ рівнозначне перемноженню ідеальної імпульсної характеристики на прямокутну вагову функцію виду:

$$\begin{aligned}
 w(n) &= \begin{cases} 1 & \text{for } |n| \leq M \\ 0 & \text{for } |n| > M \end{cases} \\
 &= \text{rect}(n/(2M+1))
 \end{aligned}$$

В частотній області це еквівалентно згортці $H_d(\omega)$ з $W(\omega)$, де $W(\omega)$, — Фур'є перетворення $w(n)$.

На практиці ідеальна частотна характеристика h_D множиться на відповідну вагову функцію $w(n)$.

Зв'язок ширини переходу (від смуги пропускання до смуги послаблення) фільтра побудованого на основі певної вагової функції, з довжиною фільтра виражається залежністю:

$$\Delta f = k/N, \tag{8}$$

Де N — довжина фільтра, k — коефіцієнт який розрахований для кожного типу вікна, а Δf — нормована ширина смуги переходу.

Характеристики основних вагових функцій зібрані в таблиці 2.

Таблиця 2 Характеристики основних вагових функцій

Тип вікна	Рівень бокових пелюсток, дБ	Ширина перехідної зони(N=256)	Аналітичний вираз
Прямокутне	-21,1	0,0053	$w[n]=1$
Тюкі	-22	0,0072	$w[n] = \left[\cos\left(\frac{\pi n}{2N}\right) \right]^2$
Трикутне	-26,5	0,0187	$w[n] = 1 - \frac{ n }{N}$
Ханна	-43,9	0,0156	$w[n] = \frac{1}{2} \left(1 + \cos\left(\frac{\pi n}{2N}\right) \right)^2$
Бомана	-51,95	0,0274	$w[n] = \left(\frac{1}{2} \left(1 + \cos\left(\frac{\pi n}{2N}\right) \right) \right)^2$
Геммінга	-53,8	0,0161	$w[n] = \left(\frac{1}{2} \left(1 + \cos\left(\frac{\pi n}{2N}\right) \right) \right)^2$
Парзена	-56,6	0,0354	$w[n] = \left(\frac{1}{2} \left(1 + \cos\left(\frac{\pi n}{2N}\right) \right) \right)^2$
Гауса	-59,2	0,0237	$w[n] = \exp\left(-\frac{1}{2} \left(\frac{n}{N} \right)^2\right)$
Блекмена	-75,3	0,0258	$w[n] = 0,42 - 0,5 \cos \frac{\pi n}{N} + 0,08 \cos \frac{4\pi n}{N}$
Наттолла	-113,9	0,0350	$w[n] = a_0 + a_1 \cos\left(\frac{\pi n}{N}\right) + a_2 \cos\left(\frac{2\pi n}{N}\right) + a_3 \cos\left(\frac{3\pi n}{N}\right)$ $a_0=0,635$ $a_1=0,4891$ $a_2=0,1365$ $a_3=0,106$ $t_n = (n - [N]/2)/[N], n=[0, N-1]$

Для якісної розробки фільтрів необхідно задати певні специфікації відносно фазової та амплітудно-частотної характеристики. При розгляді фазової характеристики достатньо вказати, яка необхідна симетрія – пара чи не парна (мається на увазі, що фазова характеристика лінійна). Амплітудно-частотна характеристика СЧХ – фільтра часто задається у вигляді схеми допусків. Така схема для ФНЧ нижніх частот зображена на рис.4. Подібну схему можна легко отримати і для інших частотно-вибіркових фільтрів. Виходячи із наведеного нижче рисунку особливий інтерес представляють такі параметри:

δ_p – відхилення в смузі пропускання (нерівномірність);

δ_s – відхилення в смузі послаблення;
 f_p – гранична частота смуги пропускання;
 f_s – гранична частота смуги послаблення;
 F_s – частота дискретизації.

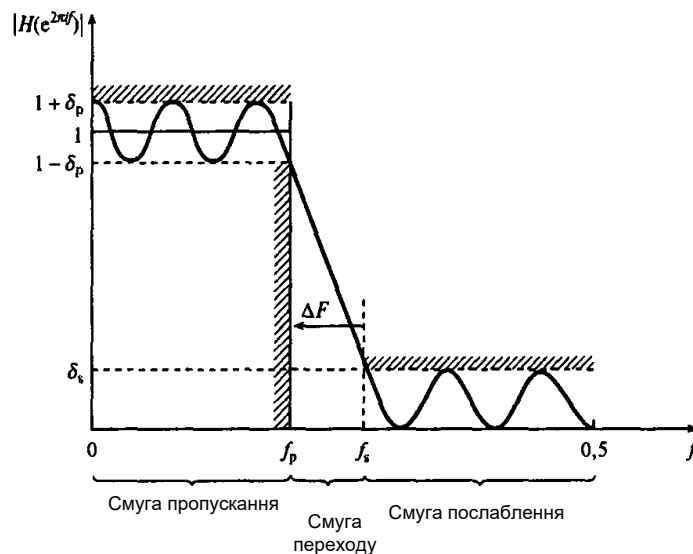


Рисунок 4 – Схема допуску для ФНЧ

Порядок виконання роботи

1. Вибрати ідеальну частотну характеристику фільтра $H_D(\omega)$ згідно варіанту.
2. Обчислити імпульсну характеристику $h_p(n)$ фільтра згідно варіанту. Для цього потрібно обрахувати перетворення Фур'є частотної характеристики заданого у завданні фільтра за виразом (7).
3. Вибрати вагову функцію, яка задовольняє вимогам до смуги пропускання або послаблення, а потім визначити число коефіцієнтів фільтра, використавши відповідний вираз для зв'язку довжини фільтра з шириною переходу, Δf (обчислити з врахуванням частоти дискретизації).
4. Отримати значення вибраної вагової функції $w(n)$ і значення коефіцієнтів СІХ фільтра $h(n)$, на основі виразу

$$h(n) = h_p(n) w(n)$$

5. Сформувати вхідний сигнал згідно варіанту завдання.
6. Профільтрувати вхідний сигнал за виразом

$$y(n) = \sum_{m=0}^{N-1} h(m) x(n-m)$$

де N - порядок фільтра.

7. Для відфільтрованого та не відфільтрованого сигналу обчислити перетворення Фур'є і порівняти отримані результати.

Звіт до лабораторної роботи

- мета роботи;
- варіант завдання ;
- розрахунок фільтра;
- графіки:
 - ідеальна та реальна (обчислена) частотна характеристика заданого в завданні фільтра;
 - вихідний (відфільтрований) та вхідний(не відфільтрований) сигнали;
 - вагова функція;
 - перетворення Фур'є вхідного (не відфільтрованого) та вихідного (відфільтрованого) сигналів;
- висновки згідно отриманих результатів.

Контрольні запитання

1. Що таке цифровий фільтр?
2. Пояснити суть імпульсної та частотної характеристики цифрових фільтрів?
3. Характеристика СІХ, НІХ фільтрів?
4. Типи цифрових фільтрів частотної вибірки?
5. Яка різниця між ФНЧ, ФВЧ, смуговим та загороджувальним фільтрами?
6. Суть ефекту Гобса?
7. Переваги застосування цифрових фільтрів?
8. Вплив віконної обробки на параметри фільтра?
9. Методи розрахунку коефіцієнтів фільтра?

Варіанти завдань

Варіант	Вхідний сигнал, $t=0..1000$	Частоти вхідного сигналу, кГц, ω_n	Фільтр	Частоти фільтра, кГц	Ширина смуги переходу, кГц	Частота дискретизації, кГц	Затухання в смузі послаблення, дБ
1	$\sum_{n=1}^3 \sin(v_n t)$	2 3,2 6,1	ФНЧ	4	0,5	16	-21,1
2	$\sum_{n=1}^3 \cos(v_n t)$	2,4 4,0 7,0	ФВЧ	4,2	0,12	18	-22
3	$\sum_{n=1}^4 \sin(v_n t)$	1,8 2,9 4,6 7,3	Смуговий	2,1..6,8	0,08	28	-26,5
4	$\sum_{n=1}^4 \cos(v_n t)$	1,0 1,4 1,58 2,0	Загороджувальний	1,3..1,6	0,07	12	-43,9
5	$\sum_{n=1}^3 \sin(v_n t)$	0,5 1,93 2,9	ФНЧ	0,8	0,36	14	-51,95
6	$\sum_{n=1}^3 \cos(v_n t)$	2,5 3,0 7,0	ФВЧ	3,2	0,12	15	-53,8
7	$\sum_{n=1}^4 \sin(v_n t)$	3,0 5,1 5,26 8,2	Смуговий	4,8..5,3	0,19	23	-56,6
8	$\sum_{n=1}^4 \cos(v_n t)$	3,5 6,0 8,5 9,8	Загороджувальний	5,9..9,0	0,27	56	-59,2
9	$\sum_{n=1}^3 \sin(v_n t)$	1,6 3,3 5,1	ФНЧ	3,1	0,31	22	-75,3
10	$\sum_{n=1}^3 \cos(v_n t)$	1,2 6,1 7,9	ФВЧ	6,4	0,28	21	-113,9
11	$\sum_{n=1}^4 \sin(v_n t)$	1,0 2,2 2,4 3,7	Смуговий	2,1..2,6	0,12	29	-21,1
12	$\sum_{n=1}^4 \cos(v_n t)$	2,6 3,7 4,2 5,8	Загороджувальний	3,9..4,5	0,18	28	-22
13	$\sum_{n=1}^3 \sin(v_n t)$	2,6 6,85 7,9	ФНЧ	6,6	0,3	21	-26,5

14	$\sum_{n=1}^3 \cos(v_n t)$	0,2 0,7 2,6	ФВЧ	0,8	0,17	8	-43,9
15	$\sum_{n=1}^4 \sin(v_n t)$	0,2 1,8 5,3 8,3	Смуговой	0,8..6,6	0,23	34	-51,95
16	$\sum_{n=1}^4 \cos(v_n t)$	4,3 7,3 8,6 11,0	Загороджу вальний	6,9..9,8	0,3	42	-53,8
17	$\sum_{n=1}^3 \sin(v_n t)$	4,6 10,0 12,0	ФНЧ	9,8	0,19	28	-56,6
18	$\sum_{n=1}^3 \cos(v_n t)$	0,05 0,26 1,3	ФВЧ	0,2	0,05	6	-59,2
19	$\sum_{n=1}^4 \sin(v_n t)$	2,3 4,9 5,06 6,8	Смуговой	4,7..5,2	0,13	30	-75,3
20	$\sum_{n=1}^4 \cos(v_n t)$	0,5 1,4 1,59 2,6	Загороджу вальний	1,2..1,9	0,15	14	-113,9
21	$\sum_{n=1}^3 \sin(v_n t)$	6,0 9,1 9,9	ФНЧ	8,7	0,24	30	-21,1
22	$\sum_{n=1}^3 \cos(v_n t)$	0,06 0,26 0,8	ФВЧ	0,3	0,14	4	-22
23	$\sum_{n=1}^4 \sin(v_n t)$	4,8 5,1 5,16 6,0	Смуговой	5,0..5,2	0,02	25	-26,5
24	$\sum_{n=1}^4 \cos(v_n t)$	6,8 7,28 8,2 9,0	Загороджу вальний	7,2..8,6	0,04	31	-43,9
25	$\sum_{n=1}^3 \sin(v_n t)$	4,0 6,7 8,5	ФНЧ	6,2	0,16	22	-51,95
26	$\sum_{n=1}^3 \cos(v_n t)$	2,3 3,6 4,2	ФВЧ	3,4	0,17	12	-53,8
27	$\sum_{n=1}^4 \sin(v_n t)$	2,3 3,6 4,9 7,2	Смуговой	3,4..6,2	0,14	24	-56,6
28	$\sum_{n=1}^4 \cos(v_n t)$	2,6 4,8 5,1 6,2	Загороджу вальний	4,4..5,9	0,18	28	-59,2

29	$\sum_{n=1}^3 \sin(\omega_n t)$	2,1 5,9 6,8	ФНЧ	8,1	0,36	25	-75,3
30	$\sum_{n=1}^3 \cos(\omega_n t)$	1,9 3,1 5,7	ФВЧ	2,7	0,27	9	-113,9

Приклад виконання

Завдання

Вар.	Вхідний сигнал, $t=0..1000$	Частоти вхідного сигналу, кГц, ω_n	Фільтр	Частоти фільтра, кГц	Ширина смуги переходу, кГц	Частота дискретизації, кГц	Затухання в смузі послаблення, дБ
1	$\sum_{n=1}^3 \sin(\omega_n t)$	1,5 5,3 6,8	ФНЧ	5,2	0,5	16	-21,1

Використовуючи таблиці 1, виберемо $h_D(n)$ для ФНЧ:

$$h_D(n) = \frac{2f_c \sin(\omega_c n)}{\omega_c}, n \in \mathbb{Z}$$

Із таблиці 2, впливає що вимоги до затухання в смузі послаблення задовільняють функції прямокутного вікна. Тоді $\Delta f = 0,5016$. Для $N=256$ смуга пропускання рівна 0,0053 то згідно формули (8) знаходимо значення коефіцієнта $K_{\text{ФНЧ}} = 0,0053$. Знаходимо значення N при $\Delta f = 3,3$, $N_{\text{ФВЧ}} = 34$, візьмемо $N=43$, і коефіцієнти будуть рівні

$$h_D(n) = 2f_c \sin(\omega_c n)$$

де

$$h_D(n) = \frac{2f_c \sin(\omega_c n)}{\omega_c}, n \in \mathbb{Z}$$

$$= 2f_c, n=0$$

$$h_D(n) = 1-2 \sin^2 \frac{\omega_c n}{2}$$

Внаслідок ефекту змазування характеристики фільтра, що вводиться ваговою функцією, частота зрізу отриманого фільтра буде відрізнятися від заданої в специфікації, щоб врахувати цей ефект, використаємо f_c — центр смуги переходу:

$$h_D(n) = \frac{2f_c \sin(\omega_c n)}{\omega_c}$$

Обчислюємо значення $h_D(n)$ згідно виразу (7).

$$h_D(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} 1 \times e^{i\omega n} d\omega = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{i\omega n} d\omega =$$

$$= \frac{2f_c \sin(n\omega_c)}{n\omega_c}, n \neq 0, -\infty \leq n \leq \infty$$

$$= 2f_c, n=0$$

Оскільки $h_D(n)$ симетрична функції то варто обчислити лише її значення на проміжку $0 \leq n \leq 21$

n=0:

$$10 \frac{1}{2} 23 58$$

$$\alpha(0)=1$$

$$10 \frac{1}{2} 23 58$$

n=1:

$$10 \frac{1}{2} 23 58$$

$$10 \frac{1}{2} 23 58$$

$$\alpha(1)=1$$

$$10 \frac{1}{2} 23 58$$

n=2:

$$10 \frac{1}{2} 23 58$$

$$10 \frac{1}{2} 23 58$$

$$\alpha(2)=1$$

$$10 \frac{1}{2} 23 58$$

...

n=21:

$$10 \frac{1}{2} 23 58$$

$$10 \frac{1}{2} 23 58$$

$$\alpha(21)=1$$

$$10 \frac{1}{2} 23 58$$

Обчисливши всі коефіцієнти $0 \leq n \leq 21$, решта коефіцієнтів знаходимо із правила симетрії $H(-n) = -H(n)$.

Отримані коефіцієнти підставляємо у вираз

$$x(n) = \sum_{m=0}^{N-1} H(m) x(n-m),$$

$$x(n) = \sum_{m=0}^{12} H(m) x(n-m),$$

$$x(n) = \sum_{m=0}^{12} H(m) x(n-m),$$

$$x(n) = \sum_{m=0}^{12} H(m) x(n-m),$$

$$x(n) = \sum_{m=0}^{12} H(m) x(n-m),$$

$$x(n) = \sum_{m=0}^{12} H(m) x(n-m),$$

Де $x(-n)=0$.

Лабораторна робота №3

Тема. Моделювання роботи препроцесора для попередньої обробки мовних сигналів.

Мета роботи: Опрацювати та випробувати в середовищі MATLAB 6.0 програму, яка реалізує етапи попередньої обробки мовних зразків.

Теоретичні відомості

Сучасні системи для розпізнавання суцільної мови з великим словником ґрунтуються на принципах статистичного розпізнавання образів [3,4]. На рис.1 показана структура типової системи статистичного розпізнавання, що включає чотири складових.

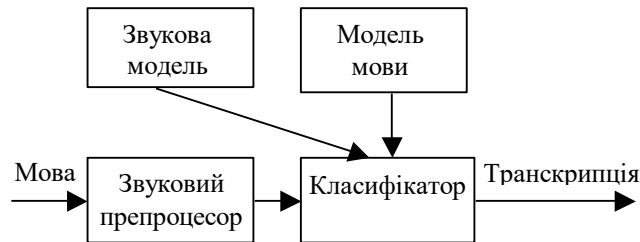


Рис.1 Структура системи розпізнавання мови

Складові виконують такі функції:

- звуковий препроцесор здійснює початковий етап обробки, на якому з мовного сигналу вибирається вся необхідна звукова інформація в компактному вигляді;
- звукова модель забезпечує обчислення правдоподібності будь-якої послідовності звукових векторів при заданій послідовності слів;
- модель мови забезпечує обчислення апіорної ймовірності послідовності слів незалежно від спостереження мовного сигналу. Для цього треба забезпечити механізм оцінки ймовірності певного слова у фразі, якщо знаємо попередні слова;
- класифікатор зводить воєдино дані від трьох раніше описаних компонент і знаходить найбільш імовірний текст (транскрипцію).

Принципи статистичного розпізнавання образів полягають у наступному [4].

На першому етапі мовний зразок перетворюється звуковим процесором на послідовність звукових векторів $Y = \vec{y}_1, \vec{y}_2, \dots, \vec{y}_T$. Кожен вектор є стислим поданням короточасного мовного спектру на інтервалі, як правило, близько 25 мс зі зсувом інтервалів на 10 мс. Типова фраза з десяти слів по 6-7 звуків у кожному може мати тривалість біля 3 с і представлятися послідовністю з $T=300$ звукових векторів.

У загальному, фраза складається з послідовності слів $W = w_1, w_2, \dots, w_n$. Робота системи розпізнавання полягає у визначенні найбільш імовірної послідовності слів \hat{W} , маючи звуковий сигнал Y . Для цього використовується правило Байєса [4]:

$$\hat{W} = \arg \max_W P(W / Y) = \arg \max_W \frac{P(W)P(Y / W)}{P(Y)}.$$

Ця рівність показує, що для знаходження найбільш правдоподібної послідовності слів W , повинна бути знайдена послідовність, що робить максимальним добуток $P(W)$ та $P(Y/W)$. Так як значення знаменника $P(Y)$ не залежить від W , то його при розпізнаванні ігнорують.

Перший із співмножників є апіорною ймовірністю спостереження W незалежно від спостереження мовного сигналу. Ця ймовірність визначається моделлю мови.

Другий співмножник є ймовірністю спостереження послідовності векторів Y при заданій послідовності слів W . Ця ймовірність визначається звуковою моделлю.

В звуковій моделі послідовності слів розбиваються на базові звуки – фонemi. Кожна індивідуальна фонема представляється прихованою моделлю за Марковим (англійська назва – hidden Markov model (НММ)). НММ-модель фонemi, як правило, має три породжуючі стани та вхідний і вихідний стан. Вхідний і вихідний стани дозволяють моделям фонем об'єднуватися, щоб утворювати слова та об'єднувати слова для утворення речень (послідовності слів).

Розглянемо більш детально функції препроцесора для попередньої обробки мовних сигналів.

Звуковий препроцесор

Потрібен початковий етап обробки, на якому з мовного сигналу вибирається вся необхідна звукова інформація в компактному вигляді.

Принципове припущення, яке робиться в сучасних розпізнавачах [3,4] є те, що мовний сигнал розглядається як стаціонарний (тобто спектральні характеристики відносно постійні) на інтервалі в кілька десятків мілісекунд. Тому основною функцією попередньої обробки є розбиття вхідного мовного сигналу на інтервали [1 - 4] і отримання для кожного інтервалу згладженої спектральної оцінки. Зсув між інтервалами звичайно рівний 10 мс. Інтервали, як правило, перекриваються і мають тривалість 25 мс. Як звичайно для обробки такого типу до кожного інтервалу на початку застосовується функція вікна (наприклад, вікно Гемінга). Часто застосовують високочастотне підсилення, щоб компенсувати послаблення, спричинене розсіюванням звуку від губ.

Щоб отримати спектральні оцінки використовується швидке перетворення Фур'є.

Фур'є спектр згладжується додаванням спектральних коефіцієнтів у межах “трикутних” частотних смуг розташованих на нелінійній (подібній до логарифмічної) Mel-шкалі [4]. Для граничної частоти мови (16 КГц) беруть 24 таких частотних смуги. Mel-шкала введена для наближення частотного розділення людського вуха, яке є лінійним до 1000 Гц та логарифмічним понад 1000 Гц.

З метою зробити статистику оціненого спектру потужності мови близькою до Гауссової до виходів набору фільтрів застосовується логарифмічний стиск.

До прологарифмованих коефіцієнтів застосовують дискретне косинусне перетворення. Це зосереджує спектральну інформацію в кепстральних коефіцієнтах з малими номерами, а також декорелює їх, дозволяючи при наступному статистичному моделюванні використовувати діагональні коваріаційні матриці. Перші 12 кепстральних коефіцієнтів та логарифм енергії інтервалу сигналу утворюють базовий 13-елементний звуковий вектор.

Є ряд додаткових перетворень, які можна застосувати для отримання остаточного звукового вектора.

Для зменшення мультиплікативного шуму на звукових векторах роблять нормалізацію кепстральних коефіцієнтів. Для кожної з дванадцяти компонент обчислюються середні значення по всіх звукових векторах даного мовного зразка. Ці середні значення віднімаються від відповідних компонент всіх звукових векторів даного мовного зразка.

Компоненти логарифму енергії даного мовного зразка, які менші від максимального значення на 50 дБ, замінюються на значення цього порогу. Потім всі значення логарифму енергії масштабуються так, щоб максимальне значення стало рівним 1,0.

Припускається, що кожен звуковий вектор не зв'язаний зі своїми сусідами. Це досить грубе припущення, бо фізичні обмеження голосового тракту людини передбачають плавні переходи між сусідніми спектральними оцінками. Проте додавання різниць та різниць

різниць базових елементів значно пом'якшує припущення. Переважно для цього беруться два попередні та два наступні вектори. У результаті отримуємо 39-елементний вектор [4]. Кілька інших можливих варіантів отримання звукових векторів описано в [3].

Попередня обробка мовних зразків (реалізація звукового препроцесора) ISOLET

База даних ISOLET складається з 7800 мовних зразків окремо вимовлених звуків англійської мови. Два зразки кожного звуку були записані від 150 осіб. Зразки дискретизовані з частотою 16 КГц та мають розрядність 16. Середня тривалість зразків - 0,5 с.

Етапи попередньої обробки мовних зразків у базі даних ISOLET детально описані нижче.

1. Оцифрований (дискретизований за часом та квантований за рівнем) мовний сигнал розбиваємо на блоки по 25.6 мс із зсувом кожні 10 мс, тобто, блоки по 409 відліків кожен блок, із зсувом на 160 відліків.

2. Пропускаємо блоки через фільтр першого порядку

$$S'_n = \begin{cases} 0, & \text{якщо } n = 1 \\ S_n - S_{n-1}, & \text{у протилежному випадку} \end{cases},$$

де S_n – n -й відлік у блоці.

3. Застосовуємо до блоку вікно Гемінга згідно з виразом

$$S''_n = 0.54 - 0.46 \cos\left(\frac{2\pi(n-1)}{408}\right) S'_n \quad \text{для } n = 1, \dots, 409$$

4. Збільшуємо довжину блоку до 512 елементів за рахунок доповнення його вкінці потрібною кількістю нулів. Після цього застосовуємо дискретне перетворення Фур'є й отримуємо 256 спектральних комплексних значень.

5. Нульовий член ігноруємо, а величини решта 255 спектральних значень усереднюємо. Усереднення реалізуємо як 24 трикутні смугопропускні фільтри. Нижня, середня та верхня частоти подані в таблиці. Вони апроксимують Mel-шкалу згідно з формулою

$$g = 2595 \log_{10}(1 + f / 700),$$

де g – частота в Mel, f – частота в Герцах.

Смуга	Нижня частота	Середня частота	Верхня частота
1	0,000 Гц	74,24 Гц	156,4 Гц
2	74,24 Гц	156,4 Гц	247,2 Гц
3	156,4 Гц	247,2 Гц	347,6 Гц
4	247,2 Гц	347,6 Гц	458,7 Гц
5	347,6 Гц	458,7 Гц	581,6 Гц
6	458,7 Гц	581,6 Гц	717,5 Гц
7	581,6 Гц	717,5 Гц	867,9 Гц
8	717,5 Гц	867,9 Гц	1034 Гц
9	867,9 Гц	1034 Гц	1218 Гц
10	1034 Гц	1218 Гц	1422 Гц
11	1218 Гц	1422 Гц	1647 Гц

12	1422 Гц	1647 Гц	1895 Гц
13	1647 Гц	1895 Гц	2171 Гц
14	1895 Гц	2171 Гц	2475 Гц
15	2171 Гц	2475 Гц	2812 Гц
16	2475 Гц	2812 Гц	3184 Гц
17	2812 Гц	3184 Гц	3596 Гц
18	3184 Гц	3596 Гц	4052 Гц
19	3596 Гц	4052 Гц	4556 Гц
20	4052 Гц	4556 Гц	5113 Гц
21	4556 Гц	5113 Гц	5730 Гц
22	5113 Гц	5730 Гц	6412 Гц
23	5730 Гц	6412 Гц	7166 Гц
24	6412 Гц	7166 Гц	8000 Гц

6. Логарифмуємо отримані 24 середні значення.

7. Обчислюємо перші 12 значень дискретного косинусного перетворення.

$$[O(u)]_i = \sum_{j=1}^{24} m_j \cos\left(\frac{\pi i(j-0,5)}{24}\right) \text{ для } i=1 \dots 12$$

де m_j - значення логарифму j -го середнього значення.

8. У кінець додаємо тринадцятий елемент, енергію поточного блока $\sum_{n=1}^{409} S_n^2$.

Далі можна обчислювати ще 26 елементів. Це "різниця" та "різниця-різниць" коефіцієнти. 13 "дельт" $[O(u)]_{14}, \dots, [O(u)]_{26}$ апроксимують степінь зміни базових коефіцієнтів косинусного перетворення та енергетичних коефіцієнтів. Їх обчислюємо так:

$$[O(u)]_{i+13} = \begin{cases} [O(u)]_{t+1} - [O(u)]_t \dots \dots \dots \text{для } t = 1, 2 \\ \frac{\sum_{r=1}^2 r([O(u)]_{t+r} - [O(u)]_{t-r})}{2 \sum_{r=1}^2 r^2} \dots \dots \dots \text{для } t = 3, \dots, T(u) - 2 \\ [O(u)]_t - [O(u)]_{t-1} \dots \dots \dots \text{для } t = T(u) - 1, T(u) \end{cases}$$

13 "дельта-дельта" $[O(u)]_{27}, \dots, [O(u)]_{39}$, які апроксимують прискорення базових коефіцієнтів косинусного перетворення Фур'є та енергетичних коефіцієнтів, отримуємо за наведеною вище формулою.

Кожний логарифмічний енергетичний профіль запису скануємо і низькочастотні значення заміняємо значенням на 50дБ нижче пікового значення. Потім енергетичний профіль масштабуємо так, щоб пікове значення було 1.0.

Зразок програми на мові сценаріїв MATLAB 6.0

Далі наведено зразок програми на мові сценаріїв MATLAB 6.0, яка реалізує описані вище етапи 1-5 попередньої обробки мовних зразків у базі даних ISOLET.

```
clear all;
vybirka = 1 ;
signal = wavread('example.wav') ;
subplot(3,3,1); plot(signal);title('example.wav');
```

```

d = length(signal) ;
i = 1 ;
while i < d - 408
    vybirka = vybirka + 1 ;
    y = signal(i : i + 408) ;
    S(1) = 0.0 ;
    for n = 2 : 409
        S(n) = y(n) - y(n - 1) ;
    end;
    for n = 1 : 409
        D(n) = (0.54 - 0.46 * cos(2 * pi * (n - 1) / 408)) * S(n) ;
    end;
    B = abs(fft(D ,512)) ;
    C = B(1:256);

    Hz = [0; 74.24; 156.4; 247.2; 347.6; 458.7; 531.6; 717.5; 867.9 ; 1034;
1213; 1422;1647; 1895; 2171;
    2475; 2812; 3184; 3596; 4052; 4556; 5113; 5730;6412; 7166; 8000];
    for n = 1 : 26
        V(n) = round (Hz(n) * 256 / 8000) ;
        V(1) = 1;
    end;
    Sum = 0 ;
    R(1 : 24) = 0;
    for j = 1 : 24
        N = V(j + 2) - V(j);
        N1 = V(j + 1) - V(j);
        D1 = 1 / N1;
        for k = 1 : N1
            Sum = Sum - S(k + V(j)) * D1 * k;
        end;
        N2 = V(j + 2) - V(j + 1);
        D2 = 1 / N2;
        for k = 1 : N2
            Sum = Sum - S(k + V(j + 1)) * (1 - D2 * k) ;
        end;
        R(j) = Sum / N ;
        Result (vybirka,j) = R(j);
    end;
    i = i + 160;
end;
subplot(3,3,2); plot(y);title('          409 values ');
subplot(3,3,3); plot(S);title('          applying filters');
subplot(3,3,4); plot(D);title('          Hemming');
subplot(3,3,5); plot(B);title('          512 fft ');
subplot(3,3,6); plot(C);title('          256 elements ');
subplot(3,3,7); plot(Result(vybirka,1:24));
title('          24-element vector');

```

Типовий мовний зразок, який обробляємо – звуковий файл example з розширенням wav (затягнутий звук ‘о’). Файли цього типу можна створювати, редагувати та прослуховувати за допомогою звукової плати, мікрофона й навушників у середовищі професійного звукового редактора Cool Edit. Вигляд вікна цього звукового редактора показано нижче (рис.2).



Рисунок 2 – Вигляд вікна цього звукового редактора Cool Edit

Графічне вікно середовища MATLAB 6.0, у якому ілюструються етапи попередньої обробки мовного зразка example.wav згідно з наведеною програмою, показано на рис.3.

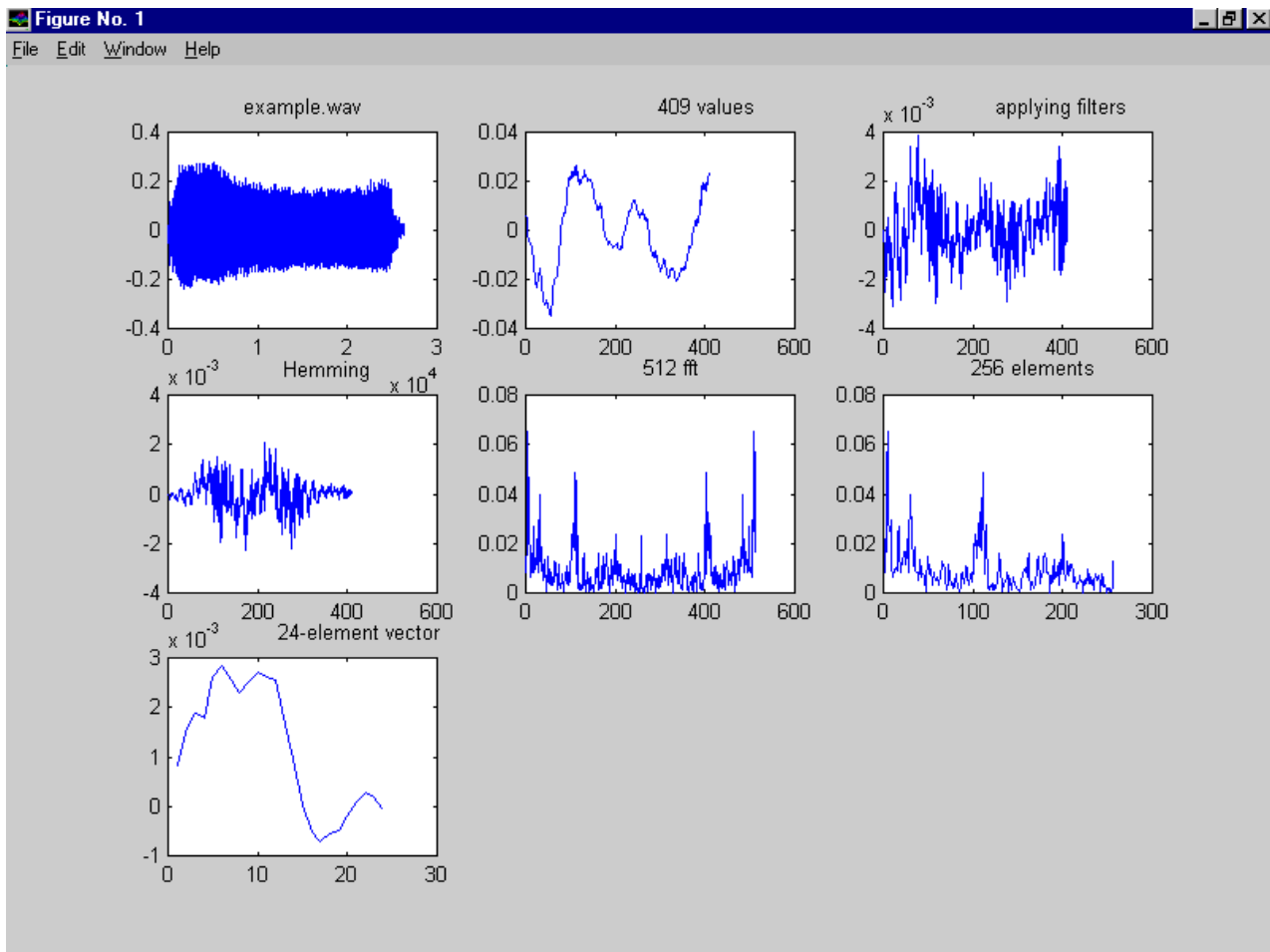


Рисунок 3 – Графічне вікно середовища MATLAB 6.0, у якому ілюструються етапи попередньої обробки мовного зразка example.wav згідно з наведеною програмою

Зміст звіту

1. Мета роботи.
2. Основні теоретичні відомості.
3. Блок-схема алгоритму попередньої обробки мовного сигналу.
4. Результат виконання індивідуального завдання
 - а) скоректована програма на мові на мові сценаріїв MATLAB 6.0, яка реалізує етапи попередньої обробки мовного сигналу;
 - б) спектральні вектори, які є результатом опрацювання заданого викладачем мовного сигналу.
5. Висновок.

Лабораторна робота №4

Тема. Стиск зображень з використанням дискретних косинусних перетворень.

Мета роботи: Опрацювати та випробувати в середовищі MATLAB 6.0 програму, яка реалізує етапи стиску зображень з використанням дискретного косинусного перетворення.

Теоретичне підґрунтя

Безвтратні методи стиску (кодування Хафмена, LZW, довжин серій) не забезпечують потрібного у багатьох випадках ступеня стиску зображень. Необхідно застосовувати методи стиску з втратою інформації. Один з таких підходів використовується у форматах стиску зображень JPEG.

Стиск даних у форматі JPEG (Joint Photographic Experts Group), який дозволяє стискати окремі (незмінні, still picture) зображення, можна умовно розбити на три етапи:

1-й етап - перетворення та субдискретизація кольорової інформації.

2-й етап – поблочні дискретні косинусні перетворення.

3-й етап – квантування та кодування значень дискретного косинусного перетворення.

Перший етап - це перетворення та субдискретизація кольорової інформації. Він полягає в наступному.

Кожна точка зображення, представлена 3 байтами в системі RGB, переводиться в систему YUV (яскравість, кольорова насиченість, кольоровий тон) згідно виразів:

$$\begin{aligned}Y &= 27/256 * R + 150/256 * G + 29/256 * B \\U &= 131/256 * R - 110/256 * G - 21/256 * B + 128 \\V &= -44/256 * R - 87/256 * G + 131/256 * B + 128\end{aligned}$$

або в матричному вигляді

$$\begin{pmatrix} Y \\ U - 128 \\ V - 128 \end{pmatrix} = \begin{pmatrix} 0,301 & 0,586 & 0,113 \\ 0,512 & -0,430 & -0,082 \\ -0,172 & -0,340 & 0,512 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Перетворення з системи YUV в систему RGB виконується за формулами:

$$\begin{aligned}R &= Y + 1,37 * (U - 128) \\G &= Y - 0,698 * (U - 128) - 0,336 * (V - 128) \\B &= Y + 1,73 * (V - 128)\end{aligned}$$

або в матричному вигляді

$$\begin{pmatrix} R + 175,4 \\ G - 132,4 \\ B + 221,4 \end{pmatrix} = \begin{pmatrix} 1 & 1,370 & 0 \\ 1 & -0,698 & -0,336 \\ 1 & 0 & 1,730 \end{pmatrix} \begin{pmatrix} Y \\ U \\ V \end{pmatrix}$$

Далі значення компоненти Y залишаються без зміни, а число значень компонент U і V зменшується (так звана субдискретизація; компоненти U і V можна загрубити без суттєвої втрати якості зображення). Можливі різні варіанти субдискретизації: просте викидання частини з сусідніх точок чи заміна значень сусідніх точок зображення на їх середні. При цьому можливі кілька варіантів об'єднання точок: дві по горизонталі, дві по вертикалі, квадрат з чотирьох сусідніх точок. Найчастіше використовується наступний варіант: число точок зменшується вдвоє, причому значення точок обчислюються згідно з виразом

$$y(n) = 1/4 * x(n-2) + 1/2 * x(n-1) + 1/4 * x(n).$$

При цьому блок 8X16 значень компоненти U або V перетворюється в блок 8X8 значень.

При відтворенні інформації для покращення якості проміжні точки рекомендується отримувати не простим повторенням, а шляхом інтерполяції між сусідніми точками. Найчастіше використовується наступний спосіб: при відтворенні зображення блок 8X8 точок реконструюється в блок 8X16 точок за формулою

$$x(n) = [y(2n) + y(2n-1)]/2.$$

Якщо використовується описаний варіант субдискретизації, то досягається стиск зображення в 1,5 раза. Дійсно 1 байт компоненти яскравості залишається без змін, а кожні 2 байти компонент U і V заміняються на 1 байт. Отже, замість 6 байт на кожних 2 точки зображення тепер припадає 4 байта. Якщо використовується варіант об'єднання чотирьох сусідніх точок, то досягається стиск зображення в 2 рази. Адже 1 байт компоненти яскравості залишається без змін, а кожні 4 байти компонент U і V заміняються на 1 байт. Тобто, замість 12 байт на кожних 4 точки зображення тепер припадає 6 байт.

Мінімальний фрагмент інформації (MCU) для обробки – це блок початкового RGB зображення розміру 8X16 елементів. У результаті обробки такого фрагменту на першому етапі отримуємо чотири блоки розміру 8X8: два блоки розміру 8X8 для компоненти яскравості Y та по одному блоку розміру 8X8 для компонент U і V. Це ілюструється наступним рисунком.

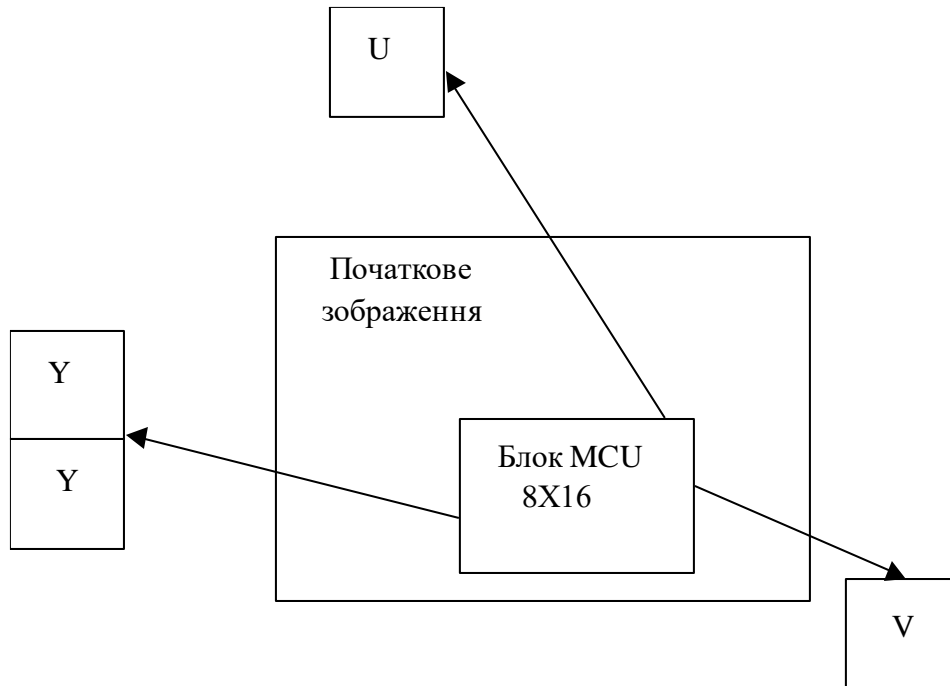


Рисунок 1 – Структура блоку початкового RGB зображення розміру 8X16 елементів.

Другий етап має в своїй основі дискретне косинусне перетворення (ДКП).

Кожна з компонент Y,U,V зображення на цьому етапі розглядається як окреме монохромне (однокольорове) зображення і її стиск проводиться окремо.

Зображення розбивається на блоки 8X8 елементів. До кожного блоку P застосовується двовимірне ДКП

$$P_{ДКП} = A P A^T,$$

де A - матриця двовимірного ДКП;

$P_{ДКП}$ - матриця значень ДКП фрагменту зображення.

Пряме ДКП задається наступним виразом

$$BD(u, v) = \frac{1}{4} c(u) c(v) \sum_{y=0}^7 \sum_{x=0}^7 b d(x, y) \cos \frac{\pi u (2x+1)}{16} \cos \frac{\pi v (2y+1)}{16}$$

а обернене ДКП задається формулою

$$b d(x, y) = \frac{1}{4} \sum_{y=0}^7 \sum_{x=0}^7 c(u) c(v) B D(u, v) \cos \frac{\pi u (2x+1)}{16} \cos \frac{\pi v (2y+1)}{16}$$

де $c(i) = \frac{1}{\sqrt{2}}$ для $i=0$ та $c(i)=1$ для $i=1, 2, \dots, 7$.

Двовимірне ДКП має ту властивість, що воно зосереджує найбільші значення у верхньому лівому куті матриці перетворення. Типовий розподіл ДКП коефіцієнтів показано в наступній матриці:

500	100	100	100	0	0	0	0
100	100	100	100	0	0	0	0
100	100	100	100	0	0	0	0
100	100	100	100	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	01	0	0	0	0	0
0	0	0	0	0	0	0	0

Ці значення треба взяти більш точно, а решта значень ДКП можна суттєво закругити. Це і є основою стиску зображень у форматі JPEG. Дії, які реалізують описану ідею, відбуваються на наступному третьому етапі.

На третьому етапі виконуються квантування та кодування значень дискретного косинусного перетворення.

Спочатку виконується квантування значень ДКП. Для цього формується матриця Q дільників з елементами $q(i,j)=1+(1+i+j)g, i,j=0,1,\dots,7$; g - параметр, який впливає на якість зображення, що отримуємо при відтворенні. Для компонент Y рекомендується брати $g=2$, а для компонент U,V значення g може бути більшим. $q(i,j)$ це не що інше як крок квантування, який залежно від позиції змінюється. При русі від верхнього лівого кута до правого нижнього кута крок квантування збільшується, тобто виконується грубіше квантування.

При $g=2$ матриця дільників має вигляд

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Значення $x(i,j)$ матриці $R_{дкп}$ діляться на відповідні значення $q(i,j)$ матриці дільників і заокруглюються до найближчого цілого. Процес квантування описується наступним виразом:

$$Q(x(i,j),i,j)=round(x(i,j)/q(i,j)).$$

Приклад, як виглядає матриця ДКП після квантування значень, наведено нижче:

30	0	0	0	0	0	0	0
-7	-8	1	1	0	0	0	0
-11	6	0	1	0	0	0	0
-5	-3	0	0	0	0	0	0
-7	-3	2	0	0	0	0	0
-4	4	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
-3	1	0	0	0	0	0	0

До коефіцієнтів ДКП, які знаходяться на першому місці (DC коефіцієнти), застосовується дельта імпульсно-кодова модуляція:

$$DC(-1)=0; del(0)=0; del(n)=DC(n)-DC(n-1)$$

До коефіцієнтів, які знаходяться на місцях крім першого (AC коефіцієнти), застосовується описане нижче кодування.

Останнім кроком є застосування до елементів отриманої матриці кодування Хафмена разом з кодуванням довжин серій для послідовностей нулів. Вважається, що поява нульового елемента у даній матриці є найбільш імовірною, а із збільшенням абсолютної величини елемента ймовірність його появи зменшується. Один з варіантів таблиці кодування виглядає наступним чином.

елемент	кодове слово
0	1
+1	0100
-1	0101
+2	0110
-2	0111
+3	00100
-3	00101
+4	00110
-4	00111
+5	0001000
-5	0001001
+6	0001010
-6	0001011
+7	0001100
-7	0001101
+8	0001110
-8	0001111
$ D >8$	00001+8розрядів значення D

Таблиця кодування може фіксуватися перед початком обробки або бути адаптивною, тобто мінятися з використанням статистики появи даних у попередніх блоках.

Кодування матриці квантованих елементів ДКП проводиться по шляху, показаному послідовними номерами (так званий зигзаг або змійка; у напрямку збільшення значень елементів матриці дільників).

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

Результатом кодування є послідовність кодових слів коду Хафмена. Відомо, що ніякі два слова в цьому коді не мають однакового початку, тобто слова в послідовності можуть іти підряд без маркерів розділення.

Певні маркери мають бути лише для того, щоб вказати, де починається послідовність кодових слів, отриманих при обробці блоку 8X8 елементів.

Замість застосування коду Хафмена стандарт допускає також використання безвтратних арифметичних кодів.

При відтворенні зображення виконуються наступні дії, зворотні до дій при стиску:

- декодування Хафмена;
- деквантування (множення на відповідні значення елементів матриці дільників);
- обернені двовимірні ДКП блоків 8X8 елементів;
- обернена кольорова субдискретизація (відтворення проміжних точок для компонент U,V шляхом інтерполяції між сусідніми точками);
- перехід від системи YUV до системи RGB.

Етапи стиску та відтворення даних при використанні формату JPEG показані на рисунку.

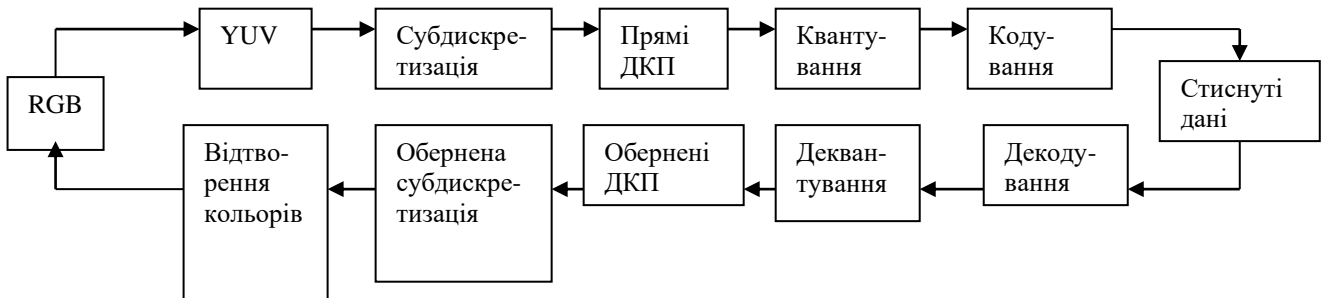


Рисунок 2 – Етапи стиску та відтворення даних при використанні формату JPEG

Степінь стиску в форматі JPEG - від 16:1 до 25:1 без помітної втрати якості.

Зразок програми на мові сценаріїв MATLAB 6.0

Далі наведено зразок програми на мові сценаріїв MATLAB 6.0, яка реалізує описані вище етапи 1-3 стиску зображень.

```

clear all;
imshow source.tif;
im_rgb=imread('source.tif');
info='Started JPEG compression'

% RGB to YUV
info='RGB to YUV...'
a_rgb=double(im_rgb(1:400,1:496,1:3));
a_y=round(77/256*a_rgb(1:400,1:496,1)+150/256*a_rgb(1:400,1:496,2)+29/256*a_rgb(1:400,1:496,3));
a_u=round(131/256*a_rgb(1:400,1:496,1)-110/256*a_rgb(1:400,1:496,2)-21/256*a_rgb(1:400,1:496,3)+128);
a_v=round(-44/256*a_rgb(1:400,1:496,1)-87/256*a_rgb(1:400,1:496,2)+131/256*a_rgb(1:400,1:496,3)+128);
%info='finished'

% downsampling
info='downsampling U,V...'
a_u_ds(1:400,1)=a_u(1:400,1);
a_v_ds(1:400,1)=a_v(1:400,1);

for j=1:400
    for i=2:248
        a_u_ds(j,i)=round(a_u(j,2*i-2)/4+a_u(j,2*i-1)/2+a_u(j,2*i)/4);
        a_v_ds(j,i)=round(a_v(j,2*i-2)/4+a_v(j,2*i-1)/2+a_v(j,2*i)/4);
    end;
end;
clear a_u a_v
%info='finished'

% dct
  
```

```

info='2-D Discrete Cosine Transform ...'
y_dct=blkproc(a_y,[8,8],'dct2');
u_dct=blkproc(a_u_ds,[8,8],'dct2');
v_dct=blkproc(a_v_ds,[8,8],'dct2');
clear a_u_ds a_v_ds
%info='finished'

%quantization
info='quantization ...'
r=3;
for i=1:8
    for j=1:8
        kvant(i,j)=1+(i+j-1)*r;
    end;
end;

for j=1:50
    for i=1:62
        y_kv(8*j-7:8*j,8*i-7:8*i)=round(y_dct(8*j-7:8*j,8*i-7:8*i)./kvant);
    end;
end;
for j=1:50
    for i=1:31
        u_kv(8*j-7:8*j,8*i-7:8*i)=round(u_dct(8*j-7:8*j,8*i-7:8*i)./kvant);
        v_kv(8*j-7:8*j,8*i-7:8*i)=round(v_dct(8*j-7:8*j,8*i-7:8*i)./kvant);
    end;
end;
clear fuctions;
clear y_dct u_dct v_dct;
%info='finished'

%Differential pulse-code modulation (DPCM) coding and runlength coding
info='Zigzag Ordering Y ...'
for j=1:50
    for i=1:62
        y_mas((j-1)*62+i,1:64)=zigzag(y_kv(8*j-7:8*j,8*i-7:8*i));
    end;
end;
info='Zigzag Ordering U,V ...'
for j=1:50
    for i=1:31
        u_mas((j-1)*31+i,1:64)=zigzag(u_kv(8*j-7:8*j,8*i-7:8*i));
        v_mas((j-1)*31+i,1:64)=zigzag(v_kv(8*j-7:8*j,8*i-7:8*i));
    end;
end;
clear fuctions;
info='Coding Y (DPCM & RunLength) ...'
x_y=0;midle=[];
for i=1:3100
    y_cod=[midle y_mas(i,1)-x_y runcode(y_mas(i,2:64))];
    x_y=y_mas(i,1);
    midle=y_cod;
end;
clear functions;
info='Coding U,V (DPCM & RunLength)'
x_u=0;x_v=0;midle_u=[];midle_v=[];
for i=1:1550
    u_cod=[midle_u u_mas(i,1)-x_u runcode(u_mas(i,2:64))];
    x_u=u_mas(i,1);
    midle_u=u_cod;
    v_cod=[midle_v v_mas(i,1)-x_v runcode(v_mas(i,2:64))];
    x_v=v_mas(i,1);
    midle_v=v_cod;
end;
clear y_kv u_kv v_kv;
clear functions;

```



```

info='Huffman coding Y ...'
m_y=0;
for i=1:length(y_cod)
    m_y=m_y+huffman(y_cod(i));
end;
clear functions;
info='Huffman coding U,V ...'
m_u=0;
for i=1:length(u_cod)
    m_u=m_u+huffman(u_cod(i));
end;
clear functions;
m_v=0;
for i=1:length(v_cod)
    m_v=m_v+huffman(v_cod(i));
end;
clear functions;

%info='finished coding'
info='Finished JPEG Compression'

m=size(im_rgb);
compression_ratio=8*m(1)*m(2)*m(3)/(m_y+m_u+m_v);
%clear m m_y m_u m_v
compression_ratio
imshow source.tif;
%clear all;
function [res]=zigzag(x)
res=[x(1,1) x(1,2) x(2,1) x(3,1) x(2,2) x(1,3) x(1,4) x(2,3) x(3,2) x(4,1)
x(5,1) x(4,2) x(3,3) x(2,4) x(1,5) x(1,6) x(2,5) x(3,4) x(4,3) x(5,2) x(6,1)
x(7,1) x(6,2) x(5,3) x(4,4) x(3,5) x(2,6) x(1,7) x(1,8) x(2,7) x(3,6) x(4,5)
x(5,4) x(6,3) x(7,2) x(8,1) x(8,2) x(7,3) x(6,4) x(5,5) x(4,6) x(3,7) x(2,8)
x(3,8) x(4,7) x(5,6) x(6,5) x(7,4) x(8,3) x(8,4) x(7,5) x(6,6) x(5,7) x(4,8)
x(5,8) x(6,7) x(7,6) x(8,5) x(8,6) x(7,7) x(6,8) x(7,8) x(8,7) x(8,8)];

function [res]=runcode(vect)
n=0;res=[];
for i=1:63
    if vect(i)==0
        n=n+1;
    else
        res=[res n vect(i)];
        n=0;
    end;
end;
res=[res 0 0];

function kod=huffman(elem)
switch elem
case 0, kod=1; % kode word=1
case 1, kod=4; % kode word=0100
case -1, kod=4; % kode word=0101
case 2, kod=4; % kode word=0110
case -2, kod=4; % kode word=0111
case 3, kod=5; % kode word=00100
case -3, kod=5; % kode word=00101
case 4, kod=5; % kode word=00110
case -4, kod=5; % kode word=00111
case 5, kod=7; % kode word=0001000
case -5, kod=7; % kode word=0001001
case 6, kod=7; % kode word=0001010
case -6, kod=7; % kode word=0001011
case 7, kod=7; % kode word=0001100
case -7, kod=7; % kode word=0001101
case 8, kod=7; % kode word=0001110

```

```
case -8, kod=7; % kode word=00011111
otherwise kod=13; % kode word=00001+elem(8 bits)

end;
```

Типове зображення, яке обробляємо – файл зображення з розширенням tiff.

Зміст звіту

1. Мета роботи.
2. Основні теоретичні відомості.
3. Блок-схема алгоритму стиску.
4. Результат виконання індивідуального завдання
 - а) скоректована програма на мові сценаріїв MATLAB 6.0, яка реалізує етапи стиску зображень з використанням дискретного косинусного перетворення;
 - б) потік бітів, які є результатом опрацювання заданого викладачем зображення;
 - в) оцінка ступеня стиску зображення.
5. Висновок.

Лабораторна робота №5

Тема. Фільтрація сигналів і зображень.

Мета роботи: Ознайомитися з методами та засобами фільтрації сигналів та зображень. Проілюструвати процес фільтрації зображення в просторовій області..

Загальні теоретичні відомості

Цифрова фільтрація даних (сигналів) є одною з основних і найпоширеніших задач цифрової обробки інформації. Під фільтрацією будемо розуміти будь-яке перетворення інформації, в нашому випадку - сигналів, при якому у вхідній послідовності оброблюваних даних цілеспрямовано змінюються певні співвідношення (динамічні або частотні) між різними компонентами цих даних. До основних операцій фільтрації інформації відносять: згладжування; прогнозування; диференціювання; інтегрування; поділ на певні складові; виділення інформаційних (корисних) сигналів; придушення шумів (завад).

Як відомо, перетворення динаміки сигналів (і даних, які несуть ці сигнали) здійснюється в системах. Системи, що вибірково міняють форму сигналів (амплітудно-частотну або фазочастотну характеристику) називають фільтрами. Відповідно, фільтри з будь-яким цільовим призначенням є частковим випадком систем перетворення сигналів, а у вузкому розгляді – лінійних дискретних систем.

У загальному випадку терміном *цифровий фільтр* (ЦФ) називають апаратну або програмну реалізацію математичного алгоритму, входом якого є цифровий сигнал, а виходом – інший цифровий сигнал з певним чином модифікованою формою і/або амплітудною і фазовою характеристикою. Класифікація цифрових фільтрів звичайно базується на функціональних ознаках алгоритмів цифрової фільтрації, відповідно до якого ЦФ підрозділяються на 4 групи:

- фільтри частотної селекції;
- оптимальні (квазіоптимальні);
- адаптивні;
- евристичні.

Найбільш вивченими і випробуваними на практиці є ЦФ частотної селекції. Оскільки фільтрація фактично є операцією згортки в часовій області, то в частотній області їй відповідає процедура множення спектру вхідного сигналу на частотну характеристику фільтру. При цьому довжина ядра згортки – це розмір фільтра. Зміна спектру в задачах звукозапису дозволяє очищати звук від шумів, міняти тембри інструментів, компенсувати перевертання сигналу, породжені, наприклад, мікрофоном. При обробці зображень, фільтрація дозволяє виділяти границі об'єктів, розмивати зображення, добиватися ефекту тиснення, вирівнювати освітлення на фотознімках і т.д.

Відомі методи цифрової обробки даних, які є методами цифрової фільтрації такі як метод згладжування відліків у ковзаючому вікні постійної довжини. Наприклад для лінійного згладжування даних за п'ятьма точками з однаковими ваговими коефіцієнтами використовується формула:

$$y_k = 0.2(x_{k-2} + x_{k-1} + x_k + x_{k+1} + x_{k+2}).$$

З точки зору цифрової фільтрації це двосторонній симетричний нерекурсивний фільтр:

$$y_k = \sum_{n=-2}^2 b_n x_{k-n}, \quad b_n = 0.2.$$

Серед статистичних методів та алгоритмів фільтрації сигналів і зображень найчастіше застосовується екстремальна та медіанна фільтрація. Зокрема, медіанну фільтрацію доцільно використовувати коли сигнал чи зображення є адитивною сумішшю корисного сигналу та імпульсних завад (тобто, на зображенні спостерігаються білі чи чорні цятки, або “сніг”).

Фільтрація зображень в часовій (просторовій) області

В часовій і просторовій області процес фільтрації сигналів описується рівнянням одновимірної згортки:

$$y(m) = \sum_{n=0}^{N-1} x(n) \cdot h(m-n),$$

де: $y(m)$ - вихідний сигнал;

$x(n)$ - вхідний сигнал;

$h(m-n)$ - імпульсна характеристика фільтру.

Фільтрація зображень в часовій області зводиться до двовимірної лінійної згортки:

$$g(p, q) = \sum_{i=-M}^M \sum_{j=-M}^M h(i, j) x(p+i, q+j),$$

де: $x(p, q)$ - вхідне зображення;

$g(p, q)$ - фільтроване зображення;

$h(i, j)$ - імпульсна характеристика фільтру (маска, що визначає вид фільтрації);

$P \times Q$ - розмір зображення, $p = 0, 1, \dots, P-1$, $q = 0, 1, \dots, Q-1$;

$M \times M$ - розмір вікна фільтрації (апертури);

$i = -M, -M+1, \dots, M$, $j = -M, -M+1, \dots, M$.

Очевидно, що процес фільтрації – це послідовне обчислення згортки обраної маски з «вікном» (частиною) зображення. Елементи вікна, розташовані в області точки для якої обчислюється згортка. Таким чином, маска фільтру, її ще називають *апертурою*, «пробігає» всі елементи зображення, утворюючи вихідне зображення.

Як правило, основну інформацію в зображеннях несуть контури об'єктів. При фільтрації зашумлених зображень ступінь згладжування контурів об'єктів залежить від розмірів апертури фільтру. При малих розмірах апертури краще зберігаються контрастні деталі зображення, але гірше пригашаються шуми. При великих розмірах апертури - навпаки. Таке протиріччя долається шляхом застосування фільтрів з адаптивним розміром апертур. В адаптивних фільтрах великі апертури застосовуються для монотонних ділянок зображення, а малі – поблизу неоднорідних деталей.

Маски для низькочастотної та високочастотної фільтрації визначають тип фільтрації. Низькочастотна фільтрація забезпечує згладжування шуму, тобто усунення високочастотних складових. Вона досягається за рахунок використання масок з додатними елементами. Прикладом таких масок можуть бути наступні масиви, що мають розмір 3×3 точки. Зауважимо, що для того, щоб процедура пригашення шуму не приводила до зміщення середньої яскравості зображення ці масиви є нормованими.

$$h = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}; h = \frac{1}{10} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}; \quad h = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Для високочастотної фільтрації можна навести такі маски:

$$h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}; \quad h = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}; \quad h = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Особливістю таких апертур є те, що алгебраїчно сума елементів кожної з них дорівнює одиниці. Використання високочастотних масок приводить до виділення границь об'єктів,

тому може бути використана, наприклад, маска Роберта, різницевий оператор Собеля, Кірша.

Широке поширення набули методи контрастування (один з випадків високочастотної - фільтрації), в яких використовується оператор Лапласа. На практиці він замінюється згорткою зображення з однією з масок:

$$h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}; \quad h = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}; \quad h = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}.$$

Алгоритми лінійної фільтрації

Виконувати лінійну фільтрацію двовимірних масивів (зображень) можна звикористанням алгоритмів, що приводяться нижче. Перший з них реалізовує процес безпосередньо за формулою згортки, а другий – зменшує необхідний об'єм пам'яті, що може суттєво впливати на швидкість обробки при великих розмірах вхідного зображення.

В обох із запропонованих алгоритмів, границі зображень обробляються без фільтрації, тобто у вихідній матриці елементи крайніх рядків та стовпців співпадають з вхідними. Такий вид обробки найбільш прийнятний для тестових прикладів, хоча в реальних системах використовуються методи копіювання сусідів, або інші, складніші алгоритми доповнення.

Алгоритм 1. Прямий.

1. *Перший і останній рядки (верхня і нижня границі зображення)*

```
for (p = 0; p < M; p++)
    for (q = 0; q < Q; q++)
        { g[p, q]=x[p, q];  g[P-1-p, q]=x[P-1-p, q]};
```

2. *Решта рядків*

```
for (p = M; p < P - M; p++)
    {
        2.1. Перші і останні M елементів рядка (ліва і права границі зображення)
        for (q=0; q < M; q++)
            { g[p, q]=x[p, q];  g[p, Q-1-q]=x[p, Q-1-q] };
        2.2. Решта елементів (основне перетворення)
        for (q=M; q < Q - M; q++)
            { Sum=0.0;
              for (i = - M; i <= M; i++)
                  for (j = -M; j <= M; j++)
                      Sum = Sum +h [i,j] * x[i+p,j+q];
              g[p,q] = Sum;
            }
    }
```

Алгоритм 2. Прямий, з мінімізацією необхідної пам'яті.

1. *Ініціалізація додаткової матриці xtemp[p, q] , розміру (M+1) x Q*

```
for (p = 0; p <= M; p++)
    for (q = 0; q < Q; q++)
        xtemp[p, q]=x[p, q];
```

2. Фільтрація

```
for (p = M; p < P - M; p++)
    {for (q=M; q < Q - M; q++ )
        { Sum=0.0;
          for (j = -M; j <= M; j++ )
              {for (i = 1; i <= M; i++ )
                  Sum = Sum +h [i,j] * x[i+p,j+q];
                for (i = -M; i < 1; i++ )
                    Sum = Sum +h[i,j] * xtemp[i+M,j+q];
              }
            g[p,q] = Sum;
          }
    }
```

Модифікація матриці *xtemp*

```
for (i = 1; q < M; q++)
    for (q = 0; q < Q; q++)
        xtemp[i-1, q]= xtemp[i, q];
for (q = 0; q < Q; q++)
    xtemp[M, q]= x[p+1, q];
}
```

Порядок виконання роботи

1. Сформувати вхідне зображення (власна фотографія у форматі **.raw**).
2. Розробити власний, або обрати один з запропонованих алгоритмів фільтрації і написати програму на довільній мові програмування високого рівня, що його реалізовує.
3. Задіяти заданий варіантом фільтр для вхідного зображення.
4. Порівняти вхідне та відфільтроване зображення та зробити висновок про властивості і можливості застосування заданого фільтру.

Зміст звіту до лабораторної роботи

1. Титульний аркуш.
2. Завдання на лабораторну роботу.
3. Теоретичний матеріал стосовно поставленого завдання.
4. Блок - схема виконання алгоритму фільтрації.
5. Лістинг програми на довільній мові програмування.
6. Результати роботи – вхідне і вихідне зображення.
7. Висновки.

Контрольні запитання

1. Що таке фільтрація сигналів?
2. Що таке апертура?
3. Для чого виконується фільтрація сигналів, а зокрема, зображень?

Варіанти завдань до лабораторної роботи

№	Імпульсна характеристика фільтру									
1	$h=1/9*[1\ 1\ 1;1\ 1\ 1;1\ 1\ 1]$									
2	$h=1/10*[1\ 1\ 1;1\ 2\ 1;1\ 1\ 1]$									
3	$h=1/16*[1\ 2\ 1;2\ 4\ 2;1\ 2\ 1]$									
4	$h=[0\ -1\ 0;-1\ -5\ -1;0\ -1\ 0]$									
5	$h=[-1\ -1\ -1;-1\ 9\ -1;-1\ -1\ -1]$									
6	$h=[1\ -2\ 1;-2\ 5\ -2;1\ -2\ 1]$									
7	$h=[1\ 1\ 1;1\ -2\ 1;-1\ -1\ -1]$									
8	<p><i>Оператор Кірса:</i> $i=0:1:7$; $g(n,m) = \max(1, \max(5S(i) - 3T(i))),$ де: $S(i)=A(i)+A(i+1)+A(i+2)$; $T(i)=A(i+3)+A(i+4)+A(i+5)+A(i+6)+A(i+7)$; а елементи $A(i)$ є сусідами поточного пікселя зображення і визначаються так:</p> <table><tr><td>A_0</td><td>A_1</td><td>A_2</td></tr><tr><td>A_7</td><td>$x(n,m)$</td><td>A_3</td></tr><tr><td>A_6</td><td>A_5</td><td>A_4</td></tr></table>	A_0	A_1	A_2	A_7	$x(n,m)$	A_3	A_6	A_5	A_4
A_0	A_1	A_2								
A_7	$x(n,m)$	A_3								
A_6	A_5	A_4								
9	$h=[-1\ 1\ 1;-1\ -2\ 1;-1\ 1\ 1]$									
10	$h=[-1\ -1\ 1;-1\ -2\ 1;1\ 1\ 1]$									
11	$h=[-1\ -1\ -1;1\ -2\ 1;1\ 1\ 1]$									
12	$h=[1\ -1\ -1;-1\ -2\ 1;1\ 1\ 1]$									
13	<p><i>Оператор Юліца:</i></p> $g(n,m) = \frac{1}{4} \log \left(\frac{ x(n,m) ^4}{A_1 * A_3 * A_5 * A_7} \right)$ <p>а елементи $A(i)$ є сусідами поточного пікселя зображення і визначаються так:</p> <table><tr><td>A_0</td><td>A_1</td><td>A_2</td></tr><tr><td>A_7</td><td>$x(n,m)$</td><td>A_3</td></tr><tr><td>A_6</td><td>A_5</td><td>A_4</td></tr></table>	A_0	A_1	A_2	A_7	$x(n,m)$	A_3	A_6	A_5	A_4
A_0	A_1	A_2								
A_7	$x(n,m)$	A_3								
A_6	A_5	A_4								
14	$h=[1\ 1\ 1;1\ -2\ -1;1\ -1\ -1]$									
15	$h=[0\ -1\ 0;-1\ -4\ -1;0\ -1\ 0]$									
16	$h=[-1\ -1\ -1;-1\ 8\ -1;-1\ -1\ -1]$									
17	$h=[1\ -2\ 1;-2\ 4\ -2;1\ -2\ 1]$									
18	$h=[-1\ 2\ -1;-1\ 2\ -1;-1\ 2\ -1]$									
19	$h=[-1\ -1\ -1;2\ 2\ 2;-1\ -1\ -1]$									
20	$h=[-1\ -1\ 2;-1\ 2\ -1;2\ -1\ -1]$									
21	<p><i>Оператор Собеля.</i> $g(n,m) = \sqrt{X^2 + Y^2}$; де: $X=(A_2+2*A_3+A_4)-(A_0+2*A_7+A_6)$; $Y=(A_0+2*A_1+A_2)-(A_6+2*A_5+A_4)$; а елементи $A(i)$ є сусідами поточного пікселя зображення і визначаються так:</p> <table><tr><td>A_0</td><td>A_1</td><td>A_2</td></tr><tr><td>A_7</td><td>$x(n,m)$</td><td>A_3</td></tr><tr><td>A_6</td><td>A_5</td><td>A_4</td></tr></table>	A_0	A_1	A_2	A_7	$x(n,m)$	A_3	A_6	A_5	A_4
A_0	A_1	A_2								
A_7	$x(n,m)$	A_3								
A_6	A_5	A_4								

22	<i>Оператор Робертса.</i> $g(n, m) = \sqrt{(x(n, m) - x(n + 1, m + 1))^2 + (x(n, m + 1) - x(n + 1, m))^2}$
23	$g(n, m) = x(n, m) - x(n + 1, m + 1) + x(n, m + 1) - x(n + 1, m) $
24	$h = [2 \ -1 \ -1; -1 \ 2 \ -1; -1 \ -1 \ 2]$
25	$h = [1 \ 1 \ 1; -1 \ -2 \ 1; -1 \ -1 \ 1]$
26	$h = [1 \ 1 \ -1; 1 \ -2 \ -1; 1 \ 1 \ -1]$
27	$h = [-1 \ -1 \ -1; 0 \ 4 \ 0; -1 \ -1 \ -1]$
28	$h = [-1 \ -1 \ -1; 2 \ 4 \ 2; -1 \ -1 \ -1]$
29	$h = [0 \ 0 \ 0; 2 \ 2 \ 2; 0 \ 0 \ 0]$
30	$h = [-1 \ -1 \ -1; 1 \ 16 \ 1; -1 \ -1 \ -1]$

Приклад виконання

Завдання:

Імпульсна характеристика фільтру задана: $h = [0 \ -1 \ 0; -1 \ -4 \ -1; 0 \ -1 \ 0]$.

Виконати фільтрацію вхідного зображення та зробити висновок про властивості фільтра із заданою імпульсною характеристикою.

Хід виконання.

Для розв'язання поставленого завдання, обираємо перший алгоритм лінійної фільтрації, оскільки він є найбільш простим для реалізації і створюємо програму в середовищі MatLab. При цьому вхідне зображення подається у стандартному двійковому форматі (.raw), який опрацьовується стандартними засобами обраного пакету. В даному випадку тестовим є типове в практиці цифрової обробки сигналів, чорно-біле зображення "Lenna", розмір якого 256 на 256 пікселів.

Фільтрація виконується за граф-схемою, яка наведена на рис. 1. Повний текст програми, що реалізовує дану граф-схему обробки наведений в Додатку.

Для того, щоб утворити вхідну матрицю, розроблено власну підпрограму <readim.m>, що дозволяє зчитати зображення у форматі .raw, переконатися у коректності відкриття/існування файлу, та присвоїти відповідні значення елементам матриці.

Результатом роботи створеного програмного засобу є матриця, що містить елементи фільтрованого зображення. За допомогою розробленої підпрограми графічного виводу <autoimage.m>, ця матриця відображається як чорно-біле зображення, розміром 256 на 256 пікселів.

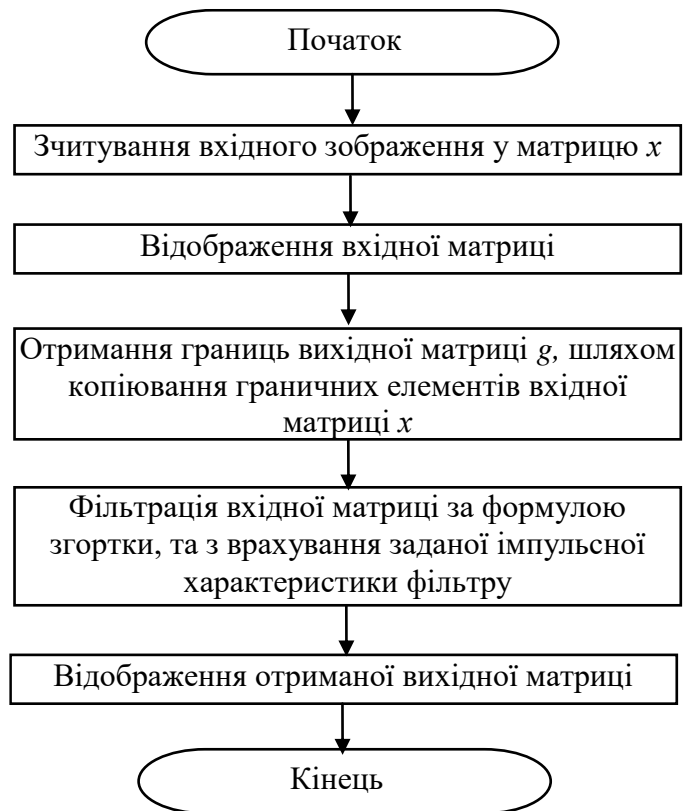


Рисунок 1 – Граф-схема алгоритму лінійної фільтрації зображення.

Результат роботи програми



Рисунок 2 – Вхідне тестове зображення



Рисунок 3 – Зображення, отримане в результаті застосування фільтру

Висновок: На рис. 2 та 3 наведено результат роботи програми фільтрації. Очевидно, що вихідне зображення є інвертованим щодо вхідного, за винятком границь. Тобто фільтр із заданою імпульсною характеристикою є фільтром високочастотної селекції і може бути використаний у відповідних задачах обробки сигналів.

Додаток

Текст програми

```
<filtr.m>
clc
clear all
close all
echo off

x = readim('Lenna.raw',[256,256]);
subplot(111); title('in');
autoimage(x);
P=256;
Q=256;
h=[0 -1 0;-1 -4 -1;0 -1 0]; % задання імпульсної характеристики фільтру
M=length(h);
%1. Перший і останній рядки (верхня і нижня границі зображення)
for p=1:1:M
    for q=1:1:Q
        g(p, q)=x(p, q);
        g(P-p, q)=x(P-p, q);
    end;
end;
%2. Решта рядків
for p=M:1: P - M;
%2.1. Перші і останні M елементів рядка (ліва і права границі зображення)
    for q=1:1:M
        g(p, q)=x(p, q);
        g(p, Q-q)=x(p,Q-q);
    end;
% 2.2. Решта елементів (основне перетворення)
    for q=M:1: Q - M
        SUM=0.0;
        for ii =1:1: M;
```

```

    for jj = 1:1: M;
        SUM = SUM +h ( ii,jj) * x( (ii-round(M/2))+p,(jj-round(M/2))+q);
    end;
end;
g(p,q) = SUM;
end
end
figure(2) subplot(111); title('out');
autoimage(g);

```

<readim.m>

```

function Image = readim(filename,par)
    fid = fopen(filename,'r');
    if fid < 0,
        disp('Error reading. ');
    else
        Image = fread(fid,par);
        fclose(fid);
    end
end

```

<autoimage.m>

```

function autoimage(img)
    mmin = min(min(img));
    mmax = max(max(img));
    image(256*(img-mmin)/(mmax-mmin))
    axis('image')
    colormap(gray(256))

```

Лабораторна робота №6

Тема. Модуляція та демодуляція сигналів. Амплітудна модуляція складених сигналів.

Мета роботи: Розглянути принципи модуляції сигналів. Проаналізувати особливості різних типів модуляції. Ознайомитись з алгоритмом отримання амплітудної модуляції звукових сигналів під задані вимоги до каналу передачі сигналів.

Теоретичні відомості

Модуляція (лат. *modulatio* - мірність, розмірність) — процес зміни одного або декількох параметрів високочастотного модульованого коливання за законом інформаційного низькочастотного повідомлення (сигналу)[1]. У результаті спектр керуючого сигналу переноситься в область високих частот, оскільки для ефективної трансляції в ефір необхідно щоб усі приймально-передавальні станції працювали на різних частотах і «не заважали» один одному. Це процес «посадки» інформаційного коливання на апіорно відому несучу. Передана інформація закладена в керуючому сигналі (модулюючий). Роль носія інформації виконує високочастотне коливання, назване несучим. Як несучі можуть бути використані коливання різної форми (прямокутні, трикутні і т.д.), однак найчастіше застосовуються гармонійні коливання.

• Класифікація видів модуляції

- За видом інформаційного сигналу:
 - аналогова (неперервна) модуляція (аналоговий сигнал);
 - дискретна модуляція (дискретний сигнал);
 - маніпуляція (0-1)
- За видом переносника (або несучої частоти):
 - гармонічна (синусоїдальний сигнал);
 - імпульсна (прямокутний періодичний імпульс);
 - цифрові;
- За видом параметрів несучої частоти, які зазнають зміни під дією інформаційного сигналу:
 - амплітудна модуляція;
 - частотна модуляція;
 - фазова модуляція;
 - широтна модуляція;
 - широтно-імпульсна модуляція.

При *гармонічній модуляції* несучими є гармонійне коливання. Розрізняють *амплітудну* (АМ), *частотну* (ЧМ) і *фазову* (ФМ) модуляції.

При *імпульсній модуляції* як несучу використовують послідовність імпульсів. При *неперервній модуляції* параметр носія який модулюється під впливом повідомлення, що передається, може приймати довільне значення в деякому неперервному інтервалі своїх значень, а при *дискретній* – скінчене число значень із деякого інтервалу своїх значень.

До складних видів імпульсної модуляції відноситься *дельта-модуляція* (ДМ) і *імпульсно-кодова модуляція* (ІКМ) [4]. Окремим випадком модуляції являється *маніпуляція сигналів*, при якому в якості модулюючого (інформаційного) сигналу використовується

послідовність одно- або двополярних прямокутних імпульсів. Використовують її в системах передачі дискретної інформації.

Модуляцію (неперервну чи дискретну) носія здійснюють за допомогою перетворювача, що називається *модулятором*. Процес відновлення впливу, здійсненого при модуляції на носій за значеннями одного або декількох його параметрів, називається *демодуляцією*, а перетворювач, за допомогою якого здійснюється процес демодуляції, - *демодулятором*. Пристрій, який одночасно виконує обидві операції, називається модемом (*модулятор-демодулятор*).

Розрізняють амплітудну модуляцію, при якій в залежності від миттєвої величини інформаційного сигналу змінюється амплітуда вихідного сигналу, і частотну модуляцію, при частота вихідного сигналу дещо змінюється в порівнянні з частотою сигналу-носія, пропорційно миттєвій величині інформаційного сигналу.

На рис.1 наведені графіки модуляції сигналів.

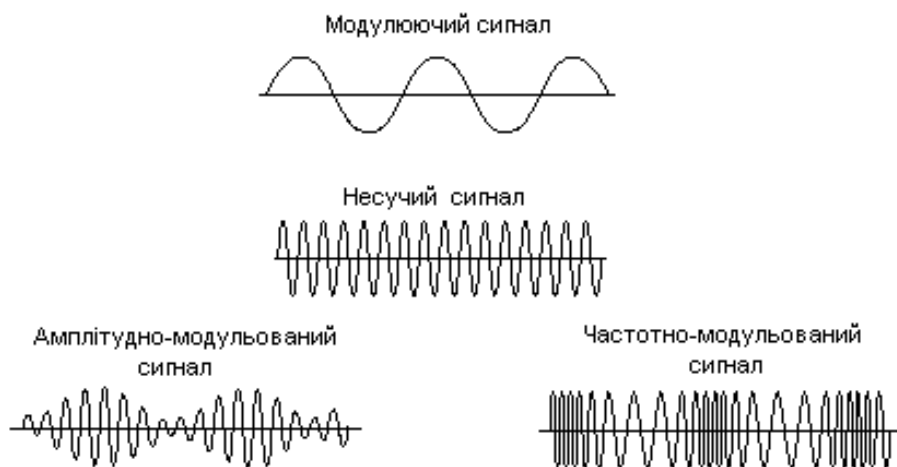


Рисунок 1 – Графіки амплітудної та частотної модуляції сигналів

Модуляція застосовується для узгодження інформаційного сигналу з параметрами лінії зв'язку.

Повідомлення низької частоти можна передавати і безпосередньо (без використання переносника високої частоти, тобто без модуляції). Проте модуляція розширює можливості передачі повідомлень з таких причин:

- збільшується кількість повідомлень, які можуть передаватись по одній лінії зв'язку шляхом використання частотного розділення сигналів і під несучих частот;
- підвищується достовірність сигналів, що передаються, при використанні завадостійких типів модуляції;
- підвищується ефективність випромінювання сигналу при передачі по радіоканалу.

Передачу інформації по лінії (каналу) зв'язку здійснюють за допомогою модуляції певних параметрів носія. Як носій можуть застосовуватись гармонійні сигнали і послідовність імпульсів. Вид і назву модуляції визначають в залежності від параметра що модулюється та форми носія.

Амплітудна модуляція

Амплітудною модуляцією (АМ) називають утворення сигналу шляхом зміни амплітуди гармонійного коливання (несучої) пропорційно миттєвим значенням напруги чи струму іншого, більш низькочастотного сигналу (повідомлення).

Несуче коливання гармонійних видів модуляції можна представити як

$$U_0 \cos(\omega_0 t + \varphi_0), \quad (1)$$

де U_0 , ω_0 і φ_0 - відповідно постійні амплітуда, кутова частота і початкова фаза гармонічного коливання.

Модулюючий (інформаційний) сигнал $f(t)$ при амплітудній модуляції впливає на постійну амплітуду коливань несучої частоти U_0 , змінюючи її пропорційно величині модулюючого сигналу:

$$U_0 + \Delta U f(t), \quad (2)$$

де ΔU - найбільше відхилення амплітуди АМ-коливання. Тоді АМ-коливання можна записати як

$$U_0 \left[1 + m_a f(t) \right] \cos(\omega_0 t + \varphi_0), \quad (3)$$

де відношення $\frac{\Delta U}{U_0} = m_a$ називається коефіцієнтом амплітудної модуляції. Щоб уникнути явища перемодуляції, при якому на виході модулятора різко розширюється спектр модульованого сигналу, коефіцієнт m_a вибирається менше одиниці. З урахуванням цього, вираз (3) запишеться як

$$U_0 \left[1 + m_a f(t) \right] \cos(\omega_0 t + \varphi_0). \quad (4)$$

Якщо інформаційний сигнал представляє собою гармонійне коливання однієї частоти з одиничною амплітудою $f(t) = \cos \Omega t$ (рис.2, а), то, з урахуванням (4), отримаємо:

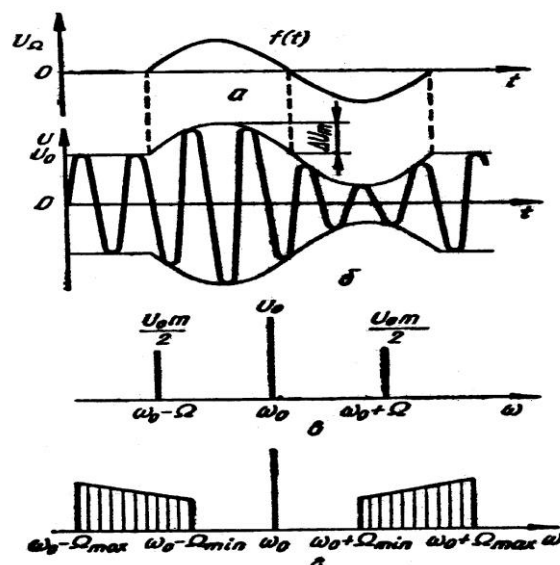


Рисунок 2 – Графіки моделюючого (а) і модульованого(б) сигналів; спектри амплітуд АМ-сигналу для простого (в) в складного (г) моделюючих сигналів

$$U_{\Sigma} = U_m \cos(\omega_0 t + \varphi) + U_m \cos(\omega_0 t - \Omega t + \varphi) + U_m \cos(\omega_0 t - \Omega t - \varphi) \quad (5)$$

де перший доданок описує не модульоване несуче коливання, другий і третій доданки з частотами $(\omega_0 + \Omega)$ і $(\omega_0 - \Omega)$ - верхню і нижню бокові частоти. Графік амплітудно модульованого коливання наведений на рис. 2, б. Спектр АМ-сигналу для розглянутого випадку містить несучу і дві бокові частоти (див рис.2, в). Ширина спектра при цьому рівна 2Ω .

При модуляції несучої частоти складним сигналом, що має широкий спектр частот, АМ-коливання буде містити верхню і нижню бокові полоси частот (див рис. 2, г). При цьому ширина спектра визначається значенням подвоєної максимальної частоти спектра модулюючого сигналу:

$$\Delta\omega_{AM} = 2\Omega_{max}.$$

Так як бокові смуги частот є дзеркальним відображенням один одного відносно несучої частоти і несуть одну і ту ж інформацію про інформаційний сигнал, то на практиці для зменшення смуги, що займає АМ-сигнал, підвищення завадостійкості та більш ефективного використання лінії зв'язку передачу, як правило, здійснюють на одній боковій смузі, зрізаючи одну з бокових полос відповідним фільтром. Така передача отримала назву *передачі на одній боковій смузі* (ОБС) і застосовується в багатьох системах передачі інформації. При цьому полоса частот, що передаються, скорочується не менш ніж в два рази.

Переваги ОБС:

у два рази збільшується кількість каналів зв'язку;

у чотири рази збільшується завадостійкість;

при прийомі до двох разів зменшується потужність завади.

Частотна модуляція

Несуче коливання $U_{\Sigma} = U_m \cos(\omega_0 t + \varphi)$

При частотній модуляції модулюючий сигнал $f(t) = \cos \Omega t$ впливає на частоту ω_0 несучого коливання, яка змінюється в часі відносно свого центрального значення за законом інформаційного сигналу $f(t)$:

$$\omega = \omega_0 + \Delta\omega \cos \Omega t, \quad (6)$$

де $\Delta\omega$ - девіація частоти, тобто відхилення кутової частоти ω_0 від центрального значення ω_0 .

З врахуванням (6) отримаємо:

$$U_{\Sigma} = U_m \cos(\omega_0 t + \Delta\omega \cos \Omega t + \varphi) \quad (7)$$

Відношення $\frac{\Delta\omega}{\Omega} = m_f$ називається *індексом (коефіцієнтом) частотної модуляції*. Так

як до виразу для модульованого сигналу входить постійна частота, то амплітуду коливання носія можна описати виразом

$$u = U_m \cos \theta, \quad (8)$$

де θ - фаза коливання, пов'язана з частотою співвідношенням

$$\omega = \frac{d\theta}{dt}.$$

Вид ЧМ-сигналу з постійною амплітудою наведений на рис. 3, а. При ЧМ фаза коливання відповідає (9).

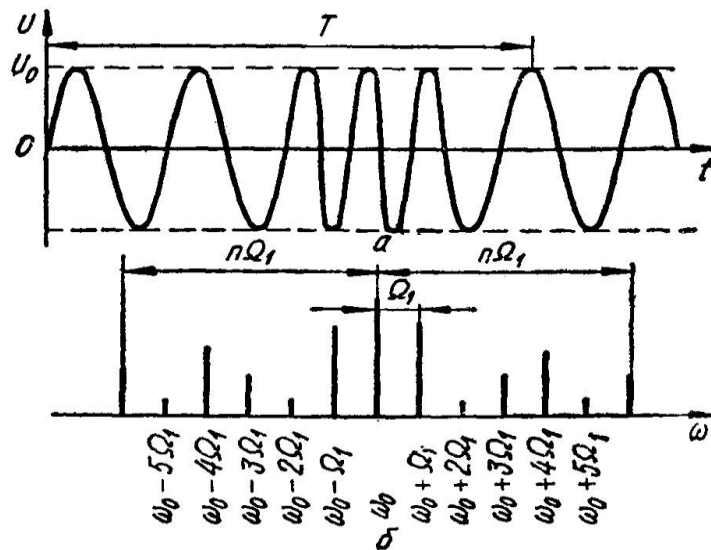


Рисунок 3 – Частотно-модульований сигнал (а) і його спектр (б)

$$\theta(t) = \omega_0 t + m_f \sin(\Omega_1 t) \quad (9)$$

Підставивши вираз (9) в (8), отримаємо

$$u(t) = U_0 \cos\left[\omega_0 t + m_f \sin(\Omega_1 t)\right] \quad (10)$$

Якщо моделююче коливання є інформаційним сигналом одної частоти Ω , то спектр ЧМ-сигналу (рис. 3,б) має дві бокові смуги, які містять нескінченну послідовність гармонічних коливань, віддалених один від одного на Ω і спадаючих по амплітуді по мірі віддалення від несучої частоти.

Приблизно ширину спектра ЧМ-сигналу можна визначити з виразу

$$\Delta\omega_{\text{ЧМ}} \approx 2\pi m_f.$$

Видно, що чим більше індекс частотної модуляції m_f , тим вужче, практично, необхідний спектр ЧМ-сигналу.

В залежності від вибраного індексу модуляції m_f розрізняють *вузькосмугову частотну модуляцію* з малими індексами m_f і *широкосмугову* – з великими індексами m_f . При вузькосмуговій ЧМ ширина спектра наближається до АМ, а при широкосмуговій – набагато більше, ніж при АМ.

Основною перевагою широкосмугової ЧМ є висока завадостійкість, значно більша ніж при АМ, так як частота сигналу менше зазнає впливу завад, ніж амплітуда.

Демодуляція АМ сигналів

Демодуляція АМ-сигналу може здійснюватися декількома методами [5]. Найпростіший метод — симулювати роботу аналогового двопівперіодного детектора [7]. Ми обчислюємо модуль вхідного АМ-сигналу, а потім згладжуємо отримані одно полярні косинусоїдальні імпульси, пропускаючи їх через ФНЧ. Проте даний метод, не буде

працювати правильно у випадку перемодуляції. Інший метод — синхронне детектування, суть якого полягає в домноженні частоти сигналу на опорне коливання з несучою частотою:



(11)

Результат перемноження містить дві складові. Перша — це вагома амплітудна функція, друга — АМ сигнал з несучою частотою $2\omega_0$. Цей високочастотний сигнал легко видаляється шляхом пропускання сигналу через ФНЧ.

Порядок виконання роботи

1. Сформуванню моделюючий та несучий сигнал на заданому проміжку згідно заданого у варіанті виразу.
2. Сформуванню модульований сигнал із заданими параметрами згідно варіанту за виразом (1).
3. Використовуючи вираз (4) отримати АМ-сигнал.
4. Визначити частотний спектр АМ-сигналу, використовуючи базові функції Matlab.
5. На основі виразу (11) виконати демодуляцію АМ-сигналу.
6. Застосовуючи базові функції фільтрації виконати фільтрацію демодульованого за виразом (11) сингала та виділити з нього корисний сигнал.

Наприклад :

$[b,a]=butter(5, 2*\omega/\pi/Fs)$, обчислення коефіцієнтів фільтра, ω частота несучого коливання, Fs - частота дискретизації.

$Z=filtfilt(b,a, y)$ – фільтрація сигналу y фільтром з коефіцієнтами b,a , де 5 - порядок фільтра.

Варіанти завдань наведені в таблиці 1. Приклад програми написаної на Matlab, яка виконує модуляцію та демодуляцію складеного сигналу наведено в додатку.

Вимоги до звіту до лабораторної роботи

- мета роботи;
- варіант завдання ;
- графіки:
 - а) вхідного (модулюючого) сигналу та його частотний спектр;
 - б) несучого коливання;
 - в) АМ-сигналу;
 - г) модульованого АМ-сигналу з подвоєною несучою;
 - д) демодульованого корисного сигналу;
- висновки згідно отриманих результатів.

Контрольні запитання

- Що таке модуляція та демодуляція?
- Пояснити суть амплітудної, частотної модуляції.
- Типи модуляції.
- Зміст поняття глибини модуляції.
- Що таке перемодуляція?
- Як відбувається процес демодуляції АМ-сигналів.
- Які області застосування модульованих сигналів.

Варіанти завдань наведені в таблиці 1.

Таблиця 1 Варіанти завдань

№	Вхідний сигнал, $t=0..1000$	Частот и вхідно го сигнал у, Гц, ω_n	Тип функції несучого коливання	Частот а несучо го колива ння ω_0 , Г ц	Ампліту -да несучого коливан ня U_0	Фаза несучо-го коливання , φ_0	Частота дискети- зації F_s , Гц	Коефіціє нт модуляц ії m_a
1	$\sum_{n=1}^3 \sin(\frac{2\pi\omega_n t}{F_s})$	1,5 5,3 6,8	$U_0 \cos(\frac{2\pi\omega_0 t}{F_s})$	10π	1	$\pi/1$	100	0.1
2	$\sum_{n=1}^3 \cos(\frac{2\pi\omega_n t}{F_s})$	2,4 4,0 7,0	$U_0 \sin(\frac{2\pi\omega_0 t}{F_s})$	11π	2	$\pi/2$	150	0.25
3	$\sum_{n=1}^4 \sin(\frac{2\pi\omega_n t}{F_s})$	1,8 2,9 4,6 7,3	$U_0 \cos(\frac{2\pi\omega_0 t}{F_s})$	9π	3	$\pi/3$	200	0.36
4	$\sum_{n=1}^4 \cos(\frac{2\pi\omega_n t}{F_s})$	1,0 1,4 1,8 2,0	$U_0 \sin(\frac{2\pi\omega_0 t}{F_s})$	5π	4	$\pi/4$	250	0.02
5	$\sum_{n=1}^3 \sin(\frac{2\pi\omega_n t}{F_s})$	0,5 1,9 2,9	$U_0 \cos(\frac{2\pi\omega_0 t}{F_s})$	4π	5	$\pi/5$	300	0.86
6	$\sum_{n=1}^3 \cos(\frac{2\pi\omega_n t}{F_s})$	2,5 3,0 7,0	$U_0 \sin(\frac{2\pi\omega_0 t}{F_s})$	8π	6	$\pi/6$	350	0.78
7	$\sum_{n=1}^4 \sin(\frac{2\pi\omega_n t}{F_s})$	3,0 5,1 5,2 8,2	$U_0 \cos(\frac{2\pi\omega_0 t}{F_s})$	15π	7	$\pi/7$	400	0.6
8	$\sum_{n=1}^4 \cos(\frac{2\pi\omega_n t}{F_s})$	3,5 6,0 8,5 9,8	$U_0 \sin(\frac{2\pi\omega_0 t}{F_s})$	13π	8	$\pi/8$	450	1.0
9	$\sum_{n=1}^3 \sin(\frac{2\pi\omega_n t}{F_s})$	1,6 3,3 5,1	$U_0 \cos(\frac{2\pi\omega_0 t}{F_s})$	18π	9	$\pi/9$	500	0.236
10	$\sum_{n=1}^3 \cos(\frac{2\pi\omega_n t}{F_s})$	1,2 6,1 7,9	$U_0 \sin(\frac{2\pi\omega_0 t}{F_s})$	26π	10	$\pi/10$	550	0.95
11	$\sum_{n=1}^4 \sin(\frac{2\pi\omega_n t}{F_s})$	1,0 2,2 2,4 3,7	$U_0 \cos(\frac{2\pi\omega_0 t}{F_s})$	20π	11	$\pi/11$	600	0.63
12	$\sum_{n=1}^4 \cos(\frac{2\pi\omega_n t}{F_s})$	2,6 3,7 4,2 5,8	$U_0 \sin(\frac{2\pi\omega_0 t}{F_s})$	7π	12	$\pi/12$	650	0.18

13	$\sum_{n=1}^3 \sin(\frac{2\pi w_n t}{Fs})$	2,6 6,8 7,9	$U_{fc}(\frac{2\pi w_n t}{Fs})$	22 π	13	$\pi/13$	700	1.0
14	$\sum_{n=1}^3 \cos(\frac{2\pi w_n t}{Fs})$	0,2 0,7 2,6	$U_{fs}(\frac{2\pi w_n t}{Fs})$	3 π	14	$\pi/14$	750	0.57
15	$\sum_{n=1}^4 \sin(\frac{2\pi w_n t}{Fs})$	0,2 1,8 5,3 8,3	$U_{fc}(\frac{2\pi w_n t}{Fs})$	19 π	15	$\pi/15$	800	0.5
16	$\sum_{n=1}^4 \cos(\frac{2\pi w_n t}{Fs})$	4,3 7,3 8,6 11,	$U_{fs}(\frac{2\pi w_n t}{Fs})$	16 π	16	$\pi/16$	850	0.37
17	$\sum_{n=1}^3 \sin(\frac{2\pi w_n t}{Fs})$	4,6 10,0 12,0	$U_{fc}(\frac{2\pi w_n t}{Fs})$	17 π	17	$\pi/17$	900	0.74
18	$\sum_{n=1}^3 \cos(\frac{2\pi w_n t}{Fs})$	0,05 0,26 1,3	$U_{fs}(\frac{2\pi w_n t}{Fs})$	3 π	18	$\pi/18$	950	0.65
19	$\sum_{n=1}^4 \sin(\frac{2\pi w_n t}{Fs})$	2,3 4,9 5,06 6,8	$U_{fc}(\frac{2\pi w_n t}{Fs})$	13 π	19	$\pi/19$	1000	0.79
20	$\sum_{n=1}^4 \cos(\frac{2\pi w_n t}{Fs})$	0,5 1,4 1,59 2,6	$U_{fs}(\frac{2\pi w_n t}{Fs})$	5 π	20	$\pi/20$	125	0.9
21	$\sum_{n=1}^3 \sin(\frac{2\pi w_n t}{Fs})$	6,0 9,1 9,9	$U_{fc}(\frac{2\pi w_n t}{Fs})$	16 π	21	$\pi/21$	175	0.6
22	$\sum_{n=1}^3 \cos(\frac{2\pi w_n t}{Fs})$	0,06 0,26 0,8	$U_{fs}(\frac{2\pi w_n t}{Fs})$	2 π	22	$\pi/22$	225	0.85
23	$\sum_{n=1}^4 \sin(\frac{2\pi w_n t}{Fs})$	4,8 5,1 5,16 6,0	$U_{fc}(\frac{2\pi w_n t}{Fs})$	21 π	23	$\pi/23$	275	0.94
24	$\sum_{n=1}^4 \cos(\frac{2\pi w_n t}{Fs})$	6,8 7,28 8,2 9,0	$U_{fs}(\frac{2\pi w_n t}{Fs})$	22 π	24	$\pi/24$	325	0.36
25	$\sum_{n=1}^3 \sin(\frac{2\pi w_n t}{Fs})$	4,0 6,7 8,5	$U_{fc}(\frac{2\pi w_n t}{Fs})$	24 π	25	$\pi/25$	375	0.49
26	$\sum_{n=1}^3 \cos(\frac{2\pi w_n t}{Fs})$	2,3 3,6 4,2	$U_{fs}(\frac{2\pi w_n t}{Fs})$	10 π	26	$\pi/26$	425	0.92
27	$\sum_{n=1}^4 \sin(\frac{2\pi w_n t}{Fs})$	2,3 3,6 4,9 7,2	$U_{fc}(\frac{2\pi w_n t}{Fs})$	18 π	27	$\pi/27$	625	0.63

28	$\sum_{n=1}^4 \cos(\frac{2\pi w_n t}{Fs})$	2,6 4,8 5,1 6,2	$U_{fs} \cos(\varphi/Fs)$	26π	28	π/28	825	0.45
29	$\sum_{n=1}^3 \sin(\frac{2\pi w_n t}{Fs})$	6,5 8,8 9,6	$U_{fs} \cos(\varphi/Fs)$	17π	29	π/29	975	0.56
30	$\sum_{n=1}^3 \cos(\frac{2\pi w_n t}{Fs})$	1,5 2,5 3,6	$U_{fs} \cos(\varphi/Fs)$	29 π	30	π/30	525	0.25

Приклад виконання роботи

№	Вхідний сигнал, t=0..1000	Частоти вхідно- го сигналу , Гц, ω _n	Тип функції несучого коливання	Частота несучого коливанн я ω ₀ , Гц	Ампліту- да несучого коливанн я U ₀	Фаза несучо- го коли- вання, φ ₀	Частота дискретиза- ції Fs, Гц	Коефіцієнт модуляції m _a
61	$\sum_{n=1}^4 \sin(\frac{2\pi w_n t}{Fs})$	0,06 1,65 4,85 8,3	$U_{fs} \cos(\varphi/Fs)$	40 π	31	π/31	875	0.5

1. Формуємо модулюючий сигнал:



2. Формуємо несучий сигнал:



3. Знаходимо АМ сигнал згідно виразу (4)



4. Обчислюємо перетворення Фур'є для сигналу $U_{AM}(t)$

5. Виконуємо демодуляцію сигналу $U_{AM}(t)$ за виразом (11)



6. Виконуємо фільтрацію сигналу $y_{DM}(t)$

$$[b,a]=butter(5, 2*80\pi/\pi/Fs)$$

$$Z=filtfilt(b,a, y_{DM}(t))$$

Лабораторна робота № 7

Тема. Діагностика роботи цифрових фільтрів шляхом аналізу їх амплітудно-частотної характеристики

Мета роботи: Дослідити і проаналізувати параметри амплітудно-частотної характеристики та вплив віконної обробки при спектральному аналізі сигналів.

Теоретичне підґрунтя

Для адекватного відтворення вхідного сигналу, що використовується в системах обробки, які розв'язують задачі спектрального аналізу сигналів, опис вхідного діагностичного сигналу представляється у формалізованому вигляді. Зазначені задачі розв'язуються цифровими методами, на основі швидких дискретних ортогональних перетворень, що представляються узагальненим класом швидких перетворень Фур'є з різними системами базисних функцій. Дані перетворення відносяться до класу лінійних ортогональних перетворень, зв'язаних з обчисленням виразів виду

$$X = \frac{1}{L} Ax,$$

де $X = [X(0), X(1), \dots, X(L-1)]^T$, $x = [x(0), x(1), \dots, x(L-1)]^T$ - вектори, відповідно, вихідних гармонік і початкових відліків, A - відтворююча ортогональна матриця розміром $L \times L$, L - кількість початкових відліків.

Системи, які реалізують ці алгоритми відносяться до стаціонарних систем з частотним коефіцієнтом передачі $K(j\omega)$:

$$K(j\omega) = \int_{-\infty}^{\infty} h(t)e^{-j\omega t} dt$$

де $h(t)$ - імпульсна характеристика, що має таку інтерпретацію: якщо на вхід системи поступає гармонійний сигнал з відомою частотою ω і комплексною амплітудою $\dot{U}_{вх}$, то комплексна амплітуда вихідного сигналу $\dot{U}_{вих}$ буде рівною:

$$\dot{U}_{вих} = \dot{K}(j\omega)\dot{U}_{вх} \quad (1)$$

Представлення частотного коефіцієнта передачі (див. формулу 1) в показниковій формі має вигляд :

$$K(j\omega) = |K(j\omega)|e^{j\varphi_k\omega},$$

де $|K(j\omega)|$ - амплітудно-частотна характеристика (АЧХ).

Оскільки для фільтрів з скінченною імпульсною характеристикою АЧХ є однією з визначальних характеристик, на основі її аналізу визначається достовірність побудови фільтра. Розглянемо варіант перевірки фільтра методом аналізу його АЧХ на прикладі системи опрацювання інформації когерентно-імпульсної РЛС з n каналами погоджених фільтрів. Для процесора, що виконує N -точкове амплітудне дискретне перетворення Фур'є згідно з формулою (2)

$$Y(n, l) = \sum_{i=0}^{N-1} \dot{U}(n, i)W(i)e^{-j(2\pi / N)li}, \quad (2)$$

де N визначає розмір перетворення, n -номер елемента віддалі, l – номер гармоніки, i -номер періоду повторення в межах інтервалу обчислення ДПФ, $W(i)$ вагова функція,

вхідний сигнал $\dot{U}_{(n,i)}$ представимо у вигляді:

$$\dot{U}_{(n,i)} = Ae^{+j2\pi(Sl+Q)i / SN}, \quad (3)$$

де A - амплітуда сигналу, S - кількість частотних діапазонів між сусідніми l , Q - визначає смугу перевірки АЧХ ($Q = -[S_{m-1} + S_i], [S_{p-1} + S_i]$, де m, p - кількість гармонік, в діапазоні яких (відносно l) перевіряється АЧХ, $m = \overline{1, N-1}$, $p = \overline{1, N-1}$, s_i - біжуче значення частотного діапазону між сусідніми l).

Процедура діагностики відбувається таким чином. Для процесора задається значення гармоніки l_j . На його інформаційні входи поступає вхідний сигнал $\dot{U}(n, i)$. Зміна значень $\dot{U}(n, i)$ (синфазна і квадратурна складові) на вході процесора відбувається на кожному періоді повторення (по i). Одне значення $Y(n, l)$ визначається сумуванням по i (див. формулу 2). Після того змінюється частота поступлення $\dot{U}(n, i)$, зміна задається значенням $s_i = \overline{0, S}$, і вираховується наступне значення $Y(n, l)$.

Повна АЧХ, для заданого l_j , отримується після поступлення на вхід $S \cdot N$ значень вхідного сигналу. На практиці обмежуються перевіркою АЧХ для $\pm 3l$, відносно l_j . Після перевірки амплітудно-частотних характеристик для всіх гармонік і елементів віддалі процес діагностики завершується. В ідеальному випадку характеристики всіх АЧХ повинні бути ідентичними.

Тобто, при використанні такого підходу процес перевірки розбивається на три етапи:

- задання значень для отримання числової послідовності вхідних сигналів;
- визначення значень $Y(n, l)$ реальної АЧХ;
- порівняння значень ідеальної і реальної АЧХ в кожній точці виміру.

Застосування підходу дозволяє:

- виявити помилки в роботі з точністю до функціонального вузла, наприклад помилки в заданні вагової функції, при сумуванні, в ОЗП проміжних результатів, при пересиланні інформації між процесорами, конструктивні та технологічні помилки при проектуванні цифрових вузлів і т.п.;

- проводити діагностику в режимі реального часу;
- перевірити правильність функціонування і рівень шумів зовнішніх пристроїв, наприклад, приймача проміжної частоти;
- оцінити вплив різних типів вагових функцій на значення вихідного сигналу;
- перевірити в РРЧ значення інформації, що поступає на вхід системи опрацювання шляхом її запису в ОЗП;
- перевірити точнісні параметри роботи процесорів;
- перевірити реакцію фільтра на поступлення збійної інформації.

Найвживаніші вагові функції, що використовуються при обробці наведені в таблиці

1.

Таблиця 1

Номер функції i	Назва	Тип функції	Діапазон зміни n
1	Рімана	$w[n] = \frac{\sin\left[\frac{n}{N} 2\pi\right]}{\left[\frac{n}{N} 2\pi\right]}$	$-N/2 \leq n \leq N/2 - 1$
2	Валле-Пусена	$w[n] = 1,0 - 6\left[\frac{n}{N/2}\right]^2 \left[1,0 - \frac{ n }{N/2}\right]$ $w[n] = 2\left[1,0 - \frac{n}{N/2}\right]^3$	$0 \leq n \leq N/4$ $N/4 \leq n \leq N/2$

3	Тюкі	$w[n] = 1,0$ $w[n] = 0,5 \left[1,0 + \cos \left[\pi \frac{n - a \frac{N}{2}}{2(1-a) \frac{N}{2}} \right] \right]$	$0 \leq n \leq a N/2$ $a N/2 \leq n \leq N/2$ 3-1 $a = 0,25$ 3-2 $a = 0,5$ 3-3 $a = 0,75$
4	Бомана	$w[n] = \left[1,0 - \frac{ n }{N/2} \right] \cos \left[\pi \frac{ n }{N/2} \right] + \frac{1}{\pi} \sin \left[\pi \frac{ n }{N/2} \right]$	$0 \leq n \leq N/2$
5	Пуасона	$w[n] = \exp \left[-a \frac{ n }{N/2} \right]$	$0 \leq n \leq N/2$ 5-1 $a = 2,0$ 5-2 $a = 3,0$ 5-3 $a = 4,0$
6	Хеннінга-Пуасона	$w[n] = 0,5 \left[1,0 + \cos \left[\pi \frac{ n }{N/2} \right] \right] \exp \left[-a \frac{ n }{N/2} \right]$	$0 \leq n \leq N/2$ 6-1 $a = 0,5$ 6-2 $a = 1,0$ 6-3 $a = 2,0$
7	Коші	$w[n] = \frac{1}{1,0 + \left[a \frac{ n }{N/2} \right]^2}$	$0 \leq n \leq N/2$ 7-1 $a = 3,0$ 7-2 $a = 4,0$ 7-3 $a = 5,0$
8	Трикутне	$w[n] = 1 - \frac{ n }{N}$	$0 \leq n \leq N/2$
9	Ханна (косинус квадрат)	$w[n] = \cos^2 \frac{\pi}{N} n = \frac{1}{2} - (1 - \cos \frac{2\pi}{N} n)$	$0 \leq n \leq N/2$
10	Геммінга	$w[n] = a - (1 - a) \cos \frac{2\pi}{N} n$	$0 \leq n \leq N/2$ $a = 0,54$
11	Блекмана	$w[n] = 0,42 - 0,5 \cos \frac{\pi}{N} n + 0,08 \cos \frac{4\pi}{N} n$	$0 \leq n \leq N/2$
12	Гауса	$w[n] = \exp - \left[-\frac{1}{2} \left(\frac{a n}{N/2} \right)^2 \right]$	$0 \leq n \leq N/2$ $a = 2,5$
13	Cos ^a	$w[n] = \cos^a \frac{\pi}{N} n$	$0 \leq n \leq N/2$ 13-1 $a = 1,0$ 13-2 $a = 3,0$ 13-3 $a = 4,0$
14	Рісса	$w[n] = 1 - \left \frac{n}{N/2} \right ^2$	$0 \leq n \leq N/2$
15		$w(n) = 0.25 + 0.75 \cos [\pi(n-16)/32]$	
16		$w(n) = 0.4 + 0.6 \cos [\pi(n-15,5)/31].$	

Примітка: Значення $w(n)$ таблиці 1 відповідає значенню $W(i)$ (див. формулу 2).

Алгоритм формування вхідних даних для формування АЧХ полягають у видачі на кожному етапі обчислень синусоїдальної і косинусоїдальної складової комплексного сигналу, фаза яких відрізняється на значення Q на двох сусідніх періодах, на кожному з яких обчислюється одне значення $U(i)$

Порядок виконання роботи

1. Налаштувати фільтр на виконання заданого варіанту - сформувати масив синусоїдальної і косинусоїдальної складової згідно з виразом:
2. Сформувати вхідний масив (синусоїдальна і косинусоїдальна складові) згідно з формулою 3.

$$e^{-j(2\pi / N)li}$$

3. Сформувати масив вагової функції
4. Скласти процедуру на мові високого рівня для обчислення АЧХ згідно з формулою 2.
5. Скласти процедуру графічного виводу значень АЧХ: без вагової функції та з ваговою функцією.
6. Порівняти значення АЧХ, пояснити отримані результати.

Примітка: Передбачити можливість зміни в програмі всіх вхідних параметрів і констант.

Зміст звіту до лабораторної роботи

1. Завдання на лабораторну роботу.
2. Теоретичний матеріал.
3. Лістинг підпрограми і результати формування вхідного масиву (таблиця або графік).
4. Лістинг підпрограми і результати формування масиву вагової функції (таблиця або графік).
5. Лістинг програми і результати формування АЧХ (таблиця і графік) для двох випадків: без вагової функції та з ваговою функцією.
6. Висновки.

Контрольні запитання

4. Дайте визначення АЧХ.
5. Фізичний зміст гармоніки?
6. Опишіть процедуру проведення діагностування.
7. На якому етапі завершується діагностування?
8. Які переваги має використання АЧХ для перевірки цифрових фільтрів?

Варіанти завдань до лабораторної роботи

№ вар	N	l	S_m	S_p	S	A	№ вагової функції
1	16	0; 5	-32	32	16	1	15
2	16	0; 8	-24	24	16	2	16
3	16	7	-48	48	16	3	14
4	16	9	-64	16	16	4	13-1
5	16	4	-16	16	8	5	13-2
6	16	10	-20	16	8	6	13-3
7	16	0; 15	-22	16	8	7	12
8	16	5	-26	16	8	8	11
9	16	0; 7	-28	16	16	9	10
10	16	0; 11	-34	16	16	10	9
11	16	8	-36	16	16	11	8
12	16	6	-38	16	16	12	7-1
13	16	3	-40	16	8	13	7-2
14	16	2	-42	16	8	14	7-3
15	16	0; 12	-44	16	8	15	6-1
16	32	14	-32	32	8	1	6-2
17	32	0; 31	-48	32	24	2	6-3
18	32	0; 15	-48	48	24	3	5-1
19	32	3	-32	48	24	4	5-2
20	32	5	-48	64	24	5	5-3
21	32	7	-48	50	8	6	4
22	32	9	-64	32	8	7	3-3
23	32	11	-64	40	8	8	3-2
24	32	13	-64	42	8	9	3-3
25	32	17	-34	44	16	10	2
26	32	19	-36	44	16	11	1
27	32	21	-38	46	16	12	15
28	32	23	-54	48	16	13	16
29	32	25	-56	60	24	14	14
30	32	27	-58	32	24	15	13-1

Приклад виконання

Завдання:

Проаналізувати амплітудно-частотну характеристику фільтру з такими параметрами сигналу: $l = 0, 1, \dots, 31$; $A = 1, 2, \dots, 100$; $S = 8, 16$; $Q = -64 \dots 64$; $N = 0, 1, \dots, 31$.

Хід роботи

Згідно поданих вище формул при заданих значення параметрів вхідного сигналу та цифрового фільтру, що тестується будуємо масиви значень синусів та косинусів за таким алгоритмом:

```

for(i = 0; i < N; i++)
{
    sin0[i] = sin(2*pi*i/N);
    cos0[i] = cos(2*pi*i/N);
}
for(i = 0; i < SN; i++)
{

```

```

sin1[i] = sin(2*pi*i/SN);
cos1[i] = cos(2*pi*i/SN);

```

}, де N – розмір перетворення, а SN – добуток N та S .

Далі обчислюється значення дійсної та уявної частини, що утворилися як результат добутку $Ae^{+j2\pi(Sl+Q)i/SN}$ та $e^{-j(2\pi/N)li}$ представлених у тригонометричній формі. Значення синусів та косинусів вибираємо із масивів отриманих на попередньому етапі.:

```

for(i = 0; i < N; i++)
{
    a = (abs(S*l+Qmin)*i)%SN;
    b = l*i%N;
    Re += A*cos1[a]*cos0[b] + A*sin1[a]*sin0[b];
    Im += A*cos1[a]*sin0[b] - A*sin1[a]*cos0[b];
}

```

Обчислюємо значення координати Y для відображення результату:

$$Y[j] = \text{sqrt}(\text{pow}(\text{Re}, 2) + \text{pow}(\text{Im}, 2));$$

Повний лістинг програми подано у Додатку.

Методика проведення дослідження

При запуску програми з'являється головне вікно (див. рис.1):

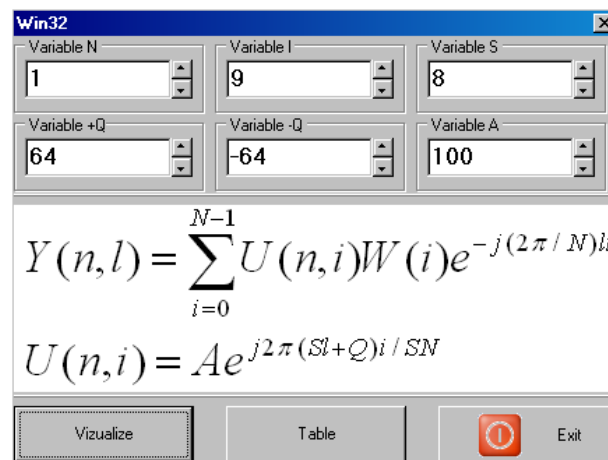


Рисунок 1 – Головне вікно програми

На рис.1. наведені основні формули, що використовується при проведенні аналізу та поля, в яких можна задати параметри сигналу та фільтру, що досліджується. Змінювати дані параметри можна натискаючи мишкою на стрілки, що містяться біля кожного поля праворуч.

Для перегляду результату у графічному представленні натисніть клавішу <Vizualize>. Графік АЧХ наведений на рис.2.

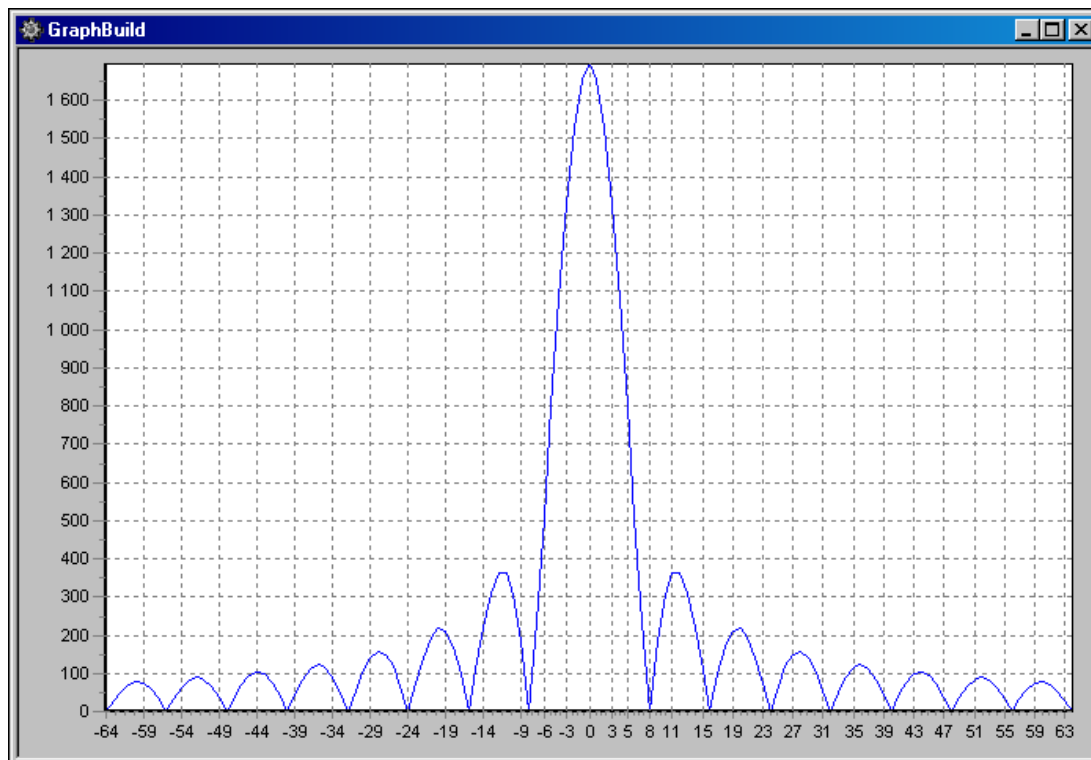


Рисунок 2 – Графік АЧХ

Для отримання табличного представлення результату натисніть клавішу <Table>. Фрагмент значень АЧХ наведений в таблиці на рис.3.

X	Y
-64	2,75764738977458E-14
-63	29,0419697584423
-62	54,3501896863475
-61	71,9446715207641
-60	78,9208447262436
-59	73,9197598843918
-58	57,3763281911878
-57	31,5023737798159
-56	4,93558757932881E-14
-55	32,4588524304012
-54	60,9147189069376
-53	80,8665934638205
-52	88,9710624140459
-51	83,5879042571257
-50	65,0853182734446

Рисунок 3 – Значення АЧХ (фрагмент)

Висновок: Розроблена програма дозволяє провести діагностику цифрових фільтрів шляхом аналізу їх амплітудно-частотної характеристики при широкому спектрі параметрів вхідного сигналу. Графічне представлення результату дозволяє легше сприймати та оцінювати інформацію, а таблиця подає точні результати обчислень на всій смузі перевірки.

Program ACH

```
//-----

#include <vcl.h>
#pragma hdrstop
USERES("Project1.res");
USEFORM("Unit1.cpp", Form1);
USEFORM("Unit2.cpp", Graph); /* TFrame: File Type */
USEFORM("Unit3.cpp", Form3);
USEFORM("Unit4.cpp", Form4);
//-----
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize();
        Application->Title = "CMP";
        Application->CreateForm(__classid(TForm1), &Form1);
        Application->CreateForm(__classid(TForm4), &Form4);
        Application->CreateForm(__classid(TGraph), &Graph);
        Application->CreateForm(__classid(TForm3), &Form3);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    return 0;
}
//-----

//-----
//define Nv          32//
//define Sv          8//Edit4->Text.ToInt()
//define SN          (Sv*Nv)//(Edit2->Text.ToInt()*Edit4->Text.ToInt())
//define Iv          1//Edit3->Text.ToInt()
//define Av          1

    double Y[1024];

    double sin0[32];
    double sin1[512];
    double cos0[32];
    double cos1[512];

//-----

#include <vcl.h>
#include <stdio.h>
#include "math.h"

#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"

//-----
#pragma package(smart_init)
#pragma link "Unit2"
#pragma resource "*.dfm"
void Apply(void);
```

```

TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
: TForm(Owner)
{
}
//-----

void __fastcall TForm1::Button3Click(TObject *Sender)
{
    Apply();
    Form3->Show();
}
//-----

void __fastcall TForm1::BitBtn1Click(TObject *Sender)
{
    Form1->Close();
}
//-----

void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Apply();
    Form4->Show();
    Form4->StringGrid1->Cells[0][0]="X";
    Form4->StringGrid1->Cells[1][0]="Y";
}
//-----

void __fastcall TForm1::QpChange(TObject *Sender)
{
    char buf[5];
    if(Qp->Text.ToInt()<= Qm->Text.ToInt())
        Qp->Text=itoa((Qm->Text.ToInt()+1),buf,10);
    Form4->StringGrid1->RowCount=abs(Qm->Text.ToInt())+abs(Qp->Text.ToInt());
}
//-----

void __fastcall TForm1::QmChange(TObject *Sender)
{
    char buf[5];
    if(Qp->Text.ToInt()<= Qm->Text.ToInt())
        Qm->Text=itoa((Qp->Text.ToInt()-1),buf,10);
    Form4->StringGrid1->RowCount=abs(Qm->Text.ToInt())+abs(Qp->Text.ToInt());
}
//-----

void Apply(void)
{
    int N = Form1->Edit2->Text.ToInt();
    int S = Form1->Edit4->Text.ToInt();
    int SN = (S*N);
    int l = Form1->Edit3->Text.ToInt();
    int A = Form1->Edit1->Text.ToInt();

    int Qmin = Form1->Qm->Text.ToInt();
    int Qmax = Form1->Qp->Text.ToInt();
    //*****
    char buf[55];
    char *buf_ptr;
        int j = 0;
        int i;
        double pi = 3.141592653589793;

```

```

        double Re = 0;
        double Im = 0;
        int a;
        int b;
//*****
Form3->Series1->Clear();
        for(i = 0; i < N; i++)
        {
                sin0[i] = sin(2*pi*i/N);
                cos0[i] = cos(2*pi*i/N);
        }

        for(i = 0; i < SN; i++)
        {
                sin1[i] = sin(2*pi*i/SN);
                cos1[i] = cos(2*pi*i/SN);
        }

        do
        {
                for(i = 0; i < N; i++)
                {
                        a = (abs(S*I+Qmin)*i)%SN;
                        b = i*i%N;
                        Re += A*cos1[a]*cos0[b] + A*sin1[a]*sin0[b];
                        Im += A*cos1[a]*sin0[b] - A*sin1[a]*cos0[b];
                }

                Y[j] = sqrt(pow(Re, 2) + pow(Im, 2));
                Re = Im = 0;

        buf[0] = ' ';
        itoa(Qmin, buf+1, 10);
        buf[strlen(buf) + 1] = '\0';
        buf[strlen(buf)] = ' ';

        Form3->Series1->AddXY(j, Y[j], (itoa(Qmin, buf+1, 10)-1), clBlue);
        Form4->StringGrid1->Cells[1][j+1] = Y[j];
        Form4->StringGrid1->Cells[0][j+1] = Qmin;
        j++;

        }
        while(Qmin++ < Qmax);
}
//-----

```

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Попович Р.Б., Ваврук Є.Я. Цифрове опрацювання сигналів та зображень. Алгоритми та реалізація// Навчальний посібник. Вид-во Нац. ун-ту “Львівська політехніка”, 2008 р.
2. Лашко О.Л., Ваврук Є.Я. Основи обробки сигналів// Навчальний посібник. Вид-во Нац. ун-ту “Львівська політехніка”, 2009 р.
3. Ваврук Є. Алгоритми та засоби обробки сигналів : навч. посібн. / Ваврук Є., Лашко О., Попович Р. – Львів : СПОЛОМ, 2021. – 240 с. : іл. – Бібл.: с. 237–239 (34 назви).
4. Айфигер, Эммануил С., Джервис, Барри У. Цифровая обработка сигналов: практический подход, 2-е изд.: Пер. с англ. – М.: Издательский дом “Вильямс”, 2004. – 992с.
5. Бабак В.П., Хандецький А.І., Шрюфер Е. Обробка сигналів: підручник для вузів., К., Либідь, 1996.- 390с.
6. Цифровая обработка сигналов/ А.Б.Сергиенко – СПб.Питер, 2002.
7. Бондарев В.Н., Трестер Г., Чернега В.С. Цифровая обработка сигналов: методы и средства. - Харьков: Конус, 2001 (підручник для вузів).
8. Л. Рабинер, Б.Гоулд. Теория и применение цифровой обработки сигналов.- М.:Мир, 1978. – 848 с.
9. Попов Б.А., Теслер Г.С. Вычисление функций на ЭВМ - К.: Наукова думка, 1984 – 600с.
10. Винцюк Т.К. Анализ, распознавание и интерпретация речевых сигналов.-К., Наукова думка,1987. – 264с.
11. Young S. Large vocabulary continuous speech recognition. IEEE Signal Processing Magazine,13(5), 1996, pp.45-57.
12. Потемкин В.Г. Система MATLAB для студентов. Справочное пособие. – М.:ДИАЛОГ-МИФИ, 1998. – 314 с.
13. Сергиенко А. Б.Цифровая обработка сигналов: Учебник для вузов. 2-е изд.— СПб.:Питер,2007. – 751 с.

ЗМІСТ

ВСТУП.....	3
ЛАБОРАТОРНА РОБОТА №1. Аналіз обчислювальної похибки при виконанні базових операцій алгоритмів цифрової обробки сигналів. Обчислення математичних функцій.....	4
ЛАБОРАТОРНА РОБОТА №2. Розрахунок і побудова цифрових СІХ фільтрів з частотною вибіркою. Фільтрація складених сигналів.	16
ЛАБОРАТОРНА РОБОТА №3. Моделювання роботи препроцесора для попередньої обробки мовних сигналів.....	28
ЛАБОРАТОРНА РОБОТА №4. Стиск зображень з використанням дискретних косинусних перетворень.....	35
ЛАБОРАТОРНА РОБОТА №5. Фільтрація сигналів і зображень.....	43
ЛАБОРАТОРНА РОБОТА №6. Модуляція та демодуляція сигналів. Амплітудна модуляція складених сигналів.....	52
ЛАБОРАТОРНА РОБОТА №7. Діагностика роботи цифрових фільтрів шляхом аналізу їх амплітудно-частотної характеристики.....	61
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	71