Й.Й. Білинський, В.В. Мотигін

ОБРОБКА ІНФОРМАЦІЇ ЗАСОБАМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ

Міністерство освіти і науки України
Вінницький національний технічний університет

Й.Й. Білинський, В.В. Мотигін

ОБРОБКА ІНФОРМАЦІЇ ЗАСОБАМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ

Затверджено Вченою радою Вінницького національного технічного університету як лабораторний практикум для студентів спеціальностей "Виробництво електронних засобів", "Технології та засоби телекомунікацій". Протокол № 11 від "30" червня 2004р.

Вінниця ВНТУ 2005

УДК 621.374
Б 61

Рецензенти:

В.В. Кухарчук, доктор технічних наук, професор
О.М. Роїк, доктор технічних наук, професор
В.М. Лисогор, доктор технічних наук, професор

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Білинський Й.Й., Мотигін В.В.

Б 61 **Обробка інформації засобами комп'ютерної математики.** Лабораторний практикум. – Вінниця: ВНТУ, 2005.– 126с.

У лабораторному практикумі розглядаються основи теорії інформації і кодування. Зокрема розглядаються питання: спектральні характеристики сигналів; квантування сигналів; дослідження випадкових процесів на стаціонарність; методи обробки зображень; завадостійке кодування; методи стиснення даних.

Засобами для дослідження розглянутих питань є системи комп'ютерної математики MatchCAD, Maple і MatLAB, які призначені для виконання інженерних і наукових розрахунків, а також візуалізації отриманих результатів. Велика увага приділена командам, функціям і операторам використаних систем, прикладам практичного застосування, завданням і тестам для поточного контролю.

Лабораторний практикум розроблений у відповідності з навчальними програмами дисциплін "Експлуатація комп'ютеризованих систем обробки інформації" і "Електронні інформаційні системи і технології".

УДК 621.374

© Й.Й. Білинський, В.В. Мотигін, 2005

Зміст

ВСТУП 5

1 РОБОТА З ПАКЕТАМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ 7

1.1 Математичний пакет MathCAD 2000 7

1.1.1 Інтерфейс користувача системи MathCAD 2000 7

1.1.2 Палітри математичних знаків і документи MathCAD 9

1.1.3 Виклик вбудованих функцій 13

1.1.4 Елементи графічної візуалізації 13

1.2 Математична система Maple 15

1.2.2 Побудова двовимірного графіка заданої користувачем функції 17

1.2.3 Побудова графіка поверхні 18

1.2.4 Керування мишею 18

1.2.5 Символьні обчислення 19

1.2.6 Основні елементи інтерфейсу 20

1.3 Матрична лабораторія MATLAB 21

1.3.1 Початок роботи із системою MATLAB 21

1.3.2 Файлова система MATLAB 24

1.3.3 Збереження робочої області 25

1.3.4 Ведення щоденника 26

1.3.5 Завантаження робочої області 26

1.3.6 Вхідна мова системи MATLAB 26

1.3.7 Оператори і функції MATLAB 28

1.3.8 Повідомлення про помилки і виправлення останніх 31

1.3.9 Формати чисел 32

1.3.10 Основи роботи з векторами і матрицями 32

1.3.11 Огляд матричних функцій 36

2 ЛАБОРАТОРНИЙ ПРАКТИКУМ 39

2.1 Спектральні характеристики сигналу 39

2.1.1 Періодичні сигнали 39

2.1.2 Практична ширина спектра сигналу 40

2.1.3 Імпульси прямокутної форми 40

2.1.6 Пилкоподібне коливання 44

2.1.7 Неперіодичні сигнали 45

2.1.8 Спектр експоненціального імпульсу 46

2.1.9 Спектр сигналу ввімкнення 47

2.1.10 Спектр дельта-функції 48

2.1.11 Лабораторна робота №1 49

Контрольні запитання 51

2.2 Квантування сигналів 57
2.2.1 Перетворення неперервних сигналів у дискретні 57
2.2.2 Квантування за рівнем 57
2.2.3 Квантування в часі 58
2.2.4 Частотний критерій Котельникова 58
2.2.5 Критерій припустимого відхилення 60
2.2.6 Лабораторна робота №2 62
Контрольні запитання 62
2.3 Дослідження випадкових процесів 68
2.3.1 Лінійні операції над випадковими функціями 68
2.3.2 Стаціонарні випадкові функції 70
2.3.3 Лабораторна робота №3 72
2. 4 Методи обробки зображень 77
2.4.1 Що таке колір? 77
2.4.2 Колірна схема RGB 77
2.4.3 Обробка кольорових (RGB) зображень 79
2.4.4 Лабораторна робота №4 80
2.5 Завадостійке кодування 84
2.5.1 Основні принципи завадостійкого кодування 84
2.5.2 Лабораторна робота №5 89
2.6 Методи стиснення даних 94
2.6.1 Код Шеннона-Фано 94
2.6.2 Лабораторна робота №6 97
Контрольні запитання 97
2.6.3 Код Хаффмена 101
2.6.4 Лабораторна робота №7 103
2.6.5 Арифметичне кодування 106
2.6.2 Алгоритм арифметичного кодування в загальному вигляді 108
2.6.6 Лабораторна робота №8 108
Контрольні запитання. 109
2.7 Динамічні методи стиснення даних 112
2.7.1 Динамічне кодування методом Хаффмена 112
2.7.2 Лабораторна робота №9 115
2.7.3 Динамічне кодування методом FGK 118
2.7.4 Лабораторна робота №10 120
2.7.5 Динамічне кодування методом Віттера 121
2.7.5 Лабораторна робота №11 123
Контрольні запитання 124
ЛІТЕРАТУРА 125

ВСТУП

Високі темпи науково-технічного прогресу різко збільшують об'єми інформації, яка підлягає обробці. Відбувається перехід до інформаційних технологій, тобто застосування комп'ютерів і комп'ютерних математичних систем для науково-технічних розрахунків у виробництві, управлінні, науці, освіті, медицині, торгівлі, банківській справі тощо.

Інтегровані системи комп'ютерної математики дозволяють отримувати не тільки дані, але й нову корисну інформацію на основі глибокого аналізу в результаті проведення обчислень різної складності без використання мов програмування. Велике число подібних розробок і бурхливий розвитком їх свідчить про значну зацікавленість ними у всьому світі

Мета запропонованого Вашій увазі лабораторного практикуму – познайомити читача з основами систем комп'ютерної математики MathCAD 2000, Maple V R5, MATLAB 6.5 R13, які є ідеальними математичними інструментами для користувача, що працює в галузі техніки або природничих наук.

Система MathCAD є такою ж зручною й потужною, як мови програмування, але при цьому такою ж легкою у вивченні як електронні таблиці. Maple є загальнодоступною комп'ютерною системою для математичних розрахунків, яка включає засоби для інтерактивної алгебри, дискретної математики, побудови графіків, числових розрахунків та інших розділів математики. Забезпечує унікальне середовище для швидкого створення математичних програм, використовуючи велику кількість бібліотек вбудованих функцій і операцій.

MATLAB – це інтерактивна система, основним об'єктом якої є масив, для якого не потрібно вказувати розмірність явно. Це дозволяє розв'язувати багато обчислювальних задач, пов'язаних з векторно-матричним формулюванням, і при цьому економити час необхідний для програмування.

В основу лабораторного практикуму, який складається з двох розділів, покладено курси лекцій з дисциплін "Експлуатація комп'ютеризованих систем обробки інформації" і "Електронні інформаційні системи і технології".

Перший розділ містить матеріал про системи комп'ютерної математики MathCAD 2000 та Maple V R5, MATLAB 6.5 R13, розглядає базові поняття, а також можливі варіанти введення даних, роботи з графікою, редагування, інтеграції в Internet.

Другий розділ присвячений лабораторному практикуму, який базується на використанні системи комп'ютерної математики в поєднанні з практичними задачами теорії інформації та кодування. В них стисло висвітлюються теоретичні відомості з таких питань:

- спектральні характеристики сигналів (розглядаються періодичні сигнали, імпульси прямокутної форми, імпульси косинусоїдальної форми, пилкоподібне коливання, неперіодичні сигнали, спектр експоненціального імпульсу, спектр сигналу ввімкнення, спектр дельта-функції);
- квантування сигналів (розглядається процес перетворення неперервних сигналів у дискретні, квантування за рівнем, квантування в часі, частотний критерій Котельникова, критерій допустимого відхилення);
- дослідження випадкових процесів на стаціонарність (розглядаються лінійні операції над випадковими функціями, стаціонарні випадкові функції);
- методи обробки зображень (розглядається кольорова схема RGB, обробка кольорових зображень);
- завадостійке кодування;
- методи стиснення даних (розглядаються коди Шеннона-Фано, Хаффмена, арифметичне кодування, динамічні методи стиснення даних);
- динамічні і частотні характеристики систем автоматичного керування.

Описи завдань за ходом виконання робіт супроводжуються підказками, роз'ясненнями та ілюстраціями

Лабораторний практикум адресований широкому колу читачів, які так чи інакше застосовують у своїй діяльності комп'ютерну техніку, та призначений для студентів спеціальності "Виробництво електронних засобів" і може бути корисним студентам інших спеціальностей, а отримані знання та навички дадуть змогу стати досвідченим знавцем сучасних інформаційних технологій.

1 РОБОТА З ПАКЕТАМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ

Комп`ютерна математика – направлення науки і техніки, яке з`явилося на стику класичної математики та інформатики. Головною основою комп`ютерної математики стали сучасні програмні системи комп`ютерної математики. З їх допомогою користувачі можуть розв`язувати практично любі математичні та прикладні задачі.

Системи комп`ютерної математики представлені розробниками ряду закордонних фірм (MathSoft, MathWorks, Maple, Wolfram та ін.)

В останні роки системи комп`ютерної математики широко використовуються для розрахунку навчальних, наукових та інженерних задач, а також в якості зручних і повних довідників з математичних обчислень. Вони забезпечують наглядне відображення даних та результатів обчислень і також є потужним інструментом підготовки електронних уроків, курсів лекцій та електронних книг з "живими" прикладами.

1.1 Математичний пакет MathCAD 2000

1.1.1 Інтерфейс користувача системи MathCAD 2000

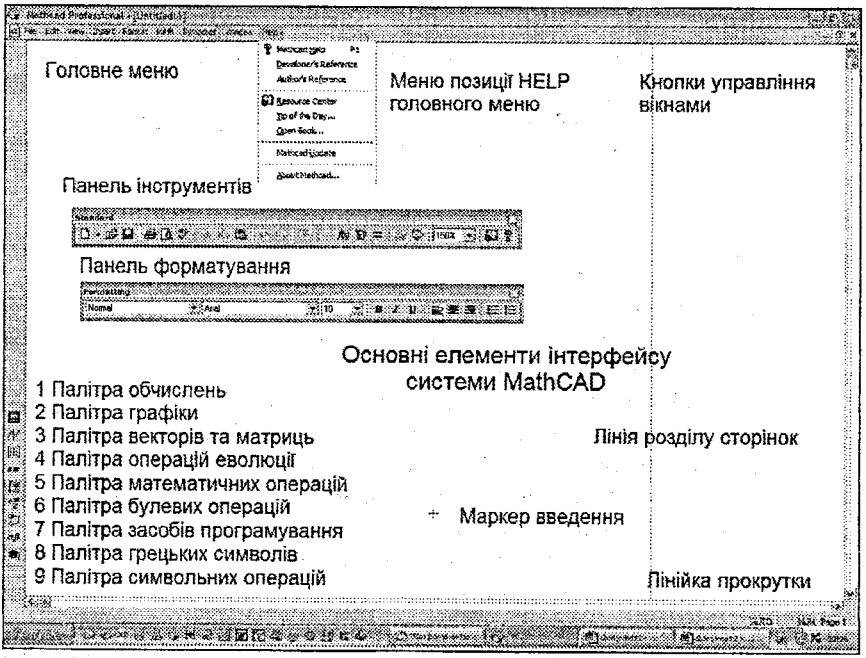


Рисунок 1.1 – Основні елементи інтерфейсу системи MathCAD

У рядку меню системи MathCAD, крім вже описаних позицій, є дві характерних для MathCAD меню: **Math** — керування процесом обчислень та **Graphics** — робота з графікою.

Елементи керування панелі інструментів MathCAD (рис.1.1) перераховані нижче.

- Кнопки операцій з файлами:
  - 1 – **New** – створення нового документа заданого стилю;
  - 2 – **Open** – завантаження раніше створеного документа за допомогою діалогового вікна;
  - 3 – **Save** – запис поточного документа з його ім'ям.
- Друк і контроль документів:
  - 4 – **Print** – друк документа;
  - 5 – **Print Preview** – попередній перегляд документа;
  - 6 – **Check Speling** – перевірка орфографії документа.
- Кнопки операцій редагування:
  - 7 – **Cut** – перенос виділеної частини документа в буфер обміну (Clipboard);
  - 8 – **Copy** – копіювання виділеної частини документа в буфер обміну;
  - 9 – **Paste** – перенос умісту буфера обміну у вікно редагування на місце, зазначене курсором;
  - 10 – **Undo** – скасування попередньої операції редагування;
  - 11 – **Redo** – повторення скасованої операції.
- Кнопки розміщення блоків:
  - 12 – **Align Across** – блоки вирівнюються по горизонталі;
  - 13 – **Align Down** – блоки вирівнюються по вертикалі, розташовуючи зверху вниз.
- Кнопки операцій з виразами:
  - 14 – Insert Function – вставка функції зі списку, що з'являється в діалоговому вікні;
  - 15 – **Insert Utit** – вставка розмірних одиниць;
  - 16 – **Calculate** – обчислення виділеного виразу.
- Доступ до нових можливостей MathCAD 8/2000 PRO:
  - 17 – **Insert Giperlink** – вставка гіперссилки;
  - 18 – **Component Wizard** – відкриває вікно Майстра, що дає зручний доступ до всіх компонентів системи;
  - 19 – **Run MathConnex** – запуск системи MathConnex, призначеної для стимулювання блочно заданих пристроїв.
- 20–21 – Список масштабу, що розкривається, відображення документа **Zoom**.
- Кнопки керування ресурсами:
  - 22 – **Resource Center** – доступ до центра ресурсів;

23 – **Help** – доступ до довідкової системи.

Елементи керування панелі форматування (рис.1.1) перераховані нижче.

- Списки, що розкриваються, для задання параметрів символів:
  - 1-2 – **Style** – вибір стилю для текстових блоків;
  - 3-4 – **Font** – вибір шрифту для текстових блоків;
  - 5-6 – **Font Size** – вибір розміру шрифту для текстових блоків.
- Кнопки зміни креслення символів:
  - 7 – **Bold** – напівжирне креслення; .
  - 8 – **Italic** – курсив;
  - 9 – **Underline** – підкреслення.
- Кнопки завдання режиму вирівнювання тексту:
  - 10 – **Align Left** – вирівнювання текстів по лівій границі;
  - 11 – **Align Center** – вирівнювання текстів по центру;
  - 12 – **Align Right** – вирівнювання текстів по правій границі.
- Кнопки для створення списків:
  - 13 – **Bullets** – створення маркірованого списку;
  - 14 – **Numbering** – створення нумерованого списку.

У нижній частині вікна системи знаходиться рядок стану системи. Якщо система не виконує дій, то в цьому рядку з'явиться напис **"Press F1 for help"** – натисни клавішу **F1** для виклику вікна довідкової системи.

1.1.2 Палітри математичних знаків і документи MathCAD

MathCAD за допомогою палітр спецзнаків (рис.1.2) дозволяє створювати документи, що містять математичні вирази в звичному для математиків вигляді. Вони можуть містити текстові коментарі, числа, таблиці, графіки і малюнки. Будь-яку палітру можна перемістити в зручне місце екрана чи закрити. Документи MathCAD схожі на сторінки математичних книг і наукових статей (рис.1.3). При цьому велику частину математичних виразів утворять оператори і функції, що вводяться за допомогою палітр. Найпоширенішими операторами є оператор присвоювання змінним значень ":=" (швидко вводиться клавішею ":") і оператор виконання "=" (його можна використовувати і як знак першого присвоювання). Навряд чи вимагають пояснення оператори арифметичних операцій.

Для введення текстових блоків досить увести символ "" (одні подвійні лапки – не плутайте з одиночними лапками чи апострофом). У прямокутник, що з'явився, можна почати вводити текст. Він редагується стандартними прийомами. Для англомовних текстів передбачений навіть орфографічний контроль із застосуванням убудованого словника.

Приведемо кілька прикладів простих обчислень:

Введення	На екрані
1.234*2.345=	$1.234 \cdot 2.345 =$
1/7=	$\frac{1}{7} = 0.148$
cos(0.5)=	$\cos(0.5) = 0.878$
e^2=	$e^2 = 7.389$
a=1	$a := 1$
b:1	$b := 1$
a+b=	$a + b = 2$
2*cos(π)=	$2 \cdot \cos(\pi) = -2$

Рисунок 1.2 – Вікно MathCAD з палітрами математичних знаків

Уже з цих простих прикладів (рис.1.3) можна помітити особливості роботи з MathCAD:

- деякі комбіновані оператори (наприклад, ":=") вводяться одним символом;
- оператор "=" використовується як оператор виконання і як оператор першого присвоювання значень змінним;

- MathCAD вставляє пропуски до і після арифметичних операторів;
- оператор множення вводиться як зірочка, але представляється точкою в середині рядка;
- оператор ділення вводиться як похила риска, але заміняється горизонтальною рискою;
- оператор піднесення до степеня вводиться знаком "^", але число степеня представляється в звичайному вигляді (степінь як верхній індекс);
- за замовчуванням десяткові числа мають представлення з трьома знаками після розділової точки;
- наявність у записі числа десяткової точки робить його дійсним (наприклад, 2 – ціле число, а 2. – дійсне);
- MathCAD ідентифікує найбільш розповсюджені константи, наприклад, e – основа натурального логарифма або π.

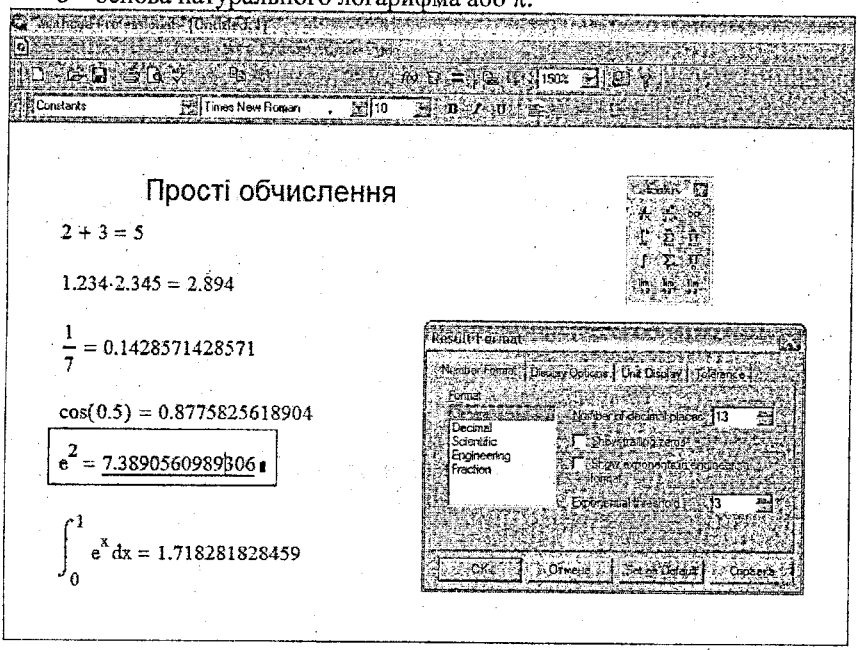


Рисунок 1.3 – Вигляд типового документа MathCAD

Таким чином, навіть без утручання користувача MathCAD намагається додати математичним вираженням звичайний вигляд. На рис.1.3 показане також вікно форматування результатів обчислень. Введіть у поле **Number of decimal places** число 15 замість 3, і одержите результат з 15 знаками після десяткової точки.

Варто пам'ятати, що MathCAD інтерпретує блоки, переглядаючи їх зліва направо і зверху вниз. Це треба враховувати, задаючи розташування блоків і переміщаючи їх у вікні документів. Не можна обчислювати блок, якщо не визначені всі вхідні в нього змінні і функції користувача. Визначення відповідають операції локального присвоювання, що вводиться знаком ":=" (знак = можна використовувати для першого присвоювання, але в основному він служить виконуваним оператором). Є й оператор глобального присвоювання "≡" (наприклад, $x\equiv 23$ чи $f(a,x)\equiv a-\sin(x)$, що дозволяє розміщати визначення змінних і функцій користувача в будь-якому місці документа.

Введення складних операторів (наприклад, обчислення сум, добутків, інтегралів і т.д.) полегшується завдяки виведенню шаблона потрібного оператора з місцями введення – чорними квадратиками. Можна клацнути на потрібному місці введення і ввести дані. На рис.1.4 показане послідовне заповнення місць введення в шаблоні обчислення визначеного інтеграла й у ряді інших шаблонів.

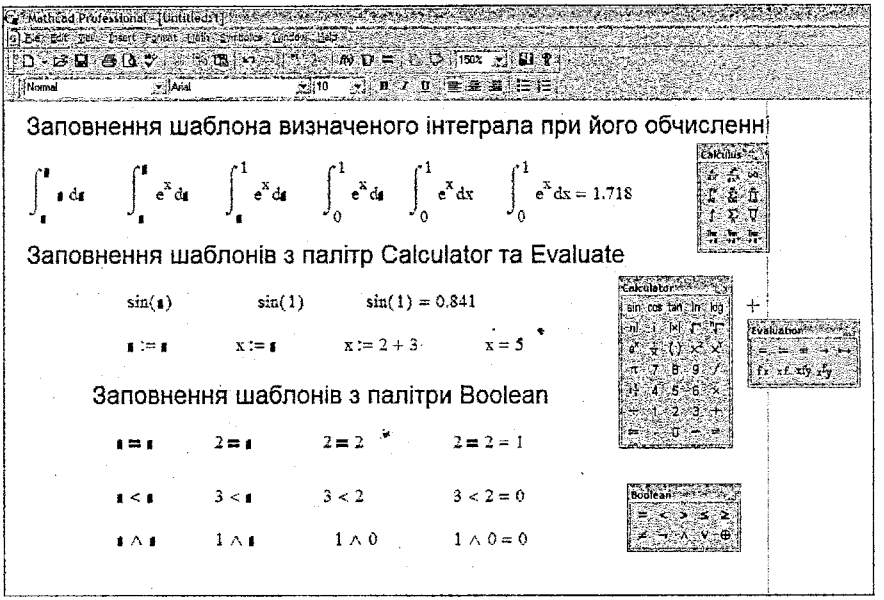


Рисунок 1.4 – Приклади послідовного заповнення шаблонів

Шаблони ряду операцій уводяться з клавіатури символами, що нагадують вигляд оператора хоча б у першому наближенні. Наприклад, оператор квадратного кореня вводиться клавішею "\", шаблон визначеного

інтеграла – клавішею "&", знак добутку ряду – клавішею "#" і т.д. У той же час знак суми членів ряду вводиться комбінацією клавіш **Ctrl+Shift+F4.**

За допомогою миші з натиснутою лівою кнопкою будується прямокутник виділення, що розширюється прямо, з пунктирних ліній. Блоки, що потрапили в нього, виділяються також пунктирною лінією. Вони утворюють блок виділених об'єктів, який можна переміщувати, копіювати в буфер обміну і видаляти.

При заданні складних обчислень робота системи може бути довгою. Для переривання роботи можна натиснути клавішу **Esc**. MathCAD виведе вікно з повідомленням про переривання обчислень і двома кнопками: **OK** – підтвердити переривання і **Cancel** – скасувати переривання. Після переривання можна відновити роботу, натиснувши клавішу **F9** чи натиснувши на кнопці **Calculate** панелі інструментів (кнопка з зображенням жирного знака рівності).

1.1.3 Виклик вбудованих функцій

MathCAD має безліч вбудованих елементарних, спеціальних і статистичних функцій. Для полегшення введення математичних функцій служить кнопка *f(x)*, що виводить вікно з повним переліком функцій, розбитим на тематичні розділи. Шаблон виділеної функції може бути перенесений у вікно документа натиснувши на кнопку **Insert**.

Функції мають параметри (аргументи), що записуються в круглих дужках після імені функції. Функції можуть мати один параметр, наприклад sin(*x*) чи cos(0.5), два параметри, наприклад ln(*m*, *x*), чи навіть кілька параметрів. Параметри можуть мати чисельне значення, бути константою, визначеної раніше змінної або математичним виразом, що повертає чисельне значення.

1.1.4 Елементи графічної візуалізації

MathCAD дозволяє будувати такі типи графіків:

- графіки функцій однієї змінної *f(x)*;
- графіки параметрично заданих функцій у полярній системі координат;
- графіки поверхні для функцій двох змінних *z(x, y)*;
- контурні графіки для функцій двох змінних *z(x, y)*;
- стовпцеві графіки для функцій двох змінних *z(x, y)*;
- точкові графіки поверхні для функцій двох змінних *z(x, y)*;
- графіки векторного поля (у вигляді стрілок).

Побудова графіків у системі MathCAD гранично спрощено. Так, для графіка функції *f(x)* досить увести функцію і потім – шаблон графіка (за допомогою команд меню **Insert** або на панелі інструментів). Після цього в

місце введення на шаблоні графіка треба вставити ім'я змінної *x*. Після клацання кнопкою миши поза шаблоном графік буде побудований.

Для контурних графіків параметрично заданих функцій треба ввести шаблон графіка і задати два вирази – по осях *X* і *Y*. Для графіків функцій двох змінних треба ввести функцію *f(x, y)*, вибрати потрібний шаблон графіка (у палітрі графіків) і в місці введення задати ім'я функції *f*.

Вигляд графіка залежить від його форматування. Команди форматування зосереджені в меню **Format**. Вікна форматування можна також викликати за допомогою контекстних меню, що з'являються при натисканні на праву кнопку миші або при подвійному натисканні на області графіка. Для найбільш складних тривимірних графіків передбачена можливість завдання графіка з усіма потрібними параметрами форматування за допомогою Майстра – команда **3D Plot Wizard** меню **Insert**.

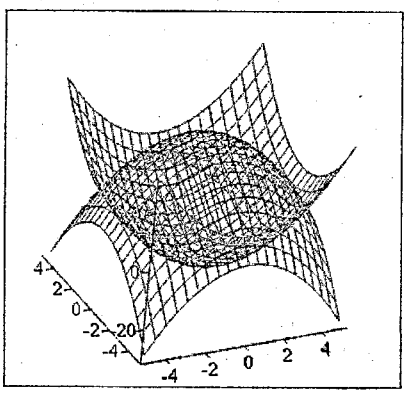
У MathCAD є можливість побудови на одному графіку ряду поверхонь. На рис.1.5 показаний приклад побудови графіка перетинних параболічних поверхонь – одна з них піднята по вертикалі, а інша опущена. Зверніть увагу на запис функцій *z1(x, y)* і *z2(x, y)*.

Побудова графіків двох перетинних поверхонь

$z1(x,y) := x^2 + y^2 - 20$

$z2(x,y) := -(x^2 + y^2) + 20$

Задаємо дві функції двох змінних



z1, z2

Рисунок 1.5 – Побудова двох перетинних поверхонь

Вигляд тривимірного графіка залежить від того, як він розгорнутий щодо точки огляду. Крім завдання кутів огляду з вікна форматування графіків, MathCAD дозволяє обертати графіки мишею. При натиснутій клавіші **Ctrl** можна мишею видаляти або наближати об'єкт до спостерігача. При натиснутій клавіші **Shift** можна мишею ініціювати анімовану картину обертання графіка. Для зупинки обертання треба клацнути лівою кнопкою миші.

1.2 Математична система Maple

Maple – система комп'ютерної алгебри, розрахована на виконання самих складних розрахунків. Її основою є ядро системи, що містить біля тисячі базових функцій. Є також основна бібліотека операторів, команд і функцій та ряд пакетів розширення системи (**packages**). Додаткові функції з них повинні застосовуватися після завантаження пакета за допомогою команди **with(name)**, де **name** – ім'я застосовуваного пакета.

1.2.1 Перше знайомство із системою Maple

Після запуску системи з'являється її основне вікно (рис. 1.6). Спочатку воно порожнє і має ім'я **Untitled**. Як показано на рис. 1.6, можна відразу приступати до виконання обчислень.

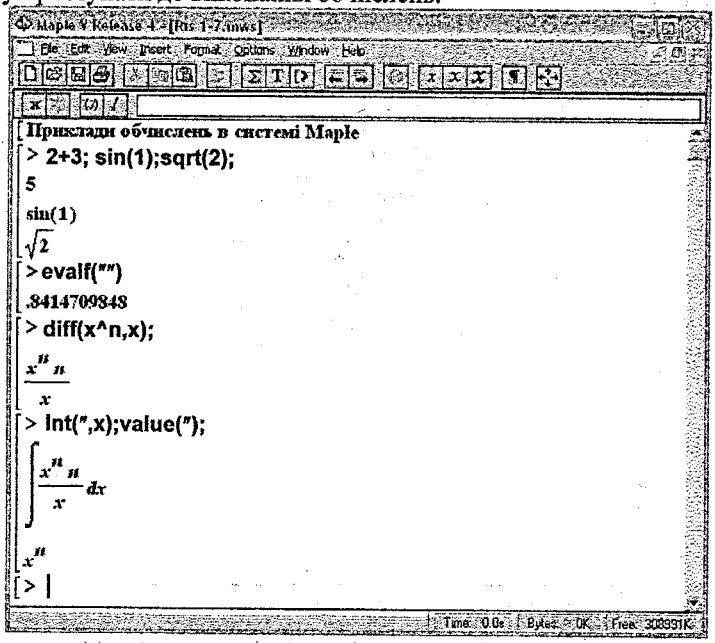


Рисунок 1.6 – Вікно системи Maple із прикладами обчислень

Вхідні вирази і результати їхніх обчислень займають окремі комірки, що позначаються ліворуч квадратними дужками. Довжина їх залежить від розміру виразів — вихідних (питання) і результатів обчислень. Знак ">" є запрошенням до введення математичного виразу. Текстовий рядок (зверху) знака ">" немає. Введення виразу задається за правилами рядкового редагування з застосуванням курсору введення – миготливої вертикальної риски.

Знак фіксації кінця вхідного виразу ";" (крапка з комою) забезпечує виведення результату на екран, а знак ":" (двокрапка) відміняє виведення. Ці знаки також є роздільниками при записі в одному рядку декількох виразів. Клавіші переміщення курсору дозволяють переміщатися по раніше введених у сеансі роботи рядках. Рядки цілком і частково можуть копіюватися в буфер обміну і витягатися з нього.

На рис. 1.6 показане застосування ряду складних вбудованих функцій для символьних обчислень – знаходження похідної **dif(*f*(*x*),*x*)** та невизначеного інтеграла **Int(*f*(*x*),*x*)**. Перша функція (ім'я починається з малої літери) є *обчислюваною.* А від функції, імена яких починаються з великою букви, називаються *інертними.* Вони не обчислюються, а забезпечують близьку до природного форму запису вхідного виразу. Це може знадобитися для візуалізації обчислень.

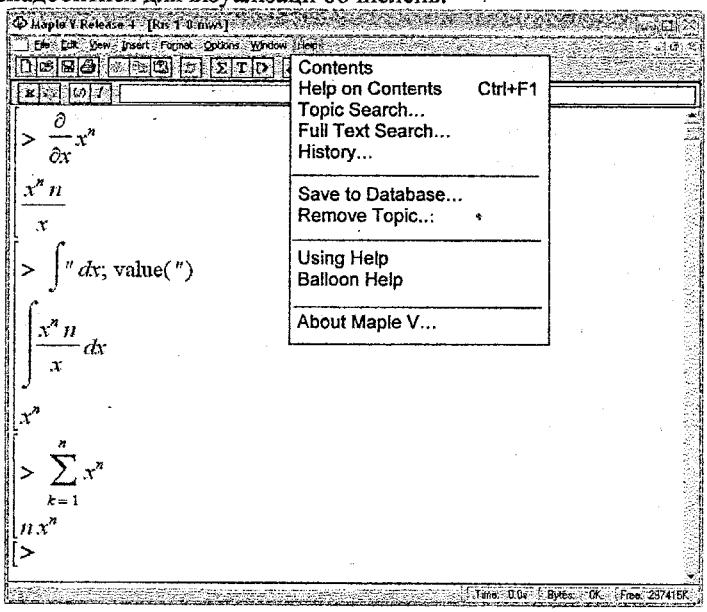


Рисунок 1.7 – Обчислення з природним представленням вхідних виразів

В комірках введення виразу з такими функціями можна перевести до звичайної математичної форми як показано на рис.1.7. Для цього потрібно установити курсор у потрібну комірку і клацнути на кнопці *x* на контекстній панелі (**Context Bar**), що буде описана пізніше. Найпоширенішим оператором є оператор присвоювання ":=". Він використовується для задання змінним конкретних значень, наприклад *a*:=2; *b*:=3; *c*:=*a*+*b*; і т.д.

На рис.1.7 показане також розкрите меню **Help**. З його допомогою можна одержати доступ до великої довідкової бази даних системи.

1.2.2 Побудова двовимірного графіка заданої користувачем функції

На рис. 1.8 представлений приклад побудови графіка функції. Тут графічний об'єкт представлений змінною *g*, значення якої (графік) виводиться в рядок висновку. Для цього рядка показане контекстне меню, що дозволяє форматувати графік. Для побудови графіка математичної функції f використовується графічна функція **plot**(*f*, *x*= –15..15). Параметр *x*= –15..15 задає змінну, щодо якої будується графік, а також область зміни значень цієї змінної (у нашому випадку від –15 до +15).

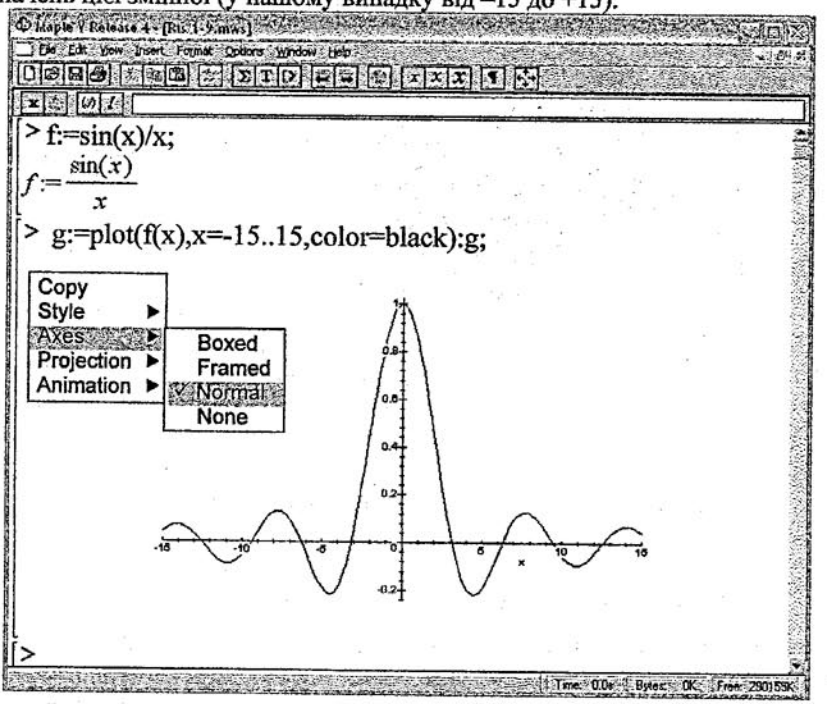


Рисунок 1.8 – Приклад побудови графіка функції

1.2.3 Побудова графіка поверхні

Настільки ж просто можна побудувати графік тривимірної поверхні чи поверхні навіть двох перетинних поверхонь (рис.1.9). Для цього використовується графічна функція **plot 3d.**

На рис. 1.9 показане контекстне меню, що забезпечує доступ до команд форматування. Можна, наприклад, задати функціональне фарбування графіка за заданою схемою висвітлення, установити тип обрамлення графіка і т.д.

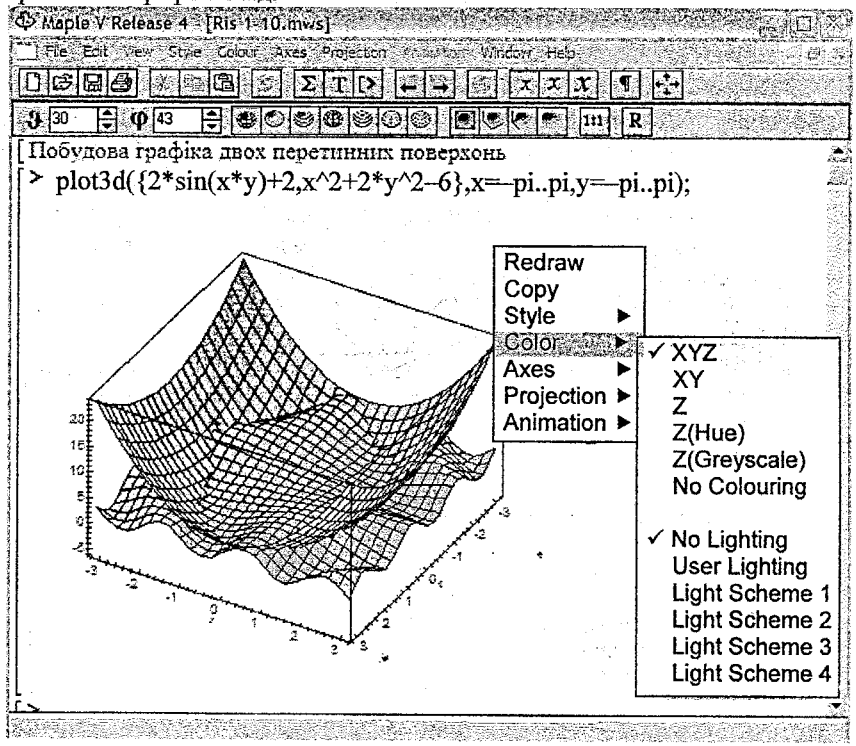


Рисунок 1. 9 – Побудова графіка двох перетинних поверхонь

1.2.4 Керування мишею

Реалізації можливостей керування мишею в Maple приділена значна увага. Крім звичайних функцій миші (клацання кнопкою, керування меню і виділення об'єктів, виразів і їхніх частин) для керування станом комірок можна використовувати контекстне меню (рис. 1.8 і 1.9). Якщо клацання

правою кнопкою миші виконується в комірці введення, контекстне меню має три команди (рис.1.10):

- **Standard Math** – вмикання/вимикання відображення вхідних виразів у природній математичній формі;
- **Maple Input** – керування статусом (текст/уведення) комірки введення;
- **Evalution** – виконання (обробка) комірки.

У меню **Help** є команда **Ballon Help**. Якщо вона активна (поруч з ім'ям команди присутній знак "√"), то при наведенні покажчика миші на елемент інтерфейсу з'являються висхідні підказки (рис. 1.10). Миша може використовуватися для виділення виразів в комірках введення, виділення самих комірок чи обертання графіка мишею.

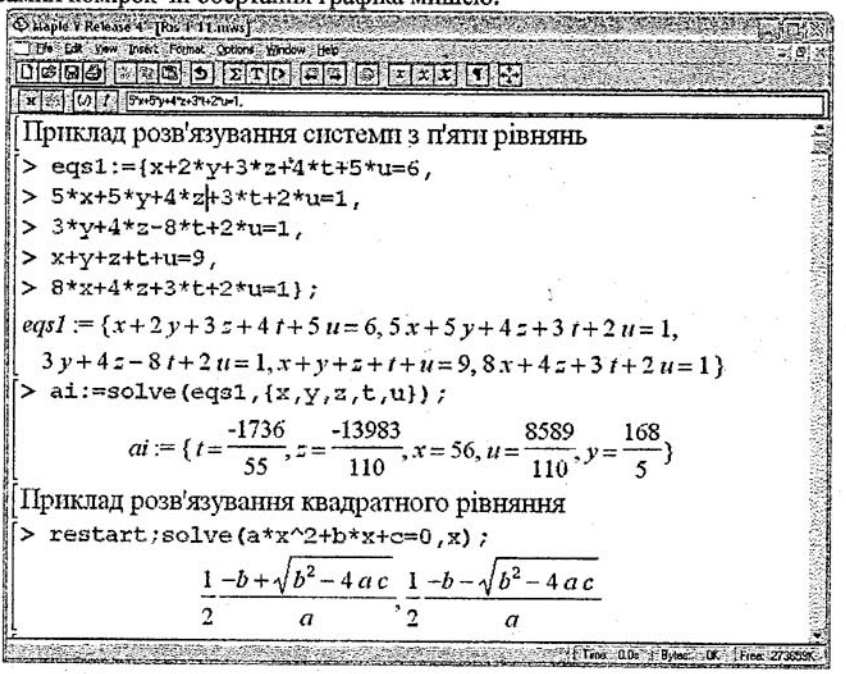


Рисунок 1.10 – Розв'язування системи з п'яти лінійних рівнянь

На рис.1.10 під панеллю інструментів показана вже згадана контекстна панель (**Context Bar**). Вона названа так тому, що склад її кнопок залежить від поточного стану системи, тобто від контексту. Контекстна панель розміщається під панеллю інструментів (якщо вона відображається). При введенні тексту ця панель перетворюється в стандартну панель форматування.

1.2.5 Символьні обчислення

На рис. 1.6 і 1.7 був приведений ряд прикладів на символьні обчислення, наприклад, на інтегрування за допомогою функції **int(*f*(*x*),*x*)**. У даному випадку показана інертна форма цієї функції **Int(*f*(*x*),*x*)** (рис.1.6). При цьому для обчислення виразу з інертною функцією використовується функція **value(%)**.

На рис.1.10 були приведені додаткові приклади – рішення системи лінійних рівнянь і квадратного рівняння за допомогою функції **solve**.

1.2.6 Основні елементи інтерфейсу

Інтерфейс користувача Maple в основному звичайний для систем комп'ютерної математики. Призначення кнопок панелі інструментів показане на рис. 1.11.

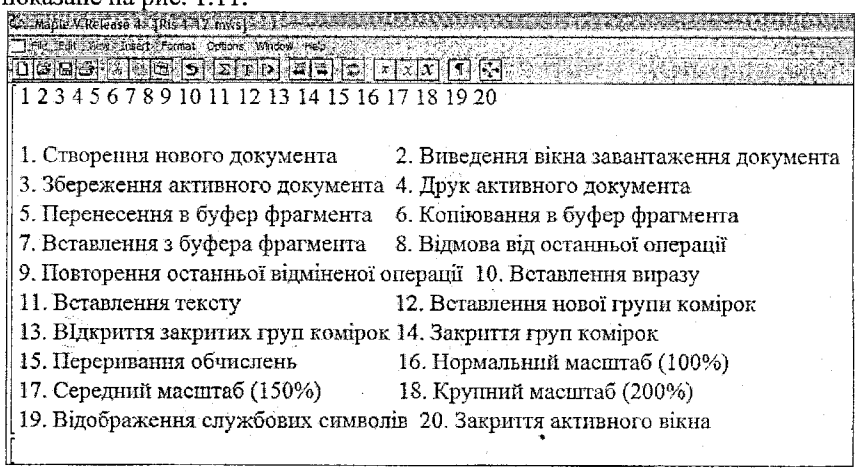


Рисунок 1.11 – Призначення кнопок панелі керування

Командою **Context Bar** (контекстна панель) можна вивести ще один важливий елемент інтерфейсу – контекстну панель. Вона може мати різний вигляд. Наприклад, при введенні тексту вона майже ідентична панелі форматування **Word**, а при введенні математичних виразів частково нагадує панель формул **Excel**.

Інший корисний засіб для полегшення роботи з форматування текстів, вхідних математичних виразів і графіків – контекстні меню, що з'являються при клацанні на об'єкті правою кнопкою миші (рис. 1.8, 1.9). Рядок стану звичайний для **Windows**-додатків. Вона відображає корисну й оперативну інформацію – насамперед про час і використання пам'яті , а

також про характер роботи із системою. Команди меню **View** дозволяють відобразити чи сховати кожну з панелей або рядок стану.

Основним способом введення математичних виразів є їхній набір в комірці введення за допомогою символьного редактора. Однак в меню **View** є команда **Palette** що дозволяє вивести три палітри математичних символів, показані на рис.1.12. Там же показаний порядок роботи із шаблоном визначеного інтеграла.

Палітри математичних символів реалізовані починаючи з Maple V R5. Попередні версії цих палітр не мають.

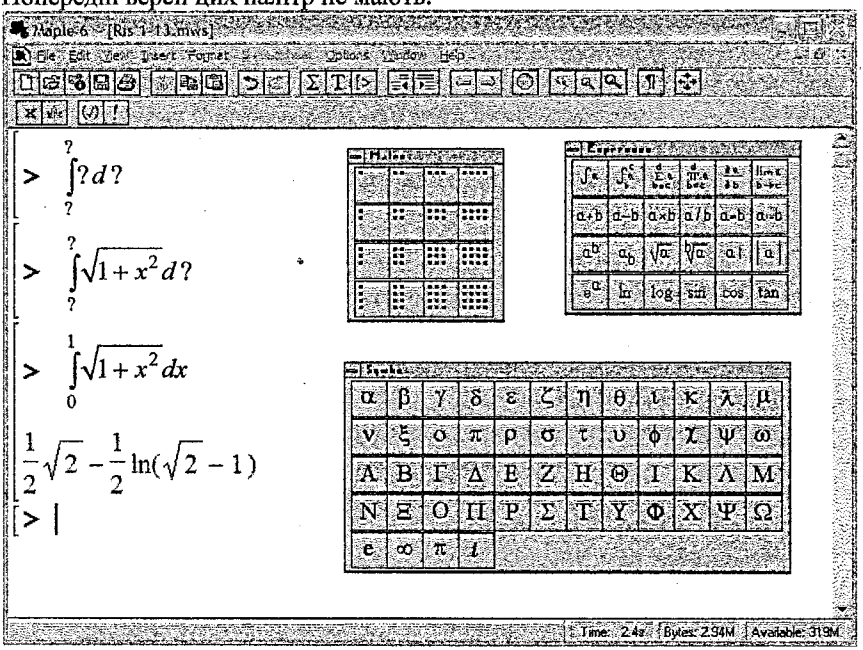


Рисунок 1.12– Робота з палітрами введення математичних символів

1.3 Матрична лабораторія MATLAB

Система комп'ютерної математики MATLAB прославилася великим числом матричних функцій, високою швидкістю чисельних операцій, могутньою мовою програмування, чудовими засобами графічної візуалізації, унікальної за обсягом інформації довідковою системою, пакетом блокового моделювання систем і пристроїв Simulink і ін. Незважаючи на свою громіздкість (MATLAB при повній інсталяції займає більш 1 Гб на жорсткому диску), ця система широко використовується в наукових лабораторіях і у ведучих університетах світу.

1.3.1 Початок роботи із системою MATLAB

Після активізації ярлика MATLAB з'являється вікно системи (рис. 1.13). Система відразу ж готова до проведення обчислень у командному режимі. Головне вікно системи містить рядок меню і панель інструментів. Рядок меню MATLAB містить п'ять звичайних для систем комп'ютерної математики позицій: `File`, `Edit`, `View`, `Window`, `Help`. Приведемо призначення кнопок панелі інструментів :

- `New file` – виведення порожнього вікна редактора;
- `Open file` – відкриття вікна завантаження файлу;
- `Cut` – перенесення виділеного фрагмента в буфер обміну;
- `Copy` – копіювання виділеного фрагмента в буфер обміну;
- `Paste` – вставлення фрагмента з буфера обміну в даний рядок уведення;
- `Undo` – скасування попередньої операції;
- `Workspace Browser` – виведення вікна перегляду ресурсів робочої області;
- `Path Browser` – виведення вікна перегляду файлової структури;
- `New Simulink Model` – відкриття вікна додатка Simulink;
- `Help Windows` – відкриття вікна довідкової системи.

Навіть з декількох простих прикладів, показаних на рис. 1.13, можна зробити висновки:

• для вказівки місця уведення вихідних даних використовується символ "»";

• для блокування виведення результату обчислень деякого виразу наприкінці виразу треба увести знак "`;`" (крапка з комою);

• якщо не зазначена змінна зі значенням результату обчислень, MATLAB призначає таку змінну, давши їй ім'я `ans`;

• знаком присвоювання є звичний математикам знак рівності "=", а не комбінований знак ":=", як у багатьох інших математичних системах;

• вбудовані функції (наприклад, `sin`) записуються малими літерами, і їхні аргументи вказуються в круглих дужках;

• результат обчислень виводиться в рядках виведення (без знаку "»");

• діалог відбувається в стилі «поставив запитання – одержав відповідь».

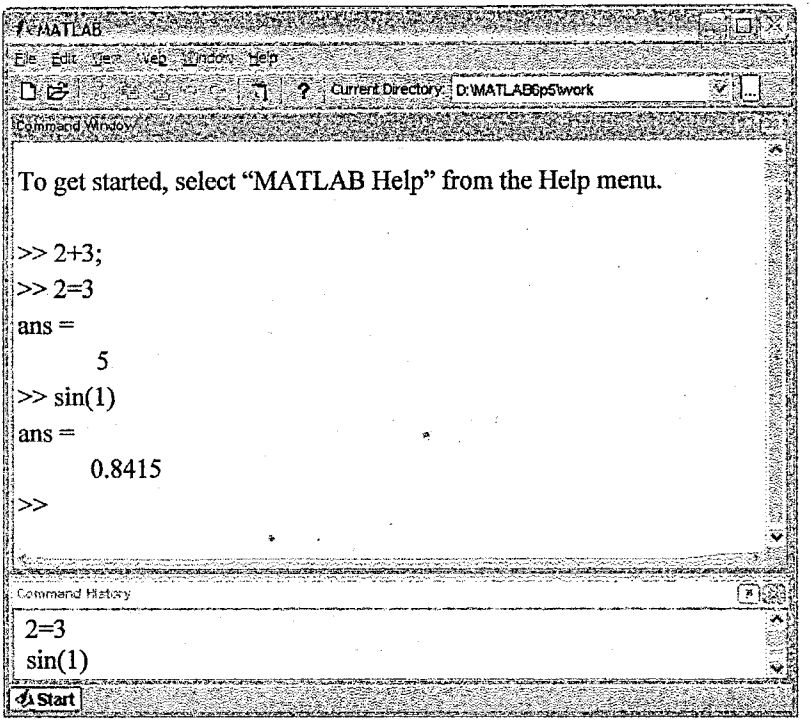


Рисунок 1. 13 – Вікно системи MATLAB після завантаження та виконання простих обчислень

Даний приклад ілюструє виконання простих векторних операцій:

```
» V= [1 2 3 4]
V=
      1   2   3   4
» sin(V)
ans
    0.8415    0.9093    0.1411   -0.7568
» exp (V)
ans
    2.7183    7.3891   20.0855   54.5982
```

Цей раз (і надалі) використаний запис копій екрана, отриманих у сеансі роботи. При цьому пробіли між рядками опускаються. У наведеному прикладі задається вектор `V` зі значеннями елементів `1, 2, 3` і `4`. А далі обчислюються функції синуса й експоненти з аргументом у вигляді вектора, а не скаляра. Вектори задаються списком своїх елементів, які поділені пробілами чи комами. Список укладається в квадратні дужки. Для виділення `n-го` елементу вектора `V` використовується вираз `V(n)`. Він задає відповідну *індексовану змінну*. Будучи матричною системою, MATLAB

будь-яке число сприймає як матрицю розмірності 1×1, а вектор з довжиною n як матрицю розмірності n×1.

Корисно відразу засвоїти деякі команди управління вікном у режимі командного рядка:

clc – очищає екран і розміщує курсор у лівому верхньому куті порожнього екрана;

home – повертає курсор у лівий верхній кут вікна;

echo ім'я_файлу on – вмикає режим виведення на екран тексту файлу сценарію (скрипта);

echo ім'я_файлу off – вимикає режим виведення на екран тексту файлу сценарію;

echo ім'я_файлу – змінює режим виведення файлу сценарію на протилежний;

echo on all - вмикає режим виведення на екран тексту всіх m-файлів;

wcho off all – вимикає режим виведення на екран тексту всіх m-файлів;

more on – вмикає режим посторінкового виведення (корисний при перегляді великих m-файлів);

more off – вимикає режим посторінкового виведення (у цьому випадку для перегляду великих файлів треба користуватися смугою прокручування).

Для завершення роботи із системою можна використовувати команди quit, exit або комбінацію клавіш Ctrl+Z. Якщо необхідно зберегти значення всіх змінних (векторів, матриць) системи, то перед цим варто ввести команду save потрібної форми. Команда load після завантаження системи зчитує значення цих змінних і дозволяє почати роботу із системою з того моменту, коли вона була перервана

1.3.2 Файлова система MATLAB

Система MATLAB складається з безлічі файлів, що знаходяться в різних папках. Найбільш важливі файли двох типів – з розширеннями .mat і .m. Перші є бінарними файлами, що представляють запис параметрів робочої області в сеансі роботи системи. Другі є текстовими файлами, що містять зовнішні визначення команд і функцій системи. Особливе значення має папка MATLAB/TOOLBOX/MATLAB. У ній міститься набір m-файлів стандартного розширення системи, які називаються Toolbox. Перегляд цих файлів дозволяє детально оцінити можливості конкретної версії системи. Нижче перераховані основні папки набору Toolbox.

• Команди загального призначення:

General – команди загального призначення.

• Оператори, конструкції мови і системні функції:

ops – оператори і спеціальні символи;
lang – конструкції мови програмування;
strfun – строкові функції;
iofun – функції введення/ виведення;
timefun – функції часу і дат;
datatypes – типи і структури даних.

• Основні математичні і матричні функції:
elmat – команди створення елементарних матриць і операцій з ними;
elfun – елементарні математичні функції;
specfun – спеціальні математичні функції;
matfun – матричні функції лінійної алгебри;
datafun – аналіз даних і перетворення Фур'є;
polyfun – поліноміальні функції і функції інтерполяції;
funfun – функції функцій і функції вирішення ОДУ;
soarfun – функції розріджених матриць.

• Команди графіки:
graph2d – команди двовимірної графіки;
graph3d – команди тривимірної графіки;
specgraph – команди спеціальної графіки;
graphics – команди графіки Handle Graphics;
uitoob – графіка інтерфейсу користувача.

Повний список файлів будь-якої вкладеної папки можна вивести для перегляду за допомогою команди `help ім'я`, де `ім'я` – назва відповідної вкладеної папки (див. вище).

1.3.3 Збереження робочої області

Змінні і визначення нових функцій у системі MATLAB зберігаються в визначеній області пам'яті іменованою робочою областю. MATLAB дозволяє зберегти значення всіх змінних і визначень поточного сеансу роботи, тобто робочої області, у вигляді бінарних файлів з розширенням `.mat`. Для цього служить команда `save`, що може мати кілька форм:

• `save fname` – зберігається робоча область зі значеннями всіх змінних у файлі бінарного формату з ім'ям `fname` і розширенням `.mat`;

• `save fname X` – зберігається тільки значення змінної `X`;

• `save fname X Y Z` – зберігаються значення змінних `X`, `Y` і `Z`.

Після параметрів команди `save` можна вказати ключі, що уточнюють формат запису файлів:

• `-mat` – двійковий формат, що використовується за замовчуванням;

• `-ascii` – формат ASCII одиничної точності (8 цифр);

• `-ascii -double` – формат ASCII подвійної точності (16 цифр);

• -ascii -double -tabs – формат з роздільником і мітками табуляції;

• V4 – запис mat-файлу в стандарті версії MATLAB;

• -append – додавання в існуючий mat-файл.

Можливе завдання необхідності збереження у форматі функції, наприклад:

```
Save('fname','var1','var2')
```

У цьому випадку імена файлів і змінних задаються рядковими константами.

1.3.4 Ведення щоденника

Для запису всього сеансу чи роботи його фрагментів служить команда diari:

• diary ім'я_файлу – запис на диск текстового файлу зі зазначеним ім'ям усіх команд у рядках введення й отриманих результатів;

• diary off – припиняє запис у файл;

• diary on – знову починає запис у файл.

Таким чином, чергуючи команди diary off і diary on, можна зберегти потрібні фрагменти сеансу в їхньому формальному вигляді. Команду diary можна задати й у вигляді функції diary('file'), де рядок 'file' визначає ім'я файлу. Рекомендується записувати файл сеансу з розширенням іншим, ніж .m.

1.3.5 Завантаження робочої області

Для завантаження робочої області попереднього сеансу роботи можна використовувати команду load fname зі вказівкою специфікацій. Вона забезпечує завантаження раніше збережених у mat-файлі визначень зі специфікаціями. Допускається також команда load('fname',...) – завантаження файлу fname у формі функції. Якщо команда (чи функція) load вводиться в ході сеансу роботи, відбудеться заміна значень поточних змінних тими значеннями, що були збережені в mat-файлі, що зчитується. Правила завдання імен файлів самі звичайні.

1.3.6 Вхідна мова системи MATLAB

У MATLAB математичні вирази записуються порядково, як це прийнято в більшості мов програмування:

```
2+3
2.301*sin(x)
4+exp(3)/5
sqrt(y)/2
sin(pi/2)
```

Математичні вирази складаються з чисел, констант, змінних, операторів і функцій, а також різних спецзнаків. Нижче представлені приклади задання чисел:

```
0         2             -3            2.301
0.00001   123.456e-24 -234.456e10  3i
2j        2+3i          -3.1411       -123.456+2.7e-3
```

Функція `real(z)` повертає дійсну частину комплексного числа `z`, а функція `imag(z)` – уявну. Для одержання модуля комплексного числа використовується функція `abs(z)`, а для обчислення фази – `angle(z)`.

Операції над числами виконуються у форматі з *подвійною точністю*. Такий формат задовольняє переважній більшості вимог до чисельних розрахунків. Символьні обчислення MATLAB може виконувати за допомогою спеціального пакета розширення Symbolic. Для переносу довгих рядків використовується знак три точки, наприклад:

```
s=1-1/2+1/3-1/4+1/5-1/6+1/7...
1/8+1/9-1/10+1/11-1/12;
```

Цей прийом може бути дуже корисним для забезпечення наочності документів (щоб рядки не виявлялися поза видимою областю вікна). Максимальне число символів в одному рядку може бути 4096, але з настільки довгими рядками працювати незручно.

У MATLAB використовуються такі системні змінні:

- `i` чи `j` – уявна одиниця (корінь квадратний з –1);
- `pi` – число «пі» 3,1415926...;
- `eps` – похибка для операцій над числами з плаваючою точкою (2^{-52});
- `realmin` – найменше число з плаваючою точкою (2^{-1022});
- `realmax` – найбільше число з плаваючою точкою (2^{1023});
- `inf` – значення машинної нескінченності;
- `ans` – змінна, що зберігає результат останньої операції і його відображення на екрані дисплея;
- `Na` – вказівку на нечисловий характер даних (Not a Number).

Системні змінні можуть *перевизначатися*. Так, можна задати системній змінній `eps` інше значення, наприклад `eps=0.0001`. Важливо те, що значення системних змінних за замовчуванням задаються відразу після завантаження системи. Тому системні змінні ніколи не можуть бути невизначеними, на відміну від звичайних змінних.

Символьна константа – це ланцюжок символів, укладених у символи одиночних лапок, наприклад:

```
'Hello, my friend!'
'2+3'
```

Якщо математичний вираз поміщений в одиночні лапки, то він не обчислюється, а розглядається просто як ланцюжок символів. Таким чином, вираз '2+3' не буде давати число 5. Однак за допомогою спеціальних функцій перетворення символьні вирази можуть бути перетворені.

Текстові коментарі вводяться за допомогою оператора %, наприклад:

```
%Нижче представлене задання функції обчислення факторіала
```

Звичайно у означеннях m-файлів перші рядки означень служать для їхнього опису, що виводяться на екран дисплея після команди

```
» help Ім'я_файлу
```

Вважається правилом гарного тону вводити досить детальні текстові коментарі, зокрема, у m-файли.

У системі MATLAB можна задавати змінним визначені значення. Для цього використовується операція присвоювання, що вводиться оператором рівності "=" (наприклад, x=123). Типи змінних заздалегідь не з'являються, а визначаються виразом, значення якого привласнюється змінній. Ім'я змінної (її ідентифікатор) може містити скільки завгодно символів, але зберігаються й ідентифікуються тільки перші 31 символ.

У пам'яті змінні займають визначене місце, яке є робочою областю (workspace). Засіб для її перегляду вже описувався. Для очищення робочої області використовується функція clear у різних формах, наприклад:

- clear – видалення означень усіх змінних;
- clear x – видалення означення змінної x;
- clear a, b, z – видалення означень змінних списку і т.д.

Для дефрагментації всієї робочої області пам'яті служить команда pack. Віддалена змінна стає невизначеною.

1.3.7 Оператори і функції MATLAB

Специфікою MATLAB є те, що більшість операторів і функцій відноситься до матричних операцій. Це може служити причиною серйозних непорозумінь. Наприклад, оператори множення "*" і ділення "/" обчислюють добуток і частку від ділення двох масивів, векторів чи матриць. Є ряд спеціальних операторів, наприклад, оператор "\" означає ділення справа наліво, а оператори ".*" і "./" означають множення і ділення окремих членів масивів. Повний список операторів можна одержати, використовуючи команду » help ops.

Приведемо арифметичні оператори:

```
» help ops
Operators and special characters.
Arithmetic operators.
     plus - Plus
     uplus - Unary plus
     minus - Minus
```

```
uminus - Unary minus
mtimes - Matrix multiply
times - Array multiply
mpower - Matrix power
power - Array power
```

Функції в загальному випадку мають список аргументів (параметрів), поміщених у круглі дужки. Наприклад, функція Бесселя записується як `bessel(NU, X)`. У даному випадку список параметрів містить два аргументи – `NU` у вигляді числа і `X` у вигляді вектора. Багато функцій допускають ряд форм запису, наприклад, що розрізняються списком своїх параметрів. Якщо функція повертає кілька значень, то вона записується у вигляді

`[Y1, Y2, ...]=func(X1, X2, ...)`,

тут `Y1, Y2, ...` – список вихідних параметрів і `X1, X2, ...` – список вхідних параметрів.

Зі списком елементарних функцій можна ознайомитися, виконавши команду `help elfun`, а зі списком спеціальних функцій – команду `help specfun`.

Функції можуть бути *вбудованими* (внутрішніми) і *зовнішніми (М-функціями)*. Так, вбудованими є найбільш розповсюджені елементарні функції, наприклад `sin(x)` і `exp(y)`, тоді як функція `sinh(x)` є зовнішньою. MATLAB підтримує також безліч спеціальних математичних функцій, використовувати які так само просто, як і елементарні.

Дуже часто необхідно зробити формування впорядкованих числових послідовностей. Такі послідовності потрібні для створення векторів чи значень абсциси при побудові графіків. Для цього в MATLAB використовується оператор "`:`" (двокрапка):

`Початкове_значення:Кроки:Кінцеве_значення`

Дана конструкція породжує послідовність чисел, що починається з початкового значення, йде з заданим кроком і завершується кінцевим значенням. При цьому діють такі правила:

`Початкове_значення<Кінцеве_значення, якщо крок > 0;`

`Початкове_значення>Кінцеве_значення, якщо крок < 0.`

Якщо крок не заданий, то він приймається рівним 1 чи –1 у зазначених співвідношеннях. Приклади застосування оператора : дані нижче:

```
» 1:5
ans =
     1   2   3   4   5
» 1=0:2:10
i =
     0   2   4   6   8   10
» j=10:-2:2
j =
```

```
     10  8    6    4    2
» V=0:pi/2:2*pi;
» V
V =
     0   1.5708   3.1416   4.7124   6.2832
» X=1:-.2:0
X =
     1.0000   0.8000   0.6000   0.4000   0.2000   0
```

Як відзначалося, належність MATLAB до матричних систем вносить корективи в визначення операторів і приводить при некоректному їх використанні до казусів. Розглянемо такий приклад:

```
» x=0:5
x =
     0   1    2    3    4    5
» cos(x)
ans =
     1,0000 0.5403 -0.4161 -0.9900 -0.6536 0.2837
» sin(x)/x
ans =
     -0.0862
```

Обчислення масиву косинусів тут пройшло коректно. А от обчислення масиву функції `sin(x)/x` дає «несподіваний» ефект – замість масиву із шістьма елементами обчислене єдине значення. Причина «парадоксу» тут у тім, що оператор "/" обчислює відношення двох матриць, векторів чи масивів. Якщо вони однієї розмірності, то результат буде одним числом, що в даному випадку і видала система. Щоб, дійсно, одержати масив значень `sin(x)/x`, треба використовувати спеціальний оператор ділення кожного члена масивів – `./`. Тоді буде отриманий масив чисел:

```
» sin(x)/x
Warning: Divide by zero.
ans =
     NaN 0.8415 0.4546 0.0470 -0.1892 -0.1918
```

Утім, і отут без особливостей не обійшлося. Так, при `x=0` значення `sin(x)/x` дає переборну невизначеність вигляду `0/0=1`. Однак, як і всяка чисельна система, MATLAB класифікує спробу розподілу на 0 як помилку і виводить відповідне попередження. А замість очікуваного числового значення виводиться символьна константа `NaN`, що означає, що невизначеність `0/0` – це все-таки незвичайне число.

Вирази з оператором ":" можуть використовуватися як аргументи функцій для одержання множинних їхніх значень. Наприклад, у прикладі, що нижче приводиться, обчислені функції Бесселя порядку від 0 до 5 зі значенням аргумента `x=0.5`:

```
x=1/2:
```

```
» bessel(0:1:5,1/2)
ans =
    0.9385 0.2423 0.0306 0.0026 0.0002 0.0000
```

А в наступному прикладі обчислено шість значень функції Бесселя нульового порядку для значень аргумента від 0 до 5 із кроком 1:

```
» bessel(0,0:1:5)
ans =
   1.0000 0.7652 0.2239 -0.2601 -0.3971 -0.1776
```

Таким чином, оператор : є дуже зручним засобом задання регулярної послідовності чисел. Він широко використовується при побудові графіків.

1.3.8 Повідомлення про помилки і виправлення останніх

Важливе значення при діалозі із системою MATLAB має діагностика помилок. Приклад виведення попередження про помилку (ділення на 0) тільки що приводився. Розглянемо ще ряд прикладів. Введемо, наприклад, помилковий вираз:

```
» sqr(2)
```

Тепер натиснемо клавішу Enter. Система повідомить про помилку:

```
??? Undefined function or variable 'sqr'.
```

Це повідомлення говорить, що не визначена змінна чи функція, і вказує яка саме – `sqr`. У даному випадку можна просто набрати правильний вираз. Однак у випадку громіздкого виразу краще скористатися редактором. Для цього досить натиснути клавішу переміщення рядків "↓". У результаті з'явиться вираз

```
» sqr(2).
```

Тепер за допомогою клавіші → варто встановити курсор після букви `r` і натиснути клавішу `t` на нижньому регістрі, а потім – клавішу Enter. Вираз прийме вигляд

```
» sqrt(2)
ans =
        1.4142
```

Тепер обчислення дають очікуваний результат – значення квадратного кореня з двох.

У системі MATLAB зовнішні означення використовуються точно так само, як і вбудовані функції й оператори. Ніяких вказівок на їхнє застосування робити не треба. Досить лише подбати про те, щоб використовувані означення дійсно існували у вигляді файлів з розширенням `.m`. Але, якщо ви забули про це чи ввели ім'я неіснуючого означення, то система відреагує на це звуковим сигналом і виведенням повідомлення про помилку:

```
» hsin(1)
    ??? Undefined function or variable 'hsin'.
```

```
» sinh(1)
ans =
          1.1752
```

Іноді, що уже відзначалося, у ході виведення даних обчислень з'являється скорочення NaN (від слів Not a Number – не число). Воно означає операцію невизначеності, наприклад вигляду 0/0 чи Inf/Inf. Можуть з'являтися і різні попередження про помилки (англійською мовою). Наприклад, при діленні на 0 кінцевого числа з'являється попередження Warning: Dievide by Zero (Увага: ділення на 0). Весь діапазон представлення чисел у системі лежить від 10^{-308} до 10^{+308}.

У MATLAB треба відрізняти попередження про помилку від повідомлення про неї. Попередження (звичайно після слова Warning) не зупиняють обчислення і лише попереджають користувача про те, що помилка здатна вплинути на хід обчислень. Повідомлення про помилку (після знаків ???) зупиняє обчислення.

1.3.9 Формати чисел

За замовчуванням MATLAB видає числові результати в нормалізованій формі з чотирма цифрами після десяткової точки й однієї цифри до неї. Для зміни формату використовується команда format:

```
x=[4/3 1.2345e-6]
format short 1.3333                  0.0000
format short e 1.3333E+000           1.2345E-006
format long 1.333333333333338        0.000001234500000
format long e 1.333333333333338E+000
                                1.234500000000000E-006
format bank 1.33                     0.00
```

Задання формату позначається тільки на формі виведення чисел. Обчислення все рівно відбуваються у форматі подвійної точності, а введення чисел можливе в будь-якому зручному для користувача вигляді.

1.3.10 Основи роботи з векторами і матрицями

Ми вже розглянули задання векторів у квадратних дужках. Задання матриці вимагає вказівки різних рядків. Для поділу рядків використовується знак “;” (крапка з комою). Цей же знак (чи знак коми) наприкінці введення запобігає виведенню матриці чи вектора на екран дисплея. Розглянемо таке введення:

```
» M=[1 2 3; 4 5 6; 7 8 9];
```

Цей вираз задає квадратну матрицю, яку можна вивести:

```
» M
M =
    1    2    3
    4    5    6
```

```
    7     8     9
```

Можливе введення елементів матриць і векторів у вигляді арифметичних виразів, що містять будь-які доступні системі функції, наприклад:

```
V= [2+2/3+4) exp(5) sqrt(10)];
» V
V =
    2.2857    148.4132   3.1623
```

Для вказівки окремого елементу чи вектору матриці використовуються вирази вигляду `V(i)` чи `M(i; j)`. Наприклад:

```
» M(2;2)
ans =
          5
```

Якщо потрібно привласнити елементу `M(i; j)` нове значення x, варто використовувати вираз `M(i; j)=x`. Наприклад, якщо елементу `M(2; 2)` треба, привласнити значення 10, варто записати таке:

```
» M(2;2)=10
```

Вираз `M(i)` з одним індексом дає доступ до елементів стовпців матриці. Така матриця утвориться з вихідної, якщо підряд виписати її стовпці. Наступний приклад пояснює такий доступ до елементів матриці `M`:

```
» M=[1 2 3; 4 5 6; 7 8 9]
M =
          1    2    3
          4    5    6
          7    8    9
» M(2)
ans =
          4
» M(8)
ans =
          6
» M(9)
ans =
          9
» M(5)=100;
» M
M =
          1    2    3
          4    100  6
          7    8    9
```

Можливе задання векторів і матриць з комплексними елементами, наприклад:

```
» i=sqrt(-1);
» CM =[1 2;3 4]+ i*[5 6; 7 8]
```

Ще приклад:

```
» CM = [1+5*i 2+6*i; 3+7*i 4+8*i]
```

Останній вираз утворює таку матрицю:

```
1+5*i       2+6*i
3+7*i       4+8*i
```

Поряд з операціями над окремими елементами матриць і векторів система дозволяє робити операції множення, ділення і піднесення до степеня відразу над всіма елементами – масивами. Для цього перед операцією ставиться знак крапки. Наприклад, знак "*" означає знак множення для векторів чи матриць, а знак ".*" – множення всіх елементів у вигляді масиву. Так, якщо `M` – матриця, то `M.*2` дасть матрицю, всі елементи якої помножені на скаляр – число 2. Утім, для множення матриці на скаляр два вирази `M*2` і `M.*2` виявляються рівноцінними.

Є також ряд особливих функцій для задання векторів і матриць. Наприклад, функція `magic(n)` задає магічну матрицю розмірністю n×n, у якої сума всіх стовпців, усіх рядків і навіть діагоналей дорівнює тому самому числу:

```
» M=magic(4)
M =
      16   2    3    13
       5   11   10    8
       9   7    6    12
       4   14   15    1
» sum(M)
ans =
      34   34   34   34
»> sum(M')
ans =
      34   34   34   34
» sum(diag(M))
ans =
      34
» M(1,2)+M(2,2)+M(3,2)+M(4,2)
ans =
      34
```

Уже сама по собі можливість утворення такої матриці за допомогою простої функції `magic` зацікавить аматорів математики. Але векторних і матричних функцій у системі безліч, і ми їх детально розглянемо надалі. Нагадаємо, що для видалення векторів і матриць з робочої області служить команда `clear`.

Описаний спосіб задання матриць дозволяє виконати операцію конкатенації – об'єднання малих матриць у велику. Наприклад, створимо спочатку магічну матрицю розмірністю 3x3:

```
» A=magic(3)
```

```
A=
    8    1    6
    3    5    7
    4    9    2
```

Тепер можна побудувати матрицю, що містить чотири матриці:

```
» У=[A A+16;A+32 A+16]
B=
    8    1    6   24   17   22
    3    5    7   19   21   23
    4    9    2   20   25   18
   40   33   38   24   17   22
   35   37   39   19   21   23
   36   41   34   20   25   18
```

Отримана матриця має вже розмірністю 6x6. Обчислимо суму її стовпців:

```
» sum(B)
ans =
     126   126   126   126   126   126
```

Цікаво, що вона однакова для всіх стовпців. А для обчислення суми рядків використовуємо команду

```
» sum(B')
ans =
      78    78    78   174   174   174
```

Тут запис `B'` означає транспонування матриці B, тобто заміну рядків стовпцями. Цього разу сума виявилася різною. Це відкидає попередньо існуюче припущення, що матриця B є магічною. Для істинно магічної матриці суми стовпців і рядків повинні бути однаковими:

```
» D=magic(6)
D=
    35    1    6   26   19   24
     3   32    7   21   23   25
    31    9    2   22   27   20
     8   28   33   17   10   15
    30    5   34   12   14   16
     4   36   29   13   18   11
» sum(D)
ans=
    111   111   111   111   111   111
» sum(D')
ans=
    111   111   111   111   111   111
```

Більш того, однаковою є і сума елементів по основних діагоналях.

Для формування матриць і виконання ряду матричних операцій виникає необхідність видалення окремих стовпців і рядків матриці. Для цьо-

го використовуються порожні квадратні дужки []. Проробимо це над матрицею M:

```
» M=[1 2 3; 4 5 6; 7 8 9]
M=
     1    2    3
     4    5    6
     7    8    9
```

Видалимо другий стовпець:

```
» M(:,2)=[ ]
M =
    1    3
    4    6
    7    9
```

А тепер видалимо другий рядок:

```
» M(2, :)=[ ]
M =
    1    3
    7    9
```

1.3.11 Огляд матричних функцій

MATLAB має найбільше число матричних функцій. Так, для створення векторів і матриць використовуються функції:

- `eye(n)` – одинична матриця розмірністю n×n;
- `ones(n)` – матриця з одиничними елементами;
- `zeros(n)` – матриця з нульовими елементами;
- `rand(n)` – матриця з випадковими елементами;
- `hadamar(n)` – матриця Адамара;
- `hilb(n)` – матриця Гільберта;
- `magic(N)` – магічна матриця;
- `pascal(n)` – матриця Паскаля;
- `toeplitz(n)` – матриця Теплиця;
- `wilkinson(n)` – матриця Уілкінсона й ін.

Для одиночної квадратної матриці A існує ряд функцій:

- `diag(A)` – головна діагональ матриці;
- `fliplr(A)` – матриця з переставленими щодо вертикальної осі елементами;
- `flipud(A)` – матриця з переставленими щодо горизонтальної осі елементами;
- `prod(A)` – вектор добутків елементів кожного стовпця;
- `sum(A)` – вектор сум елементів кожного стовпця;
- `rot90(A)` – поворот матриці на 90°;
- `tril(A)` – нижня трикутна частина матриці;
- `triu(A)` – верхня трикутна частина матриці й ін.

Є ряд матричних функцій лінійної алгебри:

• cond(A) – число обумовленості основане на другій нормі;

• condeig(A) – вектор чисел обумовленості для власних значень матриці

• det(A) – детермінант (визначник) матриці;

• rank(A) – ранг матриці;

• norm(A) – найбільше сингулярне значення;

• orth(A) – ортонормальний базис матриці;

• null (A) – ортонормальний базис для нульового простору;

• rref (A) – трикутна форма (приведення за методом виключення Гаусса з частковим вибором ведучого елементу);

• trace(A) – слід матриці;

• chol(A) – верхня трикутна матриця, отримана за методом розкладання Холецького;

• inv(A) – обернена матриця (A^{-1});

• lu(A) – LU-розкладання матриці;

• qr(A) – QR-розкладання матриці;

• eig(A) – вектор власних значень матриці;

• hess(A) – верхня форма Хессенберга;

• schur(A) – матриця Шура і т.д.

Більшість елементарних і спеціальних функцій можуть мати аргументи у вигляді вектора чи матриці і обчислюють перетворені вектор чи матрицю. У MATLAB є можливість вести обробку розріджених матриць і багатовимірних масивів, у тому числі з різнотипними даними – записами. Приведені функції мають більш складні варіанти, зокрема, що допускають задання матриць довільної розмірності m×n при нерівних m і n.

Наступний приклад дає рішення системи з чотирьох лінійних рівнянь A·X=У трьома точними методами:

```
A=[2      1       0       1;
   1     -3       2       4;
   -5     0      -1      -7;
   1     -6       2       6];
B=[8   9    -5    0];
X1=B/a
X2=B*A^-1
X3=B*inv(A)
X1=
    3.0000   -4.0000  -1.0000  1.0000
X2 =
    3.0000   -4.0000  -1.0000  1.0000
x3 =
    3.0000   -4.0000  -1.0000  1.0000
```

Функція `nnls(A, У)` і ряд її варіантів вирішують систему лінійних рівнянь з мінімізацією помилки за методом найменших квадратів (див. приклад нижче):

```
» C=[4 3; 12 5; 3 12]; b=[1, 45, 41]; D=nnls(C, b')
    D =
        2.2242
        2.6954
```

Для рішення систем лінійних рівнянь з розрідженими матрицями MATLAB пропонує ряд функцій, що реалізують різні методи рішення:

- `bicg(A, У)` – двоспрямований метод спряжених градієнтів;
- `bicgstab(A, У)` – стійкий двоспрямований метод;
- `cgs(A, У)` – квадратичний метод спряжених градієнтів;
- `gmres(A, У, restart)` – метод мінімізації узагальненої нев'язки;
- `pcg (A, У)` – метод спряжених градієнтів;
- `qmr(A, У)` – метод квазімінімізації нев'язки (відхилу).

2 ЛАБОРАТОРНИЙ ПРАКТИКУМ

2.1 Спектральні характеристики сигналу

2.1.1 Періодичні сигнали

Відомо, що будь-яка періодична функція, яка задовольняє умови Діріхле, може бути представлена у вигляді нескінченної у загальному випадку суми гармонічних складових – рядом Фур'є. Умова Діріхле полягає у такому: функція $x(t)$ повинна бути обмеженою, кусково-неперервною та мати протягом періоду кінцеве число екстремумів.

Відомо дві форми розкладання в ряд Фур'є: тригонометрична й комплексна. Тригонометрична форма розкладання виражається у вигляді

$$x(t) = \frac{1}{2}A_0 + \sum_{k=1}^{\infty} A_k \cos(K\omega_0 t - \varphi_k). \qquad (2.1)$$

де $\frac{1}{2}A_0$ - постійна складова функції $x(t)$; $A_k \cos(K\omega_0 t - \varphi_k)$ – k-та гармонічна складова; $A_k, K\omega_0, \varphi_k$ – амплітуда, частота та початкова фаза k-та гармонічна складова; $\omega_0 = \frac{2\pi}{T}$ – частота основної (першої) гармоніки; T – період зміни функції $x(t)$.

В математичному відношенні зручніше оперувати комплексною формою ряду Фур'є представленою у вигляді

$$x(t) = \frac{1}{2}\sum_{k=-\infty}^{\infty} A_k \exp\{jK\omega_0 t\}, \qquad (2.2)$$

де $A_k = A_k \exp\{-j\varphi_k\}$ - комплексна амплітуда гармонійної складової частоти $\omega_k = K\omega_0$.

Комплексна амплітуда визначається через тимчасову функцію $x(t)$ за допомогою формули

$$A_k = \frac{2}{T}\int_{t_1}^{t_1+T} x(t)\exp\{-jK\omega_0 t\}dt. \qquad (2.3)$$

Сукупність амплітуд і відповідних частот гармонік прийнято називати *спектром амплітуд*.

Сукупність початкових фаз і відповідних частот гармонік називають *спектром фаз*.

На рис. 2.1 подані графічні зображення спектра амплітуд і спектра фаз періодичного сигналу.

Окремі спектральні складові у графічному зображенні спектра амплітуд називають *спектральними лініями*.

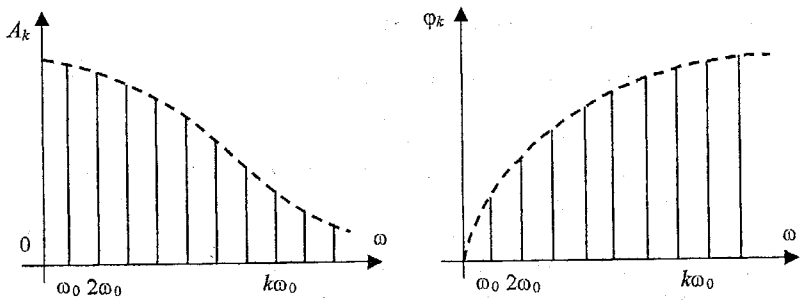


Рисунок 2.1 – Графічні зображення спектра амплітуд і спектра фаз періодичного сигналу

2.1.2 Практична ширина спектра сигналу

Практично всі канали зв'язку мають обмежену смугу пропускання. Отже, при передачі сигналу через реальний канал зв'язку може бути передана лише частина його частотного спектра.

За практичну ширину спектра сигналу приймають діапазон частот, в межах якого знаходиться найбільш вагома частина спектра сигналу. Вибір практичної ширини спектра сигналу визначається двома критеріями: енергетичним критерієм та критерієм допустимих спотворень форми сигналу.

2.1.3 Імпульси прямокутної форми

Розглянемо для прикладу послідовність прямокутних імпульсів тривалістю t, амплітудою A, із періодом проходження T (рис. 2.2).

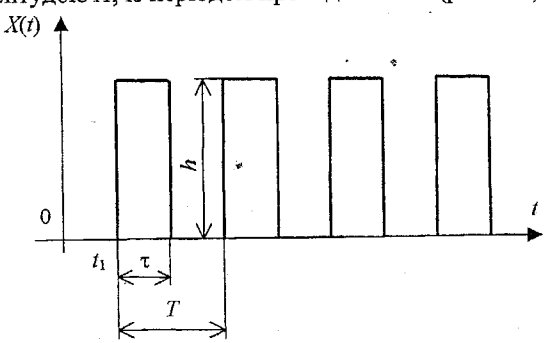


Рисунок 2.2 – Імпульси прямокутної форми

Розкладемо в ряд Фур'є періодичної послідовності прямокутних імпульсів представляється у вигляді

$$x(t)=\frac{\tau h}{T}\left[1+2\sum_{k=1}^{\infty}\frac{\sin\frac{K\omega_0\tau}{2}}{\frac{K\omega_0\tau}{2}}\cos K\omega_0\tau\right]. \qquad (2.4)$$

Спектр амплітуд такого сигналу показаний на рис. 2.3

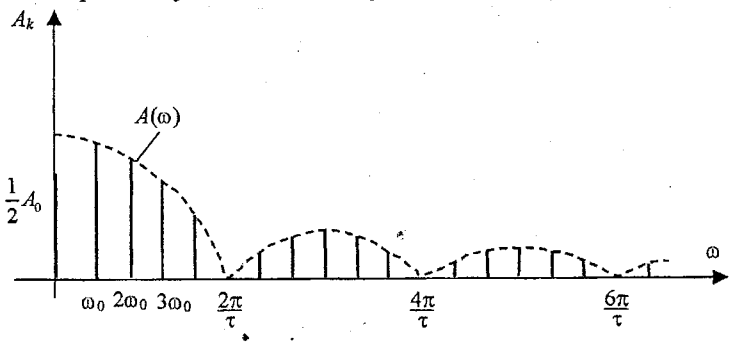


Рисунок 2.3 – Спектр амплітуд

Обвідна його визначається рівнянням

$$A(\omega)=2\frac{\tau h}{T}\left[\frac{\sin\frac{\omega\tau}{2}}{\frac{\omega\tau}{2}}\right], \qquad (2.5)$$

де $\omega = K\omega_0$ - для K-ої гармоніки.

2.1.4 Практична ширина спектра для прямокутних імпульсів

Можна показати, що для періодичної послідовності імпульсів прямокутної форми тривалістю $\tau=\frac{T}{2}$ достатньо практичну ширину спектра вибрати рівною $3\omega_0=\frac{6\pi}{T}=\frac{3\pi}{\tau}$. В цій області частот зосереджено 95% всієї потужності сигналу. Розглянемо одиночний прямокутний імпульс тривалістю T та величиною h. Спектральна густина такого сигналу визначається виразом

$$S(j\omega)=\int_{-\infty}^{\infty}x(t)\exp\{-j\omega t\}dt=\int_{\frac{-\tau}{2}}^{\frac{\tau}{2}}h\exp\{-j\omega t\}dt=\tau h\frac{\sin\frac{\omega\tau}{2}}{\frac{\omega\tau}{2}} \qquad (2.6)$$

Енергія сигналу, зосереджена в смузі частот від 0 до ω_0,

$$W_1 = \frac{1}{\pi}\int_0^{\omega_0}[S(\omega)]^2 d\omega = \frac{\tau^2 h^2}{\pi}\int_0^{\omega_1}\left[\frac{\sin\frac{\omega\tau}{2}}{\frac{\omega\tau}{2}}\right]^2 d\omega . \qquad (2.7)$$

Для оцінювання впливу ширини смуги пропускання каналу зв'язку на викривлення форми сигналів розглянемо проходження прямокутного імпульсу тривалістю T та величиною U через канал зв'язку, що являє собою ідеальний фільтр низьких частот. Коефіцієнт передачі цього фільтру виражається залежністю: $K(j\omega) = K\exp\{-j\omega T_0\}$, при цьому в діапазоні частот $0 \le \omega \le \omega_b$ модуль коефіцієнту передачі $K(\omega) = K = const$ і аргумента $\varphi(\omega) = \omega T_0$; поза цим діапазоном $K(\omega) = 0$.

В теорії кіл показано, що вихідний сигнал в цьому випадку може бути представлений в аналітичному вигляді

$$a_{вих}(t) = UK\left\{\int_0^{\omega_b}\frac{\sin\omega(t-T_0)}{\omega}d\omega - \int_0^{\omega_b}\frac{\sin\omega[t-(T_0-\tau)]}{\omega}d\omega\right\}. \qquad (2.8)$$

Форми переднього й заднього фронтів імпульсу спотворюються однаково. На рис. 2.4 показана форма переднього фронту вихідного сигналу.

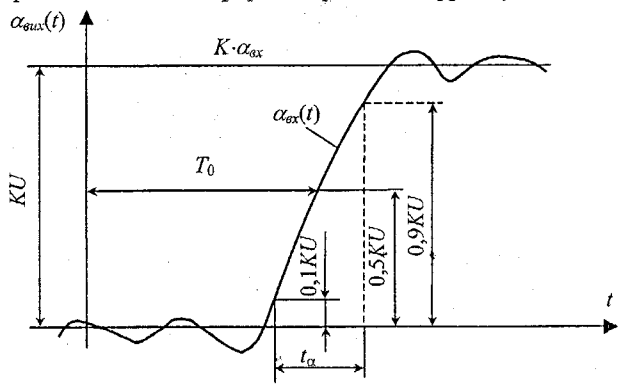


Рисунок 2.4 – Форма переднього фронту вихідного сигналу

Щоб вихідний сигнал зміг досягнути найбільшого значення, активна тривалість переднього фронту $t_{фа}$ повинна бути не більше тривалості вхідного імпульсу τ. При цьому, як показав аналіз, повинна бути справедлива умова $\tau \approx \frac{0,4}{f_b}$ або $f_b \approx \frac{0,4}{\tau}$.

2.1.5 Імпульси косинусоїдальної форми

Функція $x(t)$, що описує даний сигнал рис.2.5, може бути представлена у вигляді:

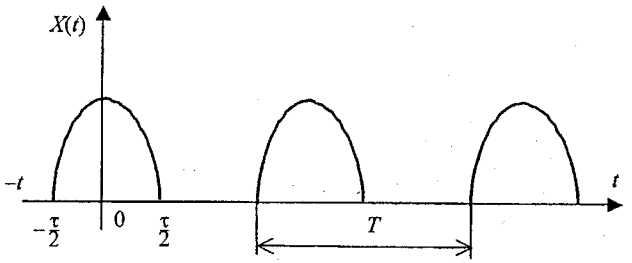


Рисунок 2.5 – Послідовність косинусоїдальних імпульсів

$$x(t)=\begin{cases} h\cos\omega_0 t & при\ -(\frac{\tau}{2}+iT)\le t\le\frac{\tau}{2}+iT \\ 0 & при\ -(\frac{3\tau}{2}+iT)<t<-\left(\frac{\tau}{2}+iT\right) \end{cases} \qquad (2.9)$$

$$\omega_0=\frac{2\pi}{T};\quad T=2\tau;\quad \frac{\tau}{2}+iT<t<\frac{3\tau}{2}+iT;\quad i=0,1,2,...$$

Комплексна амплітуда сигналу має вигляд

$$A_k=\frac{2}{T}\int\limits_{-\frac{\tau}{2}}^{\frac{\tau}{2}} h\cos\omega_0 t\exp\{-jK\omega_0 t\}\,dt\,. \qquad (2.10)$$

Після перетворення отримуємо

$$A_k=-\frac{h}{\pi(K-1)}\cos\frac{K\omega_0\tau}{2}+\frac{h}{\pi(K+1)}\cos\frac{K\omega_0\tau}{2}. \qquad (2.11)$$

Оскільки

$$\left|\cos\frac{K\omega_0\tau}{2}\right|=\left|\cos\frac{K\pi}{2}\right|=\begin{cases}1 & \text{при парному}\\ 0 & \text{при непарному}\end{cases},$$

то спектр сигналу містить тільки парні гармоніки рис. 2.6. При цьому комплексна амплітуда

$$A_k=-\frac{2h}{\pi(K^2-1)}, \qquad (2.12)$$

модуль комплексної амплітуди

$$A_k=\frac{2h}{\pi(K^2-1)}. \qquad (2.13)$$

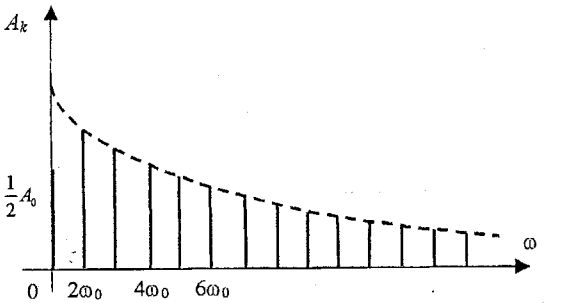


Рисунок 2.6 – Спектр амплітуд косинусоїдальних імпульсів

2.1.6 Пилкоподібне коливання

З подібними коливаннями завжди маємо справу в пристроях розгортки телевізорів та осцилографів рис. 2.7, а також в пристроях автоматичного регулювання. Оскільки дана функція є непарною, ряд Фур'є для неї має тільки синусні складові.

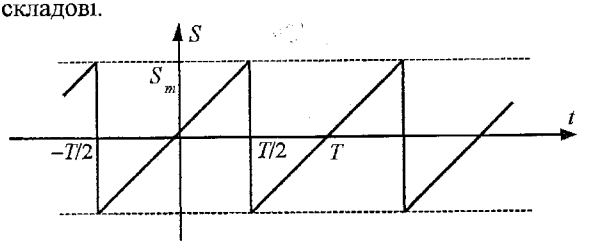


Рисунок 2.7 – Пилкоподібне коливання

Знайдемо загальний вираз для коефіцієнтів ряду, враховуючи, що на відрізку

$$-\frac{T}{2} < t < \frac{T}{2} \qquad x(t) = \frac{2S_m t}{T},$$

де S_m - постійна амплітуда

$$b_n = \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} x(t)\sin n\omega_1 t dt\,. \tag{2.14}$$

Після перетворення отримаємо

$$b_n = -\frac{2S_m}{n\pi}\cos n\pi \tag{2.15}$$

З виразу очевидно, що $b_n = \begin{cases} \dfrac{2S_m}{n\pi} & при \quad n = 1,3,5,... \\ -\dfrac{2S_m}{n\pi} & при \quad n = 2,4,6,... \end{cases}$ (2.16)

Враховуючи те, що $A_k = \sqrt{a_n^2 + b_n^2}$ амплітуди синусних гармонік $A_k = b_n$, а початкові фази $\varphi_k = 0$.

Тоді остаточний вигляд ряду Фур'є для даного сигналу

$$x(t) = \frac{2S_n}{\pi}\left[\sin\omega_1 t + \frac{1}{2}\sin(2\omega_1 t + \pi) + \frac{1}{3}\sin\omega_1 t + \frac{1}{4}\sin(4\omega_1 t + \pi)...\right] \qquad (2.17)$$

Графіки амплітудного й фазового спектрів такого коливання показані на рис. 2.8.

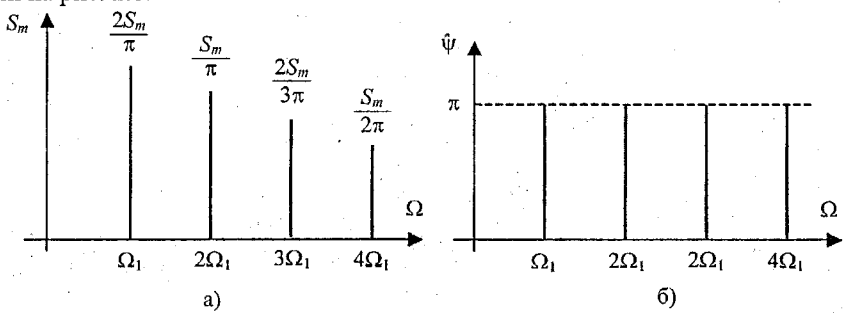


Рисунок 2.8 – Графіки амплітудного (а) і фазового (б) спектрів коливання ряду Фур'є

2.1.7 Неперіодичні сигнали

Будь-який неперіодичний сигнал можна розглядати як періодичний, період зміни якого дорівнює нескінченності. У зв'язку з цим розглянутий раніше спектральний аналіз періодичних процесів може бути узагальнений і на неперіодичний сигнал.

Розглянемо як буде змінюватись спектр неперіодичного сигналу при необмеженому збільшенні періоду зміни сигналу. При збільшенні періоду T інтервали між суміжними частотами в спектрі сигналу і амплітуди спектральних складових зменшується і в границі при $T \to \infty$ стають нескінченно малими величинами. При цьому спектральний розклад неперіодичного сигналу відображається рядом Фур'є.

Комплексна форма неперіодичного сигналу має вигляд

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(j\omega)^* \exp(j\omega t)d\omega, \qquad (2.18)$$

де $S(j\omega)=S(\omega)\exp(j\phi(\omega))$ - спектральна щільність сигналу, $S(\omega)=|S(j\omega)|$ - амплітудно-частотна характеристика сигналу, $\phi(\omega)$ - фазо-частотна характеристика сигналу.

Попередній вираз називається формулою зворотного перетворення Фур'є.

Представлення неперіодичної функції інтегралом Фур'є можливо при виконанні таких умов:

1. Функція $x(t)$ задовольняє умову Діріхле;
2. Функція $x(t)$ абсолютно інтегрована

$$\int_{-\infty}^{\infty}|x(t)|dt<\infty. \tag{2.19}$$

Таким чином, спектр неперіодичного сигналу на відміну від спектра періодичного сигналу є суцільним і являє собою суму нескінченної кількості гармонічних складових із нескінченно малими складовими.

Амплітуди гармонічних складових можуть бути представлені у вигляді

$$dA=\frac{1}{2\pi}S(j\omega)d\omega, \tag{2.20}$$

звідки спектральна щільність визначається виразом

$$S(j\omega)=2\pi\frac{dA}{d\omega}. \tag{2.21}$$

Спектральна щільність пов'язана з функцією часу через пряме перетворення Фур'є

$$S(j\omega)=\int_{-\infty}^{\infty}x(t)^{*}\exp(-j\omega t)dt. \tag{2.22}$$

Спектральна щільність однозначно відображає неперіодичний сигнал і задовольняє умови:

1. $\lim\limits_{\omega\to\infty}S(\omega)=0$;
2. Модуль спектральної щільності є парною, а аргумент непарною функцією частоти

$$S(\omega)=S(-\omega);\quad \varphi(\omega)=-\varphi(-\omega). \tag{2.23}$$

2.1.8 Спектр експоненціального імпульсу

Експоненціальний імпульс визначається функцією та приведений на рис. 2.9

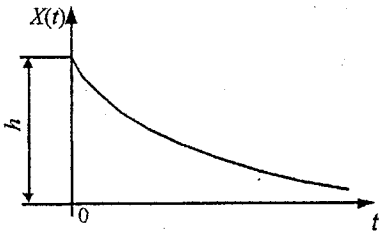


Рисунок 2.9 – Експоненціальний імпульс

$$x(t)=\begin{cases} h\exp(-\beta t) & при \quad t\geq 0; \\ 0 & при \quad t<0. \end{cases} \tag{2.24}$$

модуль і фаза спектральної щільності визначаються відповідно виразами

$$S(\omega)=\frac{h}{\sqrt{\beta^2+\omega^2}}; \qquad \varphi(\omega)=arctg\frac{\omega}{\beta}, \tag{2.25}$$

де h – початковий рівень сигналу.

На рис. 2.10 приведені графіки модуля і фази спектральної щільності й експоненціального імпульсу.

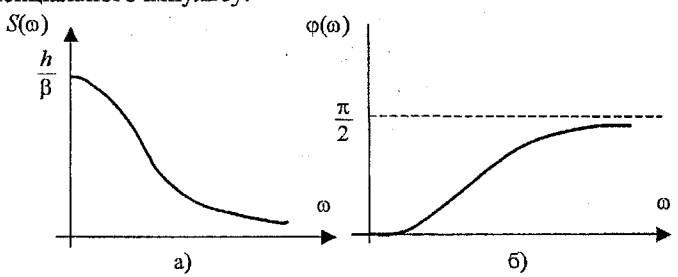


Рисунок 2.10 – Графіки модуля (а) і фази (б) спектральної щільності експоненціального імпульсу

2.1.9 Спектр сигналу ввімкнення

Сигнал ввімкнення Рис. 2.11 визначається функцією

$$x(t)=A_{ex}1(t)=\begin{cases} A_{ex} & при \quad t\geq 0; \\ 0 & при \quad t<0, \end{cases} \tag{2.26}$$

де $A_{ex}=U$ – величина сигналу ввімкнення.

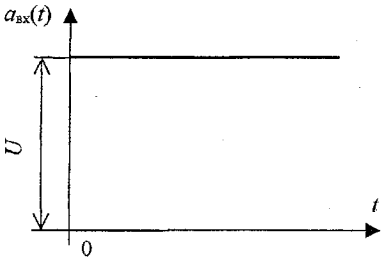


Рисунок 2.11 – Сигнал ввімкнення

Одинична функція $1(t)$ не задовольняє умову абсолютного інтегрування, тому до неї не можна застосувати перетворення Фур'є. Однак її можна розглядати як утворення з імпульсу експоненціальної форми при необмеженому зменшенні його коефіцієнта затухання $\beta \to 0$. У відповідності з цим спектральна щільність сигналу ввімкнення буде дорівнювати

$$S(j\omega) = \lim_{\beta \to 0} \frac{A_{вх}}{\beta + j\omega} = \frac{A_{вх}}{\omega} \exp(-j\frac{\pi}{2}). \qquad (2.27)$$

Звідки модуль і фаза спектральної щільності визначаються

$$S(\omega) = \frac{A_{вх}}{\omega}; \qquad \varphi(\omega) = \frac{\pi}{2}. \qquad (2.28)$$

Модуль і фаза спектральної щільності сигналу ввімкнення приведені на рис. 2.12.

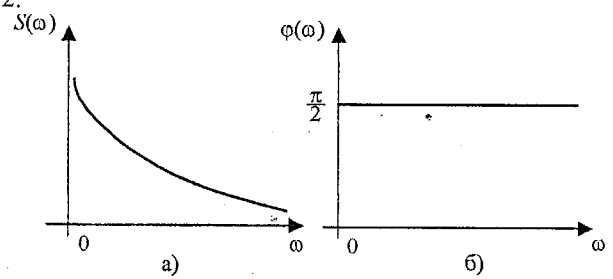


Рисунок 2.12 – Графіки модуля (а) і фази (б) спектральної щільності сигналу ввімкнення

2.1.10 Спектр дельта-функції

Дельта-функцію (рис. 2.13) можна трактувати як граничну форму прямокутного імпульсу тривалістю τ і амплітуди $\frac{1}{\tau}$, яка отримується при $\tau \to 0$.

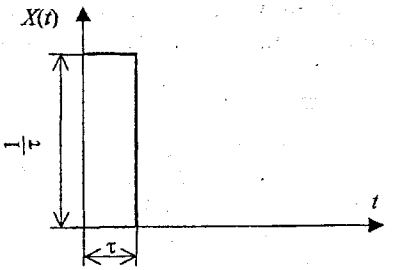


Рисунок 2.13 – Дельта-функція

Тоді прийнявши амплітуду імпульсу рівною $h=\frac{1}{\tau}$ отримаємо

$$S(j\omega)=\lim_{\tau\to 0}\frac{\sin\frac{\omega\tau}{2}}{\frac{\omega\tau}{2}}=1. \qquad (2.29)$$

Модуль і фаза спектральної щільності дорівнюють

$$S(\omega)=1; \qquad \varphi(\omega)=0.$$

Графік спектральної щільності дельта-функції показаний на рис. 2.14.

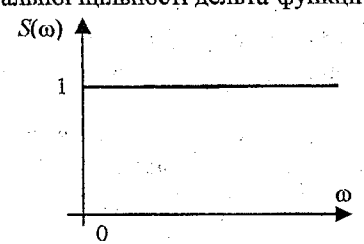


Рисунок 2.14 – Графік спектральної щільності дельта-функції

2.1.11 Лабораторна робота №1

Мета роботи: Виконати класичний спектральний аналіз і синтез функції. Виконати перетворення сигналів, використовуючи пряме й зворотне перетворення Фур'є. Виконати спектральний аналіз і синтез функції за допомогою швидкого перетворення Фур'є (ШПФ).

Хід роботи

Завдання 1. Обчислити перші шість пар коефіцієнтів розкладання в ряд Фур'є функції $f(t)$ на відрізку $[0, 2\pi]$. Побудувати графіки 1, 2 і 3 гармонік.

Виконати гармонічний синтез функції $f(t)$ за 1, 2 і 3 гармоніками. Результати синтезу відобразити графічно. Варіанти завдань приведені в Таблиці 2.1.

Таблиця 2.1 - Варіанти завдання 1

вар.	$f(t)$	вар.	$f(t)$	вар.	$f(t)$
1	$\frac{\cos t}{1+\cos^2 2t}$	6	$\cos t \cos\lvert\sin t\rvert$	11	$\sin\left(\sqrt{1+t^2}\right)$
2	$\frac{\sin t}{1+\cos^2 2t}$	7	$arctg\left(\cos\frac{1}{2}t\right)$	12	$\cos\left(\sqrt{1+t^2}\right)$
3	$\frac{\sin 2t+\sin^2 3t}{3+\sin t+\cos 2t}$	8	$e^{\sin\frac{1}{3}t}$	13	$e^{-10(t-\pi)^2}$
4	$\frac{\sin 3t}{\lvert\sin t\rvert+\lvert\cos t\rvert}$	9	$\lvert\sin t\rvert+\lvert\sin 2t\rvert$	14	$e^{\cos\frac{1}{3}t}$
5	$\cos(e^{\lvert\sin 3t\rvert})$	10	$\sin\left(\frac{1}{2}t\right)^2$	15	$e^{-\cos\frac{1}{2}t}\cos(\sin t)$

Завдання 2. Виконати класичний спектральний аналіз і синтез функції $f(t)$. Відобразити графічно спектри амплітуд і фаз, результат спектрального синтезу функції $f(t)$.

Завдання 3. Виконати чисельний спектральний аналіз і синтез функції $f(t)$. Для цього необхідно задати вихідну функцію $f(t)$ дискретно в 32 відліках. Відобразити графічно спектри амплітуд і фаз; результат спектрального синтезу функції $f(t)$.

Завдання 4. Виконати спектральний аналіз і синтез функції $f(t)$ за допомогою ШПФ. Для цього необхідно:

- задати вихідну функцію $f(t)$ дискретно в 128 відліках;
- виконати пряме ШПФ за допомогою функції *fft* і відобразити графічно знайдені спектри амплітуд і фаз перших шести гармонік;
- виконати зворотне ШПФ за допомогою функції *ifft* і відобразити графічно результат спектрального синтезу функції $f(t)$.

Завдання 5. Виконати фільтрацію функції $f(t)$ за допомогою ШПФ:

- синтезувати функцію $f(t)$ у вигляді корисного сигналу, представленого 128 відліками вектора v;
- до корисного сигналу v приєднати шум за допомогою функції *rnd* (*rnd*(2) - 1) і сформувати вектор з 128 відліків зашумленого сигналу s;

- перетворити сигнал із шумом *s* із часової області в частотну, використовуючи пряме ШПФ (функція *fft*). У результаті вийде сигнал *f* з 64 частотних складових;
- виконати фільтруюче перетворення за допомогою функції Хевісайда (параметр фільтрації $\alpha = 2$);
- за допомогою функції *ifft* виконати зворотне ШПФ і одержати вектор вихідного сигналу *h*;
- побудувати графіки корисного сигналу *v* і сигналу, отриманого фільтрацією зашумленого сигналу *s*.

В звіті повинні бути подані часові та частотні діаграми характеристик сигналів, основні параметри, висновки за результатами досліджень.

Контрольні запитання

1. При яких умовах періодична функція може бути представлена рядом Фур'є?
2. Що розуміють під спектром амплітуд?
3. Що розуміють під спектром фаз?
4. Які характерні особливості спектра періодичного сигналу?
5. Що розуміють під практичною шириною спектра періодичного сигналу?
6. Які відомі критерії вибору практичної ширини спектра сигналу?
7. Як визначається відносна величина енергії одиночного імпульсу?
8. Як визначається спектральна щільність прямокутного імпульсу?
9. При яких умовах неперіодична функція може бути представлена інтегралом Фур'є?
10. Яким чином можна отримати спектр неперіодичного сигналу безпосередньо зі спектра відповідного періодичного сигналу?
11. Які характерні особливості спектра неперіодичного сигналу?
12. Що розуміють під практичною шириною спектра неперіодичного сигналу?

Приклад. Виконати класичний спектральний аналіз і синтез функції.

Завдання 1. Обчислимо перші шість пар коефіцієнтів розкладання в ряд Фур'є функції $f(t)$ на відрізку $[0, 2\pi]$. Побудуємо графіки 1, 2 і 3 гармонік.

$t := 0, 0.001 .. 2\pi \quad f1(t) := e^{\sin\left(\frac{1}{3}\right)t} \quad f(t) := if(t < 2\pi, f1(t), -f1(t))$

$T := 2\pi \quad \omega 1 := 2\frac{\pi}{T} \quad k := 0 .. 6$

$a_k := \left(\frac{2}{T}\right) \cdot \int_0^T f(t)\cos(\omega 1 \cdot k \cdot t)\, dt \qquad b_k := \left(\frac{2}{T}\right) \cdot \int_0^T f(t)\sin(\omega 1 \cdot k \cdot t)\, dt$

$a_1 = 0.641 \quad a_2 = 0.173 \quad a_3 = 0.078 \quad a_4 = 0.044 \quad a_5 = 0.028 \quad a_6 = 0.02$

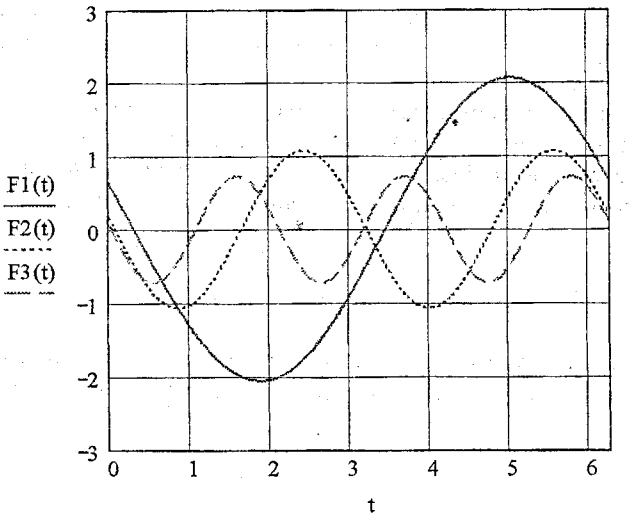
$b_1 = -1.959 \quad b_2 = -1.056 \quad b_3 = -0.714 \quad b_4 = -0.539 \quad b_5 = -0.432 \quad b_6 = -0.36$

Гармонічний аналіз

$F1(t) := a_1 \cdot \cos(\omega 1 \cdot 1t) + b_1 \sin(\omega 1 \cdot 1 \cdot t)$

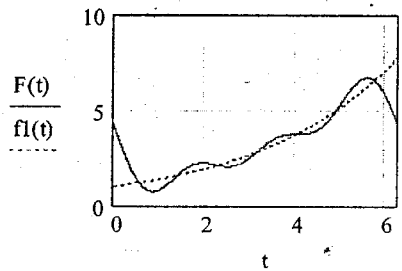
$F2(t) := a_2 \cdot \cos(\omega 1 \cdot 2t) + b_2 \sin(\omega 1 \cdot 2 \cdot t)$

$F3(t) := a_3 \cdot \cos(\omega 1 \cdot 3t) + b_3 \sin(\omega 1 \cdot 3 \cdot t)$



Гармонічний синтез

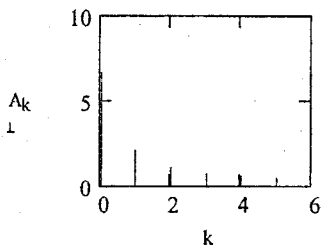
$$F(t) := \left(\frac{a_0}{2}\right) + \left[\sum_{k=1}^{3} \left(a_k \cdot \cos(\omega 1 \cdot k \cdot t) + b_k \cdot \sin(\omega 1 \cdot k \cdot t)\right)\right]$$

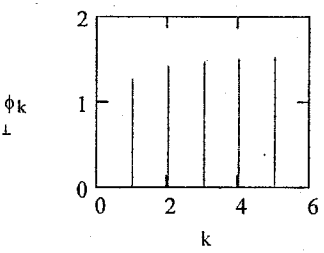


Завдання 2. Виконаємо класичний спектральний аналіз і синтез функції *f(t)*. Відобразимо графічно спектри амплітуд і фаз, результат спектрального синтезу функції *f(t)*.

Класичний спектральний аналіз

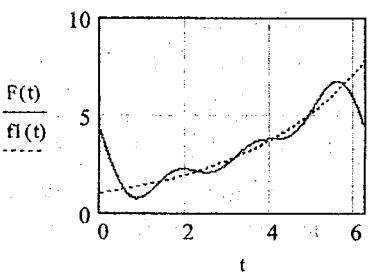
$$A_k := \sqrt{(a_k)^2 + (b_k)^2} \qquad \phi_k := -\operatorname{atan}\left(\frac{b_k}{a_k}\right)$$





Спектральний синтез

$$F(t) := \frac{-a_0}{2} + \left(\sum_{k=1}^{3} A_k \cdot \cos(\omega 1 \cdot k \cdot t + \phi_k)\right)$$



Завдання 3. Виконаємо чисельний спектральний аналіз і синтез функції $f(t)$. Для цього задамо вихідну функцію $f(t)$ дискретно в 32 відліках.

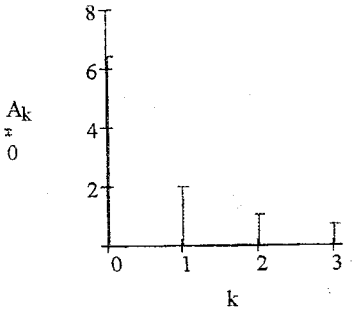
Чисельний спектральний аналіз

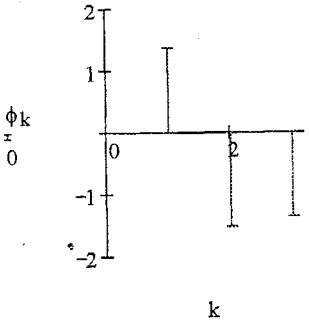
$$T := 2\pi \quad \omega 1 := \frac{2\pi}{T} \quad k := 0..3 \quad N := 32 \quad \Delta t := \frac{T}{N}$$

$$k := 0..63 \quad t_k := k \cdot \frac{T}{N} \quad v_k := f1(t_k)$$

$$a_k := \frac{2}{N} \cdot \sum_{i=0}^{N-1} v_i \cos(\omega 1 \cdot k \cdot i\Delta t) \qquad b_k := \frac{2}{N} \cdot \sum_{i=0}^{N-1} v_i \sin(\omega 1 \cdot k \cdot i\Delta t)$$

$$A_k := \sqrt{(a_k)^2 + (b_k)^2} \qquad \phi_k := -\operatorname{atan}\left(\frac{b_k}{a_k}\right)$$





Завдання 4. Виконаємо спектральний аналіз і синтез функції $f(t)$ за допомогою ШПФ. Для цього:

- задамо вихідну функцію $f(t)$ дискретно в 128 відліках;
- виконаємо пряме ШПФ за допомогою функції *fft* і відобразимо графічно знайдені спектри амплітуд і фаз перших шести гармонік;
- виконаємо зворотне ШПФ за допомогою функції *ifft* і відобразимо графічно результат спектрального синтезу функції $f(t)$.

$$k := 0 .. 15 \quad t_k := \frac{T \cdot k}{128} \quad v_k := f(t_k) \quad F := \mathrm{fft}(v)$$

F =

	0
0	112.001
1	30.812-54.397i
2	8.089-35.137i
3	2.24-24i
4	-0.662-17.912i
5	-2.083-14.71i
6	-2.357-12.523i
7	-2.428-10.443i
8	-2.865-8.772i
9	-3.285-7.823i
10	-3.277-7.163i
11	-3.137-6.312i
12	-3.281-5.458i
13	-3.536-4.965i
14	-3.532-4.675i
15	-3.38-4.214i

$$d := 6$$

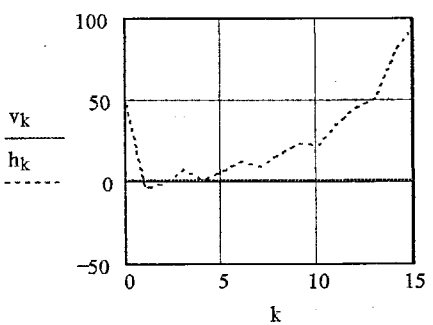
$$i := 0 .. d$$

$$A_i := \sqrt{\left(\mathrm{Re}\left(F_i\right)^2 + \mathrm{Im}\left(F_i\right)^2\right)}$$

$$\phi_i := -\arg\left(F_i\right)$$

$$j := 0 .. 8 \qquad g_j := \mathrm{if}\left(j < d, F_j, 0\right)$$

$$h := \mathrm{ifft}(g)$$



Завдання 5. Виконаємо фільтрацію функції *f(t)* за допомогою ШПФ:

- синтезуємо функцію *f(t)* у виді корисного сигналу, представленого 128 відліками вектора *v*;
- до корисного сигналу *v* приєднаємо шум за допомогою функції *rnd* (*rnd*(2) - 1) і сформуємо вектор з 128 відліків зашумленого сигналу *s*;
- перетворимо сигнал із шумом *s* із часової області в частотну, використовуючи пряме ШПФ (функція *fft*);
- виконаємо фільтруюче перетворення за допомогою функції Хевісайда (параметр фільтрації $\alpha = 2$);
- за допомогою функції *ifft* виконаємо зворотне ШПФ і отримаємо вектор вихідного сигналу *h*;
- побудуємо графіки корисного сигналу *v* і сигналу, отриманого фільтрацією зашумленого сигналу *s*.

$k := 0 .. 127 \qquad t_k := k \cdot \frac{T}{128} \qquad v_k := f(t_k)$

$S_k := v_k + (rnd(2) - 1)$

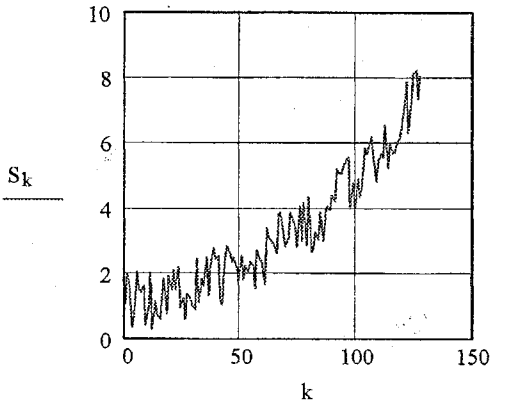
$\alpha := 2$

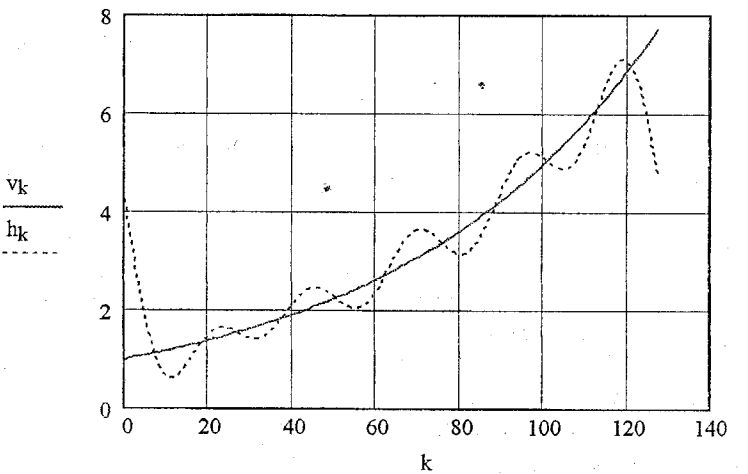
$f := fft(S)$

$j := 0 .. 64$

$g_j := f_j \cdot \Phi\left(|f_j| - \alpha\right)$

$h := ifft(g)$





2.2 Квантування сигналів

2.2.1 Перетворення неперервних сигналів у дискретні

Сигнали, які використовуються в сучасній апаратурі можна зберігати, передавати та обробляти як у вигляді неперервних, так і у вигляді дискретних. Враховуючи рівень розвитку обчислювальної техніки, перевага надається використанню дискретних сигналів. Це обумовлено тим, що передача неперервних сигналів зустрічає труднощі, які пов'язані з появою апаратурних помилок, що виникають внаслідок нестабільності параметрів ліній зв'язку і т. ін. У зв'язку з цим виникає необхідність у квантуванні сигналу для перетворення його в дискретний.

2.2.2 Квантування за рівнем

При квантуванні за рівнем неперервна множина значень функції $x(t)$ замінюється множиною дискретних значень. Для цього в діапазоні неперервних значень функції $x(t)$ вибирається скінчена кількість дискретних значень цієї функції (дискретних рівнів) і в процесі квантування значення функції $x(t)$ в кожен момент часу замінюється дискретним значенням, яке ближче до нього. В результаті квантування виходить східчаста функція $x_g(t)$.

Фактично квантування за рівнем може виконуватись двома шляхами. При першому миттєве значення функції $x(t)$ замінюється меншим дискретним значенням (рис. 2.15,а). При другому методі квантування миттєве значення функції замінюється найближчим меншим або більшим значенням у залежності від того, яке з цих дискретних значень ближче до миттєвого значення функції. В цьому випадку перехід східчастої функції з одного ступеня на інший відбувається в ті моменти часу, коли початкова неперервна функція $x(t)$ перетинає середину між відповідними сусідніми дискретними рівнями (рис. 2.15,б).

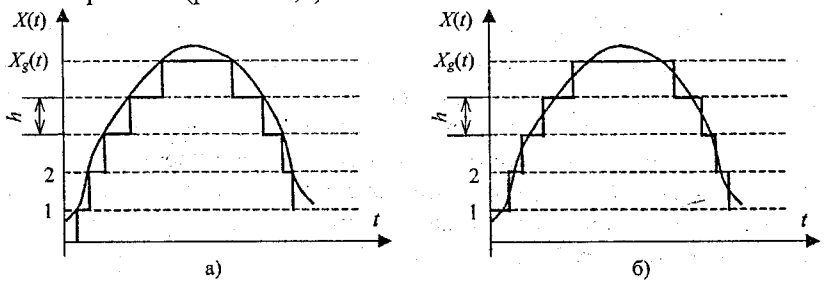


Рисунок 2.15 – Квантування за рівнем

2.2.3 Квантування в часі

При квантуванні і часі неперервна за аргументом функція $x(t)$, яка описує сигнал, перетворюється в функцію $x_g(t)$ дискретного аргумента (рис. 3.2).

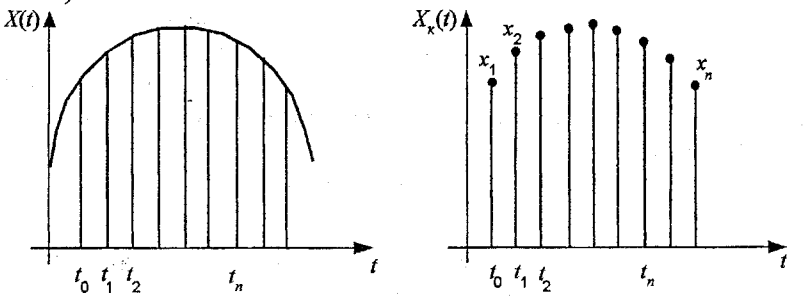


Рисунок 2.16 – Квантування в часі

Таке перетворення може бути виконано шляхом заміни функції її миттєвими значеннями в певні моменти часу: $t_0, t_1, t_2, ..., t_n$. В результаті функція $x(t)$ буде описуватися сукупністю її миттєвих значень $x_g(t_i)$, де $i = \{0..n\}$.

Часовий інтервал $T_k = t_i - t_{i-1}$ між двома сусідніми фіксованими моментами часу, в межах яких задається функція, називається *інтервалом часового квантування*. Величина, обернена до інтервалу часового квантування називається *частотою квантування*

$$f_k = \frac{1}{T_k}.$$

Квантування сигналів у часі може бути рівномірним та нерівномірним. У випадку рівномірного квантування інтервал квантування T_k - величина постійна і вибирається на основі апріорних відомостей про характеристики сигналу. При нерівномірному квантуванні інтервал квантування змінюється за певним законом (у залежності від динамічної зміни характеристик сигналу). Відомо кілька критеріїв вибору інтервалів часового квантування. До них відносяться: частотний критерій В.А. Котельникова та кореляційний критерій Н.А. Железнова.

2.2.4 Частотний критерій Котельникова

Частотний критерій Котельникова базується на теоремі Котельникова: якщо неперервна функція $x(t)$ відповідає умовам Діріхле (тобто, об-

межена, кусково-неперервна і має кінцеву кількість екстремумів) та її спектр обмежений деякою частотою f_c - то вона повністю визначається послідовністю своїх значень в точках, які розташовані одна від одної на відстані

$$T_k = \frac{1}{2f_c}. \tag{2.30}$$

Аналітично теорема Котельникова записується інтерполяційним рядом

$$x(t) = \sum_{k=-\infty}^{\infty} x(k\Delta t)\frac{\sin[\omega_c(t-k\Delta t)]}{\omega_c(t-k\Delta t)}, \tag{2.31}$$

де $\Delta t = \frac{\pi}{\omega_c} = \frac{1}{2f_c}$.

З формули (2.31) видно, що неперервна функція з обмеженим спектром може бути представлена у вигляді суми нескінченно великої кількості членів, кожний з яких є множником функції вигляду $\frac{\sin y}{y}$ (функції відліку) та коефіцієнта $x(k\Delta t)$, який визначає значення функції $x(t)$ в момент відліку.

Функція вигляду $\frac{\sin\omega_c\tau}{\omega_c\tau}$ являє собою реакцію ідеального фільтра нижніх частот з граничною частотою ω_c на дельта-функцію. Тоді, якщо через такий фільтр пропустити квантований в часі сигнал із частотою квантування $f_c = 2f_c = \frac{\omega_c}{\pi}$, то знаходячи добуток вихідних сигналів фільтра, можна отримати вихідний неперервний сигнал функції $x(t)$.

При використанні теореми Котельникова для квантування сигналів реальний спектр сигналу умовно обмежують деяким діапазоном частот від нуля до ω_c, в якому зосереджена основна частина енергії спектра сигналу.

Енергія сигналу, що відсікається призводить до появи похибки, яка в загальному вигляді виражається відношенням енергії відсіченої частини сигналу до загальної енергії сигналу. Але похибку зручніше виразити як дисперсію приведеної похибки

$$D_{\varpi} = \frac{\gamma\omega_c P_T}{(x_{\max} - x_{\min})^2}, \tag{2.32}$$

де похибка дорівнює

$$\gamma\omega_c = \frac{\int_{\omega}^{x} S(j\omega)^2 d\omega}{\int_{0}^{x} S(j\omega)^2 d\omega}, \tag{2.33}$$

Середня потужність сигналу може бути обчислена за формулою

$$P_T = \frac{1}{\pi T_c}\int_0^{\infty} |S(j\omega)|^2 \, d\omega = \int_{x_{\min}}^{x_{\max}} x^2 \omega(x) dx \,. \qquad (2.34)$$

Якщо відомі $D_{\omega_c}, x_{\max}, x_{\min}$ та спектри функції $x(t)$ можна визначити частоту, яка обмежує спектр сигналу.

2.2.5 Критерій допустимого відхилення

Труднощі практичної реалізації методів Котельникова та Желєзнова обумовлюють необхідність використання в ряді випадків інших методів вибору необхідної частини квантування сигналів у часі. У тих випадках коли закон зміни безупинної функції з визначеною вірогідністю відомий, доцільним може виявитись метод, заснований на заміні вихідної функції апроксимуючою. У загальному випадку вихідна функція апроксимується поліномом, крива якого збігається з кривою функції для заданих дискретних моментів часу. При цьому інтервали тимчасового квантування повинні вибиратися таким чином, щоб у їхніх межах максимальна величина чи середнє квадратичне значення відхилення апроксимуючої функції від вихідної не перевищували допустимих значень.

На практиці найбільш часто застосовується кусково-лінійна чи східчаста апроксимація рис 2.17

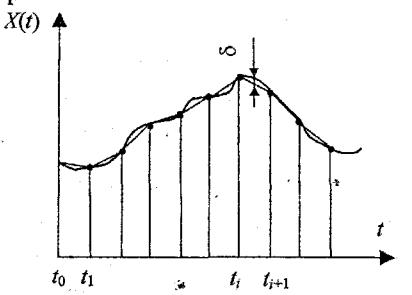


Рисунок 2.17 – Кусково-лінійна апроксимація

При кусково-лінійній апроксимації поліномом першого порядку всі точки кривої вихідної функції, що відповідають відліковим моментам часу, з'єднуються відрізками прямих. Якщо дотримуватись критерію найбільшого відхилення, то вибір частоти квантування потрібно робити за умови, щоб відхилення апроксимуючої ламаної прямої від вихідної функції на кожному з інтервалів дискретизації не перевищувало б заданого значення.

Задача може бути вирішена за допомогою інтерполяційної формули Ньютона, відповідно до якої значення функції для будь-якого моменту ча-

су у середині інтервалу $\Delta t = t_{i+1} - t_i$ визначається виразом $x^*(t) = x(t_i) + a_i\left[x(t_{i+1} - t_i)\right]$,

де
$$a_i = \frac{t - t_i}{t_{i+1} - t_i}, \quad (2.35)$$

Похибка апроксимації δ визначається залишковим членом $L(t)$ інтерполяційної формули $|\delta| = |L(t)| = x \cdot (t) - x(t)$.

У розглянутому випадку залишковий член виражається в такий спосіб

$$L(t) = \frac{1}{2}\frac{d^2x(t)}{dt^2}(t - t_i)(t - t_{i+1}). \quad (2.36)$$

Очевидно, максимальне значення похибки апроксимації

$$|\delta|_{\max} = |L(t)|_{\max} = \frac{1}{2}\left(\frac{d^2x(t)}{dt^2}\right)_{\max}\left(\frac{t_{i+1} - t_i}{2}\right) = \frac{1}{8}\left(\frac{d^2x(t)}{dt^2}\right)_{\max} \cdot \Delta_i^2. \quad (2.37)$$

Отже, задаючись допустимою похибкою апроксимації $\delta_{\max}$, можна визначити інтервал і частоту квантування

$$\Delta t = \sqrt{\frac{8\delta_{\max}}{\left(\frac{d^2x(t)}{dt^2}\right)_{\max}}}; \quad (2.38)$$

$$f_k = \sqrt{\frac{\left(\frac{d^2x(t)}{dt^2}\right)_{\max}}{8\delta_{\max}}}. \quad (2.39)$$

При східчастій апроксимації поліномом нульового порядку про значення функції в будь-який момент часу t в інтервалі $t_i \le t \le t_{i+1}$ судять за значенням функції $x(t)$, зафіксованої в точці t_i. Залишковий член інтерполяційної формули визначається в цьому випадку виразом

$$L(t) = \frac{dx(t)}{dt}(t - t_i). \quad (2.40)$$

Максимальне значення похибки апроксимації

$$|\delta|_{\max} = |L(t)|_{\max} = \left(\frac{dx(t)}{dt}\right)_{\max} \cdot \Delta t, \quad (2.41)$$

звідки необхідні значення інтервалу й частоти квантування визначається виразом

$$\Delta t = \frac{|\delta|_{\max}}{\left(\frac{dx(t)}{dt}\right)_{\max}}; \qquad (2.42)$$

$$f_k = \frac{\left(\frac{dx(t)}{dt}\right)_{\max}}{|\delta|_{\max}}. \qquad (2.43)$$

Критерій найбільшого відхилення відноситься до детермінованого випадку, коли передбачається відомий закон зміни вихідної функції $x(t)$. Оскільки реальні сигнали практично завжди мають випадковий характер, то цей критерій страждає визначеною неточністю. При випадковому характері вихідної функції доцільно скористатись критерієм середньо-квадратичного відхилення.

2.2.6 Лабораторна робота №2

Мета роботи. Ознайомитися з методами квантування сигналів, дослідити параметри сигналів при квантуванні та наступному їх відновленню, провести практичні розрахунки параметрів квантування.

Хід роботи

1. Визначити період квантування T_k на основі критерію Котельникова:
   а) побудувати графік функції заданого сигналу;
   б) зобразити даний сигнал за допомогою інтерполяційного ряду теореми Котельникова;
   в) визначити спектральну щільність сигналу;
   г) обчислити середню потужність сигналу;
   д) обчислити дисперсію приведеної похибки.
2. Визначити період квантування T_k на основі критерію допустимого відхилення:
   а) побудувати графік функції заданого сигналу;
   б) визначити крок квантування;
   в) визначити значення апроксимації функції у вузлах апроксимації;
   г) визначити функцію апроксимації;
   д) побудувати функцію апроксимації.
   е) визначити експериментально похибку апроксимації;

Контрольні запитання

1. В чому полягає сутність квантування неперервного сигналу у часі та за рівнем?

2. Охарактеризуйте переваги дискретної та неперервної передачі інформації.
3. Яка ціль квантування сигналів?
4. Сформулюйте частотний критерій Котельникова.
5. Сформулюйте критерій Железнова.
6. Сформулюйте критерій допустимого відхилення.
7. Які переваги та недоліки відомих критеріїв?
8. Поясніть фізичну можливість заміни неперервної функції з обмеженим спектром сукупністю її миттєвих значень.
9. Які теоретичні незручності при використанні в якості моделі сигналу функції з обмеженим спектром?
10. В чому полягають труднощі технічної реалізації способу передачі інформації, який ґрунтується на теоремі Котельникова?

Таблиця 2.2 – Варіанти завдань

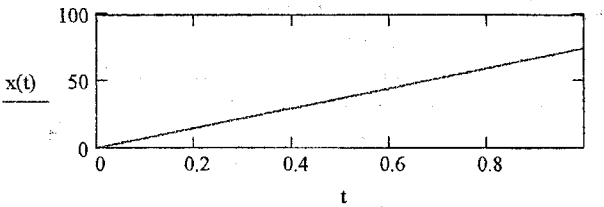
Варіант	Функція	A_0	Параметр a	Межі	
				t_1	t_2
1	$x(t) = A_0 \exp(-at)$	2	5	0	2
2	$x(t) = A_0 \cos(at)$	1	π	0	2π
3	$x(t) = A_0 x^{2a}$	0,5	1	0	6
4	$x(t) = A_0 (at)^2$	0,3	2	0	3
5	$x(t) = A_0 + tg(at)t^2$	3	0,5	0	3
6	$x(t) = A_0 + arctg(at)t^3$	2	0,3	0	4
7	$x(t) = A_0 \arcsin(at)t$	2	0,3	0	3
8	$x(t) = A_0 \arccos(at)t$	3	0,25	0	3
9	$x(t) = A_0 \sin(at)$	1	2	0	2
10	$x(t) = (A_0 t)^2 \lg(at)$	0,5	0,25	0	3

Приклад . Дослідження параметрів квантованого сигналу.

1. Знайдемо період квантування T_k на основі критерія Котельникова.
Побудуємо графік функції заданого сигналу

$A0 := 3 \qquad \alpha := 5 \qquad t := 0..1$

$x(t) := A0 \cdot (\alpha \cdot t)^2$

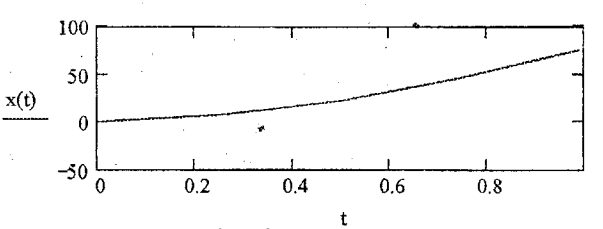


$\Delta t := 0.14 \qquad t := 0, 0.248..1$

$\omega c(\Delta t) := \frac{\pi}{\Delta t} \qquad \omega c(\Delta t) = 22.44$

Зобразимо даний сигнал інтерполяційним рядом

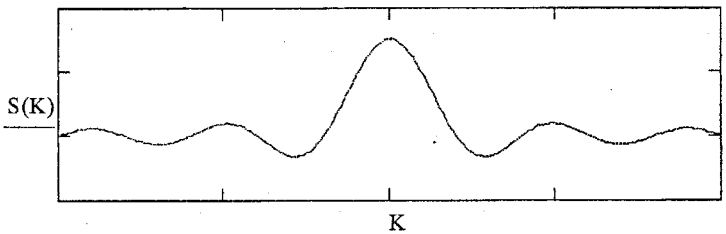
$$x(t) := \sum_{k=0}^{10} A0 \cdot (\alpha \cdot k \cdot \Delta t)^2 \cdot \frac{\sin[\omega c(\Delta t) \cdot (t - k \cdot \Delta t)]}{\omega c(\Delta t) \cdot (t - k \cdot \Delta t)}$$



Визначаємо спектральну щільність сигналу

$K := -10, -9.9..10 \qquad \tau := 0.1$

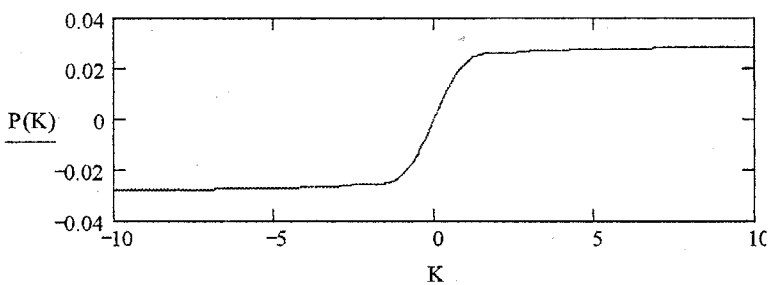
$$S(K) := A0 \cdot \tau \cdot \left[\frac{\sin\left[\frac{\omega c(\Delta t) \cdot (K \cdot \Delta t)}{2}\right]}{\frac{\omega c(\Delta t) \cdot (K \cdot \Delta t)}{2}} \right]$$



Обчислюємо середню потужність сигналу

$T := 1$

$$P(K) := \frac{1}{\pi \cdot T} \cdot \int_0^K (|S(K)|)^2 \, dK$$



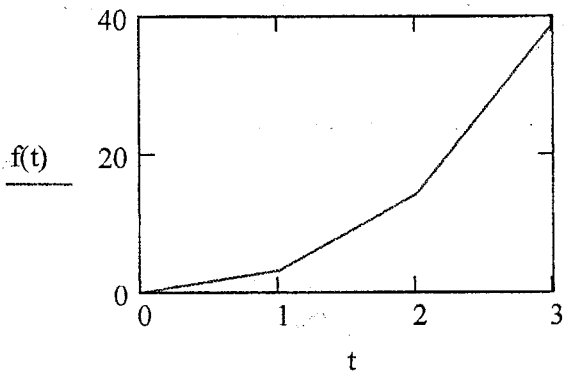
Обчислюємо дисперсію приведеної похибки

$$D(\omega c) := \frac{\frac{1}{\pi \cdot T} \cdot \int_0^{10} (|S(K)|)^2 \, dK}{(75.798)^2} \qquad D(\omega c) = 4.885 \times 10^{-6}$$

2.Знайдемо період квантування T_k на основі критерію допустимого відхилення.

Побудуємо графік функції заданого сигналу

$$f(t) := t + t^2 + t^3 \qquad t := 0..3$$



Визначаємо експеременталько похибку апроксимації

$$sigma := 0.04 \qquad t := 0 \qquad t1 := 0 \qquad t2 := 3$$

Визначаємо інтервал квантування

$$dt := \sqrt{\frac{8 \cdot sigma}{\max\left(\frac{d^2}{dt^2} f(t)\right)}} \qquad dt = 0.4$$

Визначаємо значення апроксимованої функції в вузлах апроксимації

$$i := t1 .. \frac{t2}{dt} \qquad array_i := f(i \cdot dt)$$

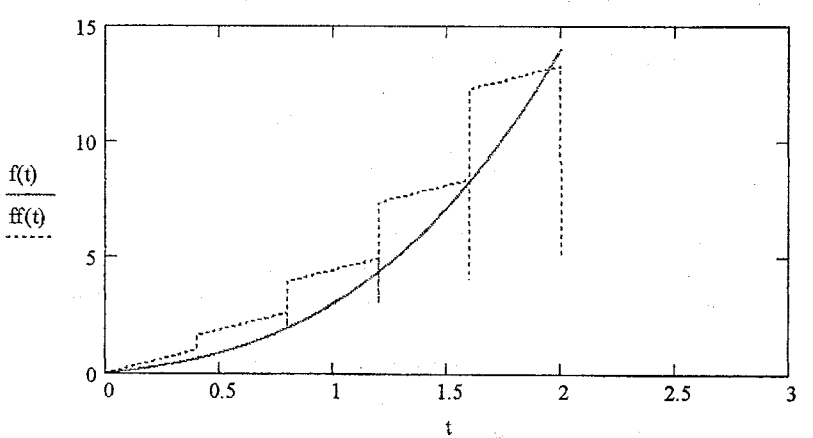
Визначаємо функцію апроксимації

$$ff(t) := \left|\begin{array}{l} \text{count} \leftarrow 0 \\ \text{for } i \in t1, dt .. (t2) \\ \quad \left|\begin{array}{l} \text{if } (i < t) \wedge (t < i + dt) \\ \quad \left|\begin{array}{l} ti \leftarrow i \\ \text{count2} \leftarrow \text{count} \end{array}\right. \\ \text{count} \leftarrow \text{count} + 1 \end{array}\right. \\ \text{index1} \leftarrow \text{count2} \\ \text{index2} \leftarrow \text{index1} + 1 \\ tii \leftarrow ti + dt \\ \text{interm} \leftarrow \text{array}_{\text{index1}} + \dfrac{t - t1}{tii - ti} \cdot \left(\text{array}_{\text{index2}} - \text{array}_{\text{index1}}\right) \\ \text{interm} \end{array}\right.$$

$$\text{array} = \begin{pmatrix} 0 \\ 0.624 \\ 1.952 \\ 4.368 \\ 8.256 \\ 14 \\ 21.984 \\ 32.592 \end{pmatrix}$$

Будуємо функцію апроксимації

$$t := 0, 0.001 .. 2$$



2.3 Дослідження випадкових процесів

2.3.1 Лінійні операції над випадковими функціями

Під *випадковим* (стохастичним) процесом розуміють таку випадкову функцію часу $U(t)$, значення якої в кожен момент часу випадкові. Конкретний вигляд випадкового процесу, зареєстрований у деякому досліді, називають *реалізацією випадкового процесу*. Дані, які характеризують всю множину можливих реалізацій називаються *ансамблем*.

Основними ознаками, за якими класифікують випадкові процеси є: простір станів, часовий параметр та статичні залежності між випадковими величинами $U(t_i)$ в різні моменти часу t_i.

Простором станів називають множину можливих значень випадкової величини $U(t_i)$. Випадковий процес у якому множини станів складають континуум, а зміна станів можлива в будь-які моменти часу, називають *неперервним випадковим процесом*. Якщо зміна станів допускається лише в кінцевому чи поточному числі моментів часу, то говорять про неперервну випадкову величину.

В відповідності до визначення випадковий процес $U(t)$ може бути описаний системою N звичайно залежних випадкових величин $U_1 = U(t_1),...,U_n - U(t_n)$, взятих в різні моменти часу $t_1,t_2,...,t_n$. При необмеженому збільшенні числа N така система еквівалентна випадковому процесу, що розглядається.

В більшості випадків для характеристики випадкових процесів використовують моментні функції перших двох порядків: математичне сподівання, дисперсія, а також кореляційну функцію

$$m[U(t_1)] - M[U(t_1)] = \int_{-\infty}^{\infty} U_1 P_1\left(\frac{U_1}{t_1}\right) dU_1, \qquad (2.44)$$

де $P_1(U_1;t_1)$ - одномірна щільність імовірності або одновимірна функція розподілення випадкового процесу. *Фізико-математичне сподівання* виражає значення сукупності вибірок випадкового процесу у визначений момент часу t_1.

Дисперсія – це математичне сподівання квадрата відхилення величини $U(t_1)$ від математичного сподівання в визначений момент часу t_1.

Дисперсія виражається формулою

$$D[U(t_1)] = M\left\{\left[U(t_1) - mU(t_1)\right]^2\right\} = \int_{-\infty}^{\infty} \left(U_1 - M\left[U(t_1)\right]^2\right) P(U_1,t_1) dU_1. \qquad (2.45)$$

Вона виражає розкид значення випадкової величини навколо математичного сподівання. Корінь квадратний з дисперсії прийнято називати *середнім квадратичним відхиленням* випадкової величини

$$G^2[U(t_1)] = M\left[U^2(t_1)\right] = \int_{-\infty}^{\infty} U^2 P_1(U_1,t_1)dU_1. \qquad (2.46)$$

Фізично початковий момент другого порядку є *повна середня потужність* випадкової величини.

Випадкові процеси можуть мати однакові математичні сподівання й дисперсію, але різко відрізняються за швидкістю зміни своїх значень у часі рис 2.18.

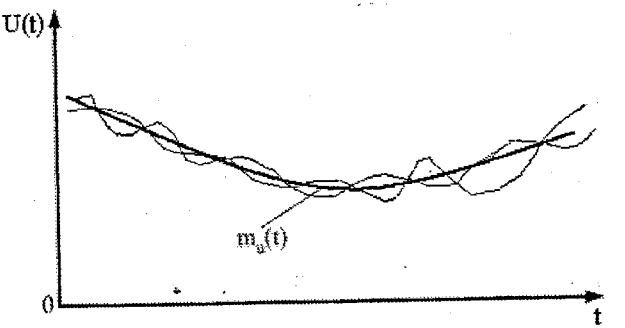


Рисунок 2.18 – Математичне сподівання для різних процесів

Тому для оцінювання ступеня статичної залежності миттєвих значень процесу $U(t)$ в будь-які моменти часу t_1 та t_2 використовується випадкова функція аргументів $R_u(t_1,t_2)$, яка називається *автокореляційною* або просто *кореляційною* функцією.

При конкретних аргументах t_1 та t_2 вона дорівнює кореляційному моменту значень процесу $U(t_1)$ та $U(t_2)$

$$R_u(t_1,t_2) = m_1[U(t_1),U(t_2)]. \qquad (2.47)$$

Двовимірним законом розподілу випадкової функції $X(t)$ називається закон розподілу $f(x_1,x_2,t_1,t_2)$ системи двох випадкових розмірів $X(t_1)$ та $X(t_2)$, що є значеннями випадкової функції для різних значень аргументів $t = t_1$ та $t = t_2$.

Математичним сподіванням випадкової функції $X(t)$ називається невипадкова функція $m_x(t)$, що при кожному даному значенні аргумента дорівнює математичному сподіванню значення випадкової функції при тому ж значенні аргумента $m_x(t) = M[X(t)]$.

Кореляційною функцією випадкової функції $X(t)$ називається невипадкова функція двох аргументів $K_x(t_1,t_2)$, що при кожній парі значень t_1 та t_2 дорівнює кореляційному моменту відповідних значень випадкової функції

$$K_x(t_1,t_2)=M\left[X(t_1),X(t_2)\right], \tag{2.48}$$

де $\overset{\circ}{X}(t)=X(t)-m_x(t)$ - центрована випадкова функція.

При $t_1=t_2=t$ кореляційна функція $K_x(t_1,t_2)$ перетворюється в дисперсію випадкової функції $X(t)$, тобто

$$K_x(t_1,t_2)=D_x(t)=\sigma_x^2(t). \tag{2.49}$$

Нормованою кореляційною функцією випадкової функції $X(t)$ називається функція

$$r_x(t_1,t_2)=\frac{K_x(t_1,t_2)}{\sigma_x(t_1)\sigma_y(t_2)}. \tag{2.50}$$

Взаємною кореляційною функцією двох випадкових функцій $X(t)$ та $Y(t)$ називається функція двох аргументів $R_{xy}(t_1,t_2)$, яка при кожній довільно обраній парі їхніх значень дорівнює кореляційному моменту відповідних значень $X(t_1)$ та $Y(t_2)$ цих випадкових функцій

$$R_{xy}(t_1,t_2)=M\left[X(t_1),Y(t_2)\right]. \tag{2.51}$$

Нормованою взаємною кореляційною функцією двох випадкових функцій $X(t)$ та $Y(t)$ називається функція

$$r_{xy}(t_1,t_2)=\frac{R_{xy}(t_1,t_2)}{\sigma_x(t_1)\sigma_y(t_2)}. \tag{2.52}$$

Випадкові функції $X(t)$ та $Y(t)$ називаються *некаліброваними*, якщо

$$R_{xy}(t_1,t_2)\equiv 0. \tag{2.53}$$

Канонічним розкладанням випадкової функції $X(t)$ називається представлення її у вигляді

$$X(t)=m_x(t)+\sum_{k=1}^{n}X_k\varphi_k(t), \tag{2.54}$$

де X_k, $(k=1,2,...,n)$ - центровані некорельовані випадкові розміри з дисперсіями D_k; $\varphi_k(t)$ - невипадкові функції.

2.3.2 Стаціонарні випадкові функції

Стаціонарні випадкові процеси – це процеси, що протікають у часі однорідно, мають вигляд неперервних випадкових коливань навколо середнього значення $(-,+)$.

Якщо математичне очікування, дисперсія, середнє квадратичне відхилення та кореляція є постійними, то такі процеси – *стаціонарні*.

Якщо ми маємо випадкові процеси, що не витримують таких умов, але на якомусь визначеному інтервалі відхиленням даних параметрів від константи можна знехтувати, то такий процес називають *квазістаціонарним*.

В будь-якій динамічній системі випадковий процес починається з так званого "перехідного" процесу і потім переходить в установлений режим, який з деяким наближенням можна вважати стаціонарним. Потрібно сказати, що стаціонарні випадкові процеси неперервні в часі, а значить не мають ні початку ні кінця. Відомо два поняття: стаціонарність в обмеженому розумінні і стаціонарність у широкому.

Під стаціонарними процесами у вузькому смислі розуміють випадкові процеси, для яких функції розподілу щільності імовірності $\omega_n(x_1,t_1;x_2,t_2;...;x_n,t_n)$ вільного порядку n не змінюється при будь-якому зсуві всієї групи точок $t_1,t_2,...,t_n$ повздовж осі часу

$$\omega_n(x_1,t_1;x_2,t_2;...;x_n,t_n)=\omega_n(x_1,t_1+\tau;x_2,t_2+\tau;...;x_n,t_n+\tau). \quad (2.55)$$

З наведеного визначення можна сказати, що для стаціонарних процесів:

а) одновимірна функція розподілу щільності імовірності не залежить від часу

$$\omega_1(x,t_1)=\omega_1(x,t_1+\tau)=\omega_1(x);$$

б) двовимірна функція розподілу щільності імовірності залежить тільки від різниці часу $t_2-t_1=\tau_1$

$$\omega_2(x_1,t_1;x_2,t_2)=\omega_2(x_1,x_2;t_2-t_1)=\omega_2(x_1,x_2,\tau);$$

в) тривимірна функція розподілу щільності імовірності залежить тільки від двох різниць часу $t_2-t_1=\tau_1$ та $t_3-t_1=\tau_2$

$$\omega_3(x_1,t_1;x_2,t_2;x_3,t_3)=\omega_3(x_1,x_2,x_3;t_2-t_1;t_3-t_1)=\omega_3(x_1,x_2,x_3,\tau).$$

Оскільки математичне сподівання і дисперсія виражаються через одновимірну функцію розподілу щільності імовірності, виходячи з чого можна сказати, що для стаціонарного процесу математичне сподівання й дисперсія не залежать від часу. Унаслідок залежності двовимірної функції розподілу тільки від різниці часу $t_2-t_1=\tau_1$, кореляційна функція стаціонарного процесу також залежить тільки від різниці часу τ.

Стаціонарною випадковою функцією в широкому смислі називається така випадкова функція $X(t)$, математичне сподівання якої постійне, а кореляційна функція залежить тільки від різниці аргументів, тобто

$$m_x(t)=const,$$
$$K_x(t_1,t_2)=k_x(\tau), \quad (2.56)$$

де $\tau=t_2-t_1$.

Дисперсія стаціонарної випадкової функції постійна

$$D_x(t)=K_x(t_1,t_2)=k_x(0)=const. \quad (2.57)$$

Нормована кореляційна функція $\rho_x(\tau)$ стаціонарної випадкової функції $X(t)$ має вигляд

$$\rho_x(\tau)=\frac{k_x(\tau)}{D_x}=\frac{k_x(\tau)}{k_x(0)}. \qquad (2.58)$$

Спектральне розкладання

$$X(t)=m_x+\sum_{k=0}^{\infty}(Y_k\cos\omega_k t+Z_k\sin\omega_k t), \qquad (2.59)$$

де Y_k, Z_k, $(k=0,1,2,...,n)$ - центровані некорельовані випадкові розміри.

Спектральна щільність будь-якої стаціонарної випадкової функції є невід'ємною функцією ω.

Спектральна щільність $S_x(\omega)$ і кореляційна функція $k_x(\tau)$ пов'язані перетворенням Фур'є. У дійсній формі вони мають вигляд

$$S_x(\omega)=\frac{2}{\pi}\int_0^{\infty}k_x(\tau)\cos\omega\tau d\tau,$$
$$k_x(\tau)=\int_0^{\infty}S_x(\omega)\cos\omega\tau d\omega, \qquad (2.60)$$

приймаючи, що $\tau=0$, $k_x(0)=D_x$ отримаємо

$$D_x=\int_0^{\infty}S_x(\omega)d\omega. \qquad (2.61)$$

Нормованою спектральною щільністю $\sigma_x(\omega)$ називається відношення спектральної щільності до дисперсії випадкової функції

$$\sigma_x(\omega)=\frac{S_x(\omega)}{D_x}. \qquad (2.62)$$

2.3.3 Лабораторна робота №3

Мета роботи. Ознайомитися з основними характеристиками випадкового процесу як моделі сигналу, провести дослідження параметрів випадкових процесів, набути навиків аналізу випадкових процесів.

Хід роботи

1. Згідно з варіантом таблиці 2.3 вибрати функції $X(t)$ та характеристики, що описують даний процес, побудувати графіки стаціонарного процесу;
2. Визначити значення кореляційної функції та отримати графік залежності;
3. Визначити значення спектральної щільність досліджуваної функції;
4. Побудувати графік залежності спектральної щільності досліджуваної функції від зміни частоти процесу.

Таблиця 2.3 – Варіанти завдань

вар.	$X(t)$	A	ω	τ	t	a
1	$\frac{\cos\omega t}{1+\cos^2 2\omega t}$	2	1000	$\frac{\pi}{3}$	0,1	0.1
2	$\frac{\sin\omega t}{1+\cos^2 2\omega t}$	3	2000	$\frac{\pi}{4}$	1	0.2
3	$\frac{\sin 2\omega t}{\sin\omega t+\cos 2\omega t}$	3	3000	$\frac{\pi}{5}$	0,2	0.3
4	$\frac{\sin 3\omega t}{\lvert\sin\omega t\rvert+\lvert\cos\omega t\rvert}$	4	2000	$\frac{\pi}{6}$	0,5	0.4
5	$\cos e^{\lvert\sin 3\omega t\rvert}$	2	1500	$\frac{\pi}{7}$	0,6	0.5
6	$\cos\omega t\lvert\sin\omega t\rvert$	5	2500	$\frac{\pi}{8}$	0,3	0.6
7	$\operatorname{ctg}\left(\cos\frac{1}{2}\omega t\right)$	6	3400	$\frac{\pi}{9}$	1,5	0.7
8	$e^{\sin\frac{1}{3}\omega t}$	5	800	$\frac{\pi}{10}$	0,8	0.8
9	$\sin\left(\frac{1}{2}\omega t\right)^2$	3	1600	$\frac{\pi}{16}$	1,2	0.1
10	$\sin\left(\omega\sqrt{1+t^2}\right)$	4	2300	$\frac{\pi}{3}$	0,5	0.2
11	$\cos\left(\omega\sqrt{1+t^2}\right)$	5	3500	$\frac{\pi}{4}$	0,6	0.3
12	$\sin(\cos\omega t)$	6	4200	$\frac{\pi}{5}$	0,4	0.4
14	$\cos(\sin\omega t)$	8	2000	$\frac{\pi}{6}$	0,2	0.6

Контрольні запитання

1. Дайте пояснення математичному сподіванню та дисперсії випадкового процесу.
2. Що таке середнє квадратичне значення та середнє квадратичне відхилення випадкового процесу?
3. Що таке спектральна щільність випадкового процесу?
4. Що таке кореляційна функція випадкового процесу?
5. Дайте пояснення стаціонарності випадкового процесу.
6. Які основні властивості кореляційної функції стаціонарного процесу?

7. Які випадкові процеси називаються ергодичними?
8. Який випадковий процес називається білим шумом?

Приклад. Знаходження кореляційної функції та спектральної щільності для стаціонарного випадкового сигналу.

В інформаційних системах часто зустрічаємося з випадковими процесами, які протікають в часі приблизно однаково. Ці процеси мають вигляд неперервних випадкових коливань навколо деякого середнього значення, причому ні середнє значення, ні характер цих коливань в часі майже не змінюється. Такі процеси називають стаціонарними.

Стаціонарні процеси не мають ні початку ні кінця, але на практиці вважають стаціонарними ті, що на протязі певного часу постійні.

Під стаціонарними процесами розуміють випадкові процеси, для яких функція розподілу щільності ймовірності $w(x_1,t_1;x_2,t_2,....,x_n,t_n)$ будь-якого порядку n не змінюються при будь-якому зсуві групи точок $t_1,t_2...t_n$ вздовж осі часу для будь-яких n та t_0+t.

1. Побудуємо графіки стаціонарного процесу

$A := 1$ – амплітуда

$\omega_0 := 1000$ – частота

$\tau := \frac{\pi}{8}$ – початковий зсув

$\phi := 0$ – початковий зсув фази

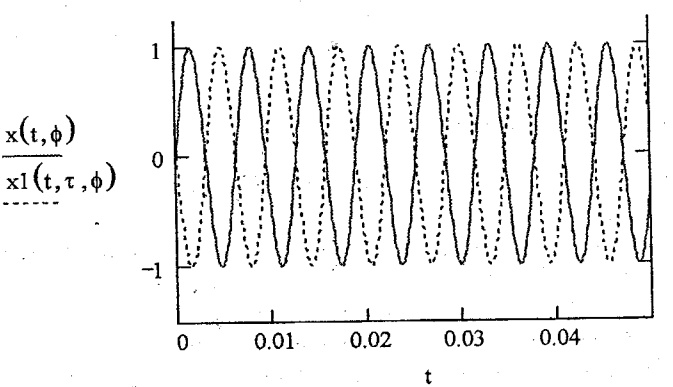
$\omega 1(\phi) := \frac{1}{2\pi}$

$t := 0, 0.0001 .. 0.1$

$x(t,\phi) := A \cdot \sin(\omega_0 \cdot t + \phi)$

$j := \sqrt{-1}$

$x1(t,\tau,\phi) := A \cdot \sin(\omega_0 \cdot t + \omega_0 \cdot \tau + \phi)$



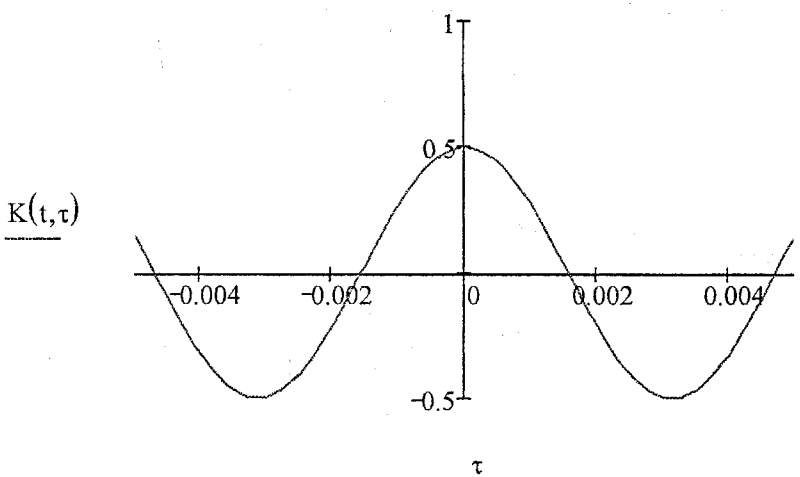
Кореляційний інтеграл має вигляд інтегрування двох функцій за фазою сигнала в межах одного повного періоду 2р. При цьому функції перемножуються x(t)·x(t+t).

$$K(t,\tau) := \int_{-\pi}^{\pi} x(t,\phi) \cdot x1(t,\tau,\phi) \cdot \omega 1(\phi)\, d\phi$$

Аналіз показує, що при збільшенні t (де $t = t_2 - t_1$) залежність між функціями спадає, і при $t >> 1$ ці величини стають незалежними. Таким чином кореляційна функція наближається до квадрата математичного сподівання випадкового стаціонарного процеса.

2. Отримаємо графік кореляційної функції стаціонарного процесу

$t := 10$ $\tau := -0.005, -0.0049 .. 0.005$



3. Визначимо спектральну щільність процеса.

Для визначення спектральної щільності скористаємося середньою потужністю процеса. Шляхом усереднення за ансамблем реалізації процесу скористаємося таким виразом для енергетичної характеристики .

$a := 0.01$ -межі зміни інтегрування

$t := 10$ -час, протягом якого досліджувилась поведінка функції

$\tau := 0.01$ -різниця часу

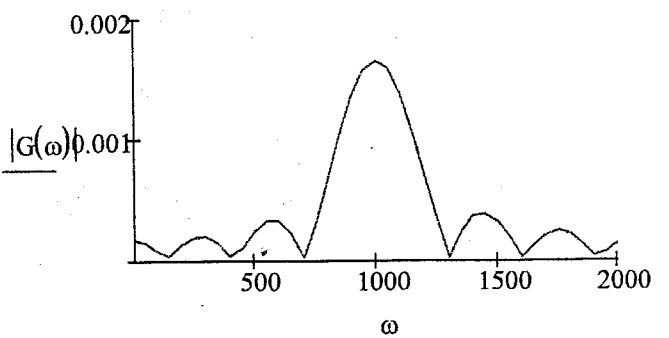
$\omega := 0$ -початкова частота

$$G(\omega) := \frac{1}{\pi} \cdot \int_{-a}^{a} K(t,\tau) \cdot \left(\sin(\omega \cdot \tau) + \cos(\omega \cdot \tau)\right) d\tau$$

$$G(\omega) = -1.732 \times 10^{-4}$$

3. Побудуємо графік залежності спектральної щільності досліджуваної функції від зміни частоти процесу.

$\omega := 0, 50 .. 2000$ -межі, в яких досліджуємо функцію



Даний графік є по суті спектром досліджуваного процесу.

2. 4 Методи обробки зображень

2.4.1 Що таке колір?

Колір являє собою єдність трьох сутностей: світла, що спостерігається, об'єкта й спостерігача. Людське око сприймає колір відповідно до довжини хвиль електромагнітних випромінювань, що складають видиме світло. Колір, що містить повний спектр у рівних пропорціях, сприймається як чистий білий. Відсутність світла око сприймає як чорне світло. Око сприймає колір як світлові хвилі різної довжини, що відповідають червоному, зеленому та синьому, які відбиваються чи поглинаються об'єктом, що спостерігається.

Світло хвилі довжиною, що відповідає червоному, зеленому та синьому кольорам складають основу всіх кольорів у природі. Саме тому червоний, зелений і синій колір часто називають *основними* чи *первинними* кольорами. Усі інші кольори спектра є комбінацією хвиль різної довжини.

При накладанні трьох первинних кольорів утворяться вторинні кольори: блакитний, пурпурний та жовтий. Первинні й вторинні кольори називають *взаємодоповнюючими* (чи *комплементарними*) кольорами, тому що будь-яка пара вторинних кольорів при змішувані утворять первинний колір.

2.4.2 Колірна схема RGB

У 1931 році міжнародна освітлювальна комісія (МОК) за результатами експериментальних робіт В. Райта й І. Гільда затвердила *колірну схему RGB*.

Міжнародна комісія вибрала за основні кольори системи *RGB* кольори однорідних випромінювань хвиль довжиною $\lambda_R = 700$ нм., $\lambda_G = 546{,}1$ нм., $\lambda_B = 435{,}5$ нм.

Визначені експериментальним шляхом Райтом і Гільдом координати кривих додавання перераховані на координати для нових основних кольорів, запропонованих МОК. Незважаючи на те, що досліди Райта й Гільда проводилися за принципово різними схемами різними спостерігачами, похибки результатів вимірювання виявилися незначними. Це дозволило МОК прийняти усереднені значення питомих координат кольору однорідних випромінювань, отриманих із дослідів Райта та Гільда як колірні характеристики середнього людського органа зору (стандартного спостерігача МОК). У таблиці стандартного спостерігача МОК крім питомих координат кольору внесені і координати кольоровості. Один рядок такої таблиці приводимо нижче в таблиці 2.4.

Для багатьох однорідних випромінювань у системі *RGB* одна з трьох питомих координат кольору негативна. При експериментальних дослі-

дженнях кольорів вибираються спостерігачі, у яких колірні характеристики зору практично збігаються з даними стандартами спостерігача МОК.

Таблиця 2.4

$\bar{r}_\lambda$	$\bar{g}_\lambda$	$\bar{b}_\lambda$	нм	r_λ	g_λ	b_λ
-0,0121	0,0068	0,3167	450	-0,0390	0,0218	1,0172

У системі RGB колірне керування кольору має вигляд

$$F \equiv r'R + g'G + b'B,$$

де R, G, B – основні кольори системи; F – колір випромінювання, що за кількісною і якісною характеристиками визначається через основні кольори системи; r', g', b' - координати кольору; $r'R$, $g'G$, $b'B$ - компоненти кольору.

Координати кольоровості в цій системі

$$r = \frac{r'}{\sigma};\ g = \frac{g'}{\sigma};\ b = \frac{b'}{\sigma},$$

де $\sigma = r' + g' + b'$.

Очевидно, що

$$r + g + b = 1.$$

Рівність суми координат кольоровості одиниці дозволяє побудувати просту діаграму кольоровості (колірні графіки), використовуючи відому з геометрії властивість рівностороннього трикутника, у якого сума відстаней від будь-якої точки до його сторін дорівнює висоті трикутника як показано на рис. 2.19. Це не єдиний спосіб графічного позначення кольору. Можна побудувати цю діаграму і в іншій системі координат, зробивши відповідні розрахунки.

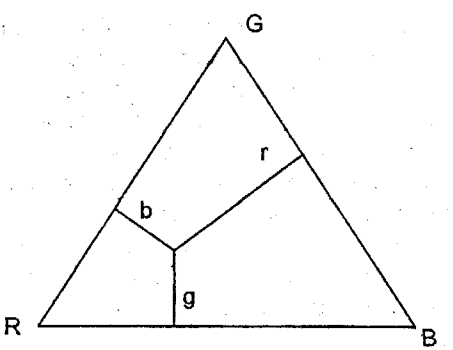


Рисунок 2.19– Діаграма кольоровості

2.4.3 Обробка кольорових (RGB) зображень

Обробка зображення виконується за допомогою команди **Picture** із меню панелі **Insert**, в комірці для формул указуємо назву рядкової змінної `file1`.

Зчитування зображення виконується за допомогою рядкової змінної `file1` та імені файлу

```
file1:="Variants\ForExampl.bmp".
```

Команда READBMP зчитує бітовий масив BMP-зображення в матрицю A. При цьому кожен піксел зображення записується як елемент матриці в діапазоні від 0 (чорний) до 255 (білий).

```
A:=READBMP(file1)
```

За допомогою команди **Picture** із меню панелі **Insert** вставляємо зображення, яке знаходиться в масиві A, в комірці для формул указуємо назву масиву, у даному випадку A).

Функція `WRITEBMP` записує у файл матрицю відтінків сірого.

```
WRITEBMP("Gray.bmp"):=A
```

Функція `READBMP` у масиві послідовно записує один за одним червону, зелену та синю складові кольору кожного піксела BMP-файлу. Якщо за допомогою команди **Picture** із меню панелі **Insert** вивести масив `RGBmatrix`, то ми побачимо 3 сірих зображення відповідно для червоного, зеленого, синього кольорів.

```
RGBmatrix:=READRGB(file1)
```

За допомогою функції `rows` та `cols` визначаємо кількість рядків та стовпців *RGB*-матриці. Отриману кількість стовпців ділимо на 3 щоб визначити кількість стовпців, яка приходиться на кожний основний колір

```
rydki:=rows(RGBmatrix)
rydki = 40
```

$$stolbci := \frac{cols(RGBmatrix)}{3}$$

```
stolbci = 40.
```

Для того щоб вивести кольорове зображення потрібно:

- визначити кількість рядків та стовпців у матриці `RGBmatrix` (дивись вище);
- утворити нульову матрицю N того ж розміру, що і матриці основних кольорів:

$i := 0..rydki - 1$

$j := 0..stolbci - 1$

$N_{i,j} := 0$

- розділити складові за допомогою функції submatrix:

стовпці від 0 до rydki-1 для червоної складової

red:=submatrix(RGBmatrix,0,rydki-1,0,stolbci-1)

стовпці від 0 до $2 \cdot$ rydki-1 для зеленої складової

green:=submatrix(RGBmatrix,0,rydki-1,stolbci,$2 \cdot$ stolbci-1)

стовпці від 0 до rydki-1 для синьої складової

blue:=submatrix(RGBmatrix,0,rydki-1,$2 \cdot$ stolbci,$3 \cdot$ stolbci-1)

Відображаємо складові на екрані за допомогою команди **Picture** із меню панелі **Insert:**

- тільки червона складова в комірках – `red, N, N`

- тільки зелена складова в комірках – `N, green, N`

- тільки синя складова в комірках – `N, N, blue`

- Кольорове зображення – `red, green, blue`

2.4.4 Лабораторна робота №4

Мета роботи. Ознайомитися з методами обробки зображень, вивчити команди, які використовуються для роботи із зображеннями в програмному математичному пакеті MathCAD.

Хід роботи

1. Згідно з варіантом оберіть своє зображення;
2. Зчитайте матриці основних кольорів свого зображення та створіть кольорове *RGB* зображення накладанням як це показано в 2.4.1;
3. Створіть ефект зміни яскравості кольорового зображення згідно 2.4.2. Змінюючи значення масиву M (збільшуючи чи зменшуючи, міняючи знак) прослідкуйте за зміною яскравості зображення;
4. Створіть ефект зміни яскравості зображення по вертикалі згідно з 2.4.3. Змінюючи крок зміни $k \leftarrow k + 0,7$ (збільшуючи чи зменшу-

ючи, міняючи знак) або змінивши знак k в формулі $g_{i,j} \leftarrow g_{i,j} + k$ прослідкуйте за змінами зображення;

5. Створіть ефект накладання зображень 2.4.4;
6. Зробити висновки по роботі.

Приклад . Обробка кольорових (*RGB*) зображень в MathCAD.

1.Зчитування матриць основних кольорів та створення кольорового *RGB* зображення їх накладанням:

- зчитуємо в масив *R, G, B* основні кольори зображення

R:=READBMP("Variants\R8.bmp")

G:=READBMP("Variants\G8.bmp")

B:=READBMP("Variants\B8.bmp")

- визначаємо кількість рядків та стовпців в отриманих 3-х масивах (масиви однакові)

$$rydki := rows(R)$$

$$Rydki = 235$$

$$stolbci := cols(R)$$

$$stolbci = 235$$

- створюємо матрицю N того ж розміру

$$i := 0..rydki - 1$$

$$j := 0..stolbci - 1$$

$$N_{i,j} := 0$$

$$M_{i,j} := 80$$

- за допомогою команди **Picture** із меню панелі **Insert** вставляємо зображення, які знаходяться в масивах *R, G, B,* в комірці для формул вказуємо:
  - тільки червона складова в комірках: *R, N, N*;
  - тільки зелена складова в комірках: *N, G, N*;
  - тільки синя складова в комірках: *N, N, B*.

Виводимо зображення, вводячи в комірки області зображення *R,G, B*.

2. Створення ефекту зміни яскравості кольорового зображення:

- створення масиву яскравості значень M, на величину яких буде змінена яскравість зображення

$$i := 0..rydki - 1$$

$$j := 0..stolbci - 1$$

$$M_{i,j} := 80$$

- збільшення яскравості зображення додаванням масиву M, утворення нових масивів R_1, G_1, B_1 (більш яскравих)

$$R_1 := R + M$$
$$G_1 := G + M$$
$$B_1 := B + M$$

- створимо функцію, яка перевіряє "освітлені" масиви на елементи значення яких після додавання стали більші ніж 255 (білий колір) і змінює їх значення на 255

$$f_1(m) := \left\| \begin{array}{l} m \leftarrow 255 \quad if \quad m > 255 \\ m \leftarrow 0 \quad if \quad m < 0 \\ m \end{array} \right.$$

- перевірка "освітлених" масивів за допомогою функції f_1 на елементи значення яких вийшли за межі 255 (білого кольору)

$$R_{1_{i,j}} := f_1(R_{1_{i,j}})$$
$$G_{1_{i,j}} := f_1(G_{1_{i,j}})$$
$$B_{1_{i,j}} := f_1(B_{1_{i,j}})$$

- виведення "освітлених" масивів основних кольорів та кольорового зображення (5.1).

3. Створення ефекту зміни яскравості зображення по вертикалі:
   - створення функції для зміни яскравості зображення по вертикалі. Дана функція збільшує кожен новий рядок масиву на k, чим досягається збільшення його яскравості в порівнянні з попереднім

$$f_2(1) := \left\| \begin{array}{l} \\ \\ \\ k \leftarrow 0 \\ g \leftarrow 1 \\ for \quad i \in 0..rydki - 1 \\ \quad \left\| \begin{array}{l} for \quad j \in 0..stolbci - 1 \\ g_{i,j} \leftarrow g_{i,j} + k \\ k \leftarrow k + 0,7 \end{array} \right. \\ g \end{array} \right.$$

   - зміна яскравості зображення за допомогою функції f_2 по вертикалі та виведення отриманих зображень на екран (5.1)

$$R_2 := f_2(R)$$
$$R_{2_{i,j}} := f_1(R_{2_{i,j}})$$
$$G_2 := f_2(G)$$
$$G_{2_{i,j}} := f_1(G_{2_{i,j}})$$
$$B_2 := f_2(B)$$
$$B_{2_{i,j}} := f_1(B_{2_{i,j}})$$

4. Ефект накладання зображень:
   - зчитування іншого зображення розкладеного на 3 основні кольори та виведення його на екран

$$r := READBMP("Variants \ R7.bmp")$$
$$g := READBMP("Variants \ G7.bmp")$$
$$b := READBMP("Variants \ B7.bmp")$$
$$rydki := rows(r) \qquad rydki = 235$$

$$stolbci := cols(r) \qquad stolbci = 235$$
$$i := 0..rydki - 1$$
$$j := 0..stolbci - 1$$
$$N_{1_{i,j}} := 0$$

   - створення масивів основних кольорів в яких накладено 2 різні зображення

$$R_{rez} := \frac{R_2 + r}{2}$$
$$G_{rez} := \frac{G_2 + g}{2}$$
$$B_{rez} := \frac{B_2 + b}{2}$$

виведення на екран отриманого зображення.

2.5 Завадостійке кодування

Під завадостійкими кодами розуміють коди, які дозволяють виявляти або виявляти та виправляти помилки, які виникають у результаті впливу завад.

Завадостійкість кодування забезпечується за рахунок введення надлишковості в кодові комбінації, тобто за рахунок того, що не всі символи в кодових комбінаціях використовуються для передачі інформації.

У блокових кодах кожне повідомлення порівнюється з кодовою комбінацією (блоком) із визначеної кількості сигналів. Блоки кодуються та декодуються окремо.

Рівномірні – коли довжина кодової комбінації n = const. Нерівномірні – n = Var

В неперервних кодах уведення надлишковості в послідовності вхідних символів здійснюється без розбивання його на окремі блоки.

Як блокові, так і неперервні коди в залежності від методів внесення надлишковості поділяються на подільні та неподільні.

В подільних кодах чітко поділяється роль окремих символів. Одні символи є інформаційними, інші – перевірочними.

Неподільні коди не мають чіткого поділу кодової комбінації на інформаційні та перевірочні символи. Цей клас кодів поки що нечисленний.

Подільні блочні коди поділяються на несистематичні, які будуються таким чином, що перевірочні символи визначаються як сума підблоків довжиною L, за якими поділяється блок інформаційних символів.

Систематичні коди – в яких перевірочні символи визначаються в результаті проведення лінійних операцій над визначеними інформаційними символами.

Для випадку двійкових кодів кожен перевірочний символ вибирається таким, щоб його сума за модулем два з визначеними інформаційними символами стала рівною нулю.

2.5.1 Основні принципи завадостійкого кодування

Кодове посилання має:

- значимість коду n;
- основу m;
- вагу кодової комбінації w (кількість одиниць у кодовій комбінації).

$U_{сер}$ 100110001; $n = 9$; $w = 4$.

Ступінь відмінності будь-яких двох кодових комбінацій даного коду характеризується так званою відстанню між кодами d.

Вона виражається числом позицій або символів, у яких комбінації відрізняються одна від одної, і визначається як вага суми за модулем 2 даних кодових комбінацій.

Наприклад, 100110001
+ 110101001
010011000 .

Нова кодова комбінація має вагу $w = 4$. Відстань між початковими кодовими комбінаціями - $d = 4$.

Помилки, які виникли внаслідок впливу завад, виявляються в тому, що в одному чи декількох розрядах кодової комбінації нулі переходять в 1 та навпаки. В результаті створюється нова помилкова комбінація.

Завадостійкість кодування забезпечується введенням надлишковості в кодових комбінаціях.

n – кількість символів у комбінаціях;

k – кількість інформаційних символів

$$k < n;$$

$N_0 = 2^n$ – кількість можливих кодових комбінацій;

$N = 2^k$ – кількість комбінацій, які використовуються для передачі.

Усі комбінації діляться на дві групи;

$N = 2^k$ – дозволені комбінації;

$N_0 - N = 2^n - 2^k$ – заборонені комбінації.

Кожна з N дозволених комбінацій може транспортуватись в нову N_0 можливих комбінацій:

N^*N_0 – кількість варіантів передачі, із них N – варіантів безпомилкові.

$N(N-1)$ – кількість варіантів переходу в інші дозволені комбінації.

$N(N_0-N)$ – кількість варіантів переходу в заборонені комбінації.

Таким чином не всі спотворення можуть бути виявлені.

Доля виявлених помилкових комбінацій складає:

$$\frac{N(N_0 - N)}{N \cdot N_0} = 1 - \frac{N}{N_0} \quad . \tag{2.64}$$

Для використання даного коду в якості виправного множина заборонених кодових комбінацій розбивається на N непересічних підмножин. Кожна з підмножин ставиться у відповідність одній з дозволених комбінацій. Якщо прийнята заборонена комбінація належить підмножині, то враховується, що відбулася передача дозволеної комбінації .

Помилка буде виправлена в тому випадку, коли отримана комбінація дійсно утворена із множини дозволеної комбінації.

Таким чином помилка виправляється в N_0-N випадках, рівних кількості заборонених комбінацій.

Частка виправлених кодових комбінацій

$$\frac{N_0 - N}{N(N_0 - N)} = \frac{1}{N} \quad . \tag{2.65}$$

Приклад: Необхідно побудувати код, який виявляє всі помилки кратністю t та нижче.

Це означає, що потрібно, щоб найменша кодова відстань задовольняла умову

$$D_{min} \geq t+1. \qquad (2.66)$$

Побудуємо код із значністю n=3. Різні комбінації такого коду представлені в табл. 2.5.

Таблиця 2.5

A1	A2	A3	A4	A5	A6	A7	A8
000	001	010	011	100	101	110	111

Матриця відстаней між кодовими комбінаціями має вигляд

	A1	A2	A3	A4	A5	A6	A7	A8
A1		1	1	2	1	2	2	3
A2	1	0	2	1	2	1	3	2
A3	1	2	0	1	2	3	1	2
A4	2	1	1	0	3	2	2	1
A5	1	2	2	3	0	1	1	2
A6	2	1	3	2	1	0	2	1
A7	2	3	1	2	1	2	0	1
A8	3	2	2	1	2	1	1	0

Для того, щоб код забезпечував виявлення однократних помилок, необхідно із усієї множини N_0=8 можливих комбінацій вибрати в якості дозволених такі, відстані між якими були б не менші d=2.

Як видно з матриці комбінації можуть бути такими:

A1=000; A4=011; A6=101; A7=110.

Для виявлення двократних помилок найменша відстань повинна бути d_{min}=3

A1=000; A8=111.

Таким чином, для виявлення помилок $d_{min} < n$ необхідно вибрати A1, A4, A6, A7.

Побудувати код для виявлення та усунення однократних помилок.

При наявності однократних помилок комбінація A1 може перейти в одну з таких заборонених комбінацій: A2, A3, A5, які можна прийняти в якості підмножини заборонених комбінацій A1.

В якості другої дозволеної комбінації вибирається комбінація на відстані d=2.

Для A4= 011. Їй повинна відповідати підмножина заборонених комбінацій

A3=010; A2=001; A8=111.

Однак виникає перехрещення підмножин A2 та A3. При прийомі заборонених сигналів A2 та A3 неможливо однозначно сказати, який був переданий сигнал A1 чи A3.

Якщо прийняти d=3, то залишається тільки підмножина А1 та А8:

– для А8 відповідають заборонені комбінації А4, А6, А7;

– для А1 відповідають заборонені комбінації А2, А3, А5.

В цьому випадку підмножини заборонених комбінацій не пересікаються. Отже однократні помилки при d=3 усуваються всі.

Приклад. Визначити корегувальна здатність коду, який має такі дозволені комбінації: 00000; 10101; 11011.

Корегувальна здатність коду визначається мінімальною кодовою відстанню. Складемо матрицю відстаней між кодовими комбінаціями в таблиці 2.6.

Таблиця 2.6

	00000	01110	10101	11011
00000	0	3	3	4
01110	3	0	4	3
10101	3	3	0	3
11011	4	3	4	0

Мінімальна кодова відстань L_{min}=3. Отже, даний код здатен:

- виявляти двократні помилки;
- виявляти однократні помилки;
- усувати однократні помилки.

В загальному випадку для усунення помилок кратності σ кодова відстань повинна задовольняти умову:

$$D_{min} = 2\sigma+1 \qquad (2.67)$$

для виправлення всіх помилок кратністю не більше σ та одночасного виявлення всіх помилок .

При розгляданні корегувальних кодів ми передбачали задану його значність n.

На практиці коди будуються у зворотному порядку: спочатку вибирається кількість інформаційних символів k, після чого забезпечується необхідна корегувальна здатність за рахунок добавлення надлишковості.

Нехай відомий обмін алфавіту джерела N (кількість комбінацій). Кількість символів

$$K = \log_2 N. \qquad (2.68)$$

Нехай відома повна кількість помилок E, яку необхідно виправити.

Завдання полягає в тому, щоб при заданих N та E визначити значність коду n.

Повне число помилкових комбінацій, які підлягають виправленню:

$$E \cdot 2^k = E \cdot N. \qquad (2.69)$$

Кількість помилкових комбінацій N_0–N

$$E \cdot N =< N_0 - N. \qquad (2.70)$$

Отже $N_0 >= (1+E)N$.

$N =< 2^n/(1+E)$ –формула виражає умову для вибору значності коду n.

При усуненні однократних помилок маємо:

n	22	33	44	55	66	77	88	99
$2^n/(1+n)$	1,33	2	3,2	5,33	9,2	16	28,4	51,2

Якщо необхідно врахувати усунення всіх помилок кратністю від 1 до L, то потрібно:

- врахувати число можливих однократних помилок $E_1 = C_n^1$;
- врахувати число можливих двократних помилок $E_2 = C_n^2$;
- врахувати число можливих L-кратних помилок $E_L = C_n^L$;
- загальна кількість помилок дорівнює $E = \sum_{i=1}^{L} C_n^L$.

При цьому кількість комбінацій приймає вигляд

$$N =< 2^n/(1+\sum C_n^L). \qquad (2.71)$$

Умова (2.71) є нижньою оцінкою для довжини корегувального коду, тобто вона визначає необхідну довжину коду n, що забезпечує виправлення помилок заданої кратності при відомому числу дозволених комбінацій N чи числі інформаційних символів $K = \log_2 N$. Ця ж умова є верхньою оцінкою для N і k , тобто визначає максимально можливе число дозволених комбінацій або інформаційних символів для коду n , що забезпечує виправлення помилок заданої кратності.

Будь-який корегувальний код характеризується рядом показників:

- довжиною n;
- основою m;
- кількістю інформаційних символів k;
- (надмірністю) надлишковістю символів $\rho = n - k$;
- $N_0 = m^n$ – повним числом усіх можливих комбінацій;
- N – числом дозволених комбінацій;
- w – вагою кодової комбінації;
- d – кодовою відстанню.

Але основним показником завадостійкості кода є здатність забезпечувати правильну передачу повідомлень в умовах завад. Використовують імовірні характеристики:

$$\rho_{пр} = 1 - \rho_{пом},$$

$\rho_{пр}$ – імовірність правильного прийому,

$\rho_{пом}$ – імовірність помилкового прийому.

Корегувальна здатність коду забезпечується за рахунок надлишковості, тобто збільшення кодової комбінації.

Коефіцієнт надлишковості:

$$K_{надл} = \rho/n = (n-k)/n,$$

де ρ – кількість надлишкових символів у кодовій комбінації.

Надлишковість є важливою характеристикою коду.

2.5.2 Лабораторна робота №5

Мета роботи. Ознайомитися з методами завадостійкого кодування та засвоїти методику побудови завадостійкого коду за допомогою математичного пакета MathCAD 2000.

Хід роботи

1. На основі виданого варіанта (таблиця 2.7) визначити необхідні параметри завадостійкого коду.
2. Побудувати згідно з варіантом завадостійкий код із відповідною корегувальною здатністю.
3. Обчислити кількість інформаційних символів.
4. Підібрати n, таким чином, щоб отримати стільки дозволених кодових комбінацій, скільки посилань, що потрібно передати (P).
5. Визначити коефіцієнт надмірності.
6. Зробити висновки по роботі.

Контрольні запитання

1. Яка ціль кодування?
2. Дайте пояснення завадостійкому кодуванню.
3. Як визначається кодова відстань?
4. Що таке вектор помилки?
5. Що таке значність і вага кодової комбінації?
6. Який зв'язок корегувальної здатності коду з кодовою відстанню?
7. Які показники оцінюють якість корегувальних кодів?
8. Які основні типи кодів Ви знаєте?

Таблиця 2.7 - Варіанти завдань

Варіант	Кількість посилань P	Кратність помилок, що виявляються t	Кратність помилок, що виправляються σ
1	20	1	1
2	10	2	1
3	8	3	1
4	6	2	2
5	4	4	1
6	2	4	2

Приклад . Знаходження дозволених кодів для завадостійкого кодування, використовуючи пакет MathCAD.

$P := 8$ - кількість посилань;

$k := \frac{\log(P)}{\log(2)}$ - кількість інформаційних символів;

k=3, n=6, - загальна кількість символів (інформаційні плюс надлишкові);

$t := 1$ - кратність помилок, які виявляються;

$\sigma := 1$ - кратність помилок, які виправляються;

$t \geq \sigma$ - умова співвідношення помилок, які виявляються і помилок, які виправляються;

$d := t + \sigma + 1$ - кодова відстань;

1. Функція перетворення десяткового числа N розрядністю n на матрицю - стовпець двійкових бітів

$dig(N,n) :=$ | for $i \in 0..n-1$
| $res_{n-1-i} \leftarrow 1$ if $mod(N,2) = 1$
| $res_{n-1-i} \leftarrow 0$ otherwise
| $N \leftarrow trunc\left(\frac{N}{2}\right)$
| res

2. Функція визначення кодової відстані між десятковими числами N1 та N2 розрядністю n

$codeDistance(N1,N2,n) :=$ | $d \leftarrow 0$
| $M1 \leftarrow dig(N1,n)$
| $M2 \leftarrow dig(N2,n)$
| for $i \in 0..n-1$
| $d \leftarrow d + (M1_i \oplus M2_i)$
| d

Початкові дані:

Розрядність посилання: $n := 6$

Кількість помилок, що виправляються: $e := 1$

3. Загальна кількість різних кодових комбінацій, які можна утворити розрядністю n завадостійкого коду, де e - число помилок визначається:

$$d := 2e + 1$$

$$N0 := \text{trunc}\left(\frac{2^n}{1 + \sum_{i=1}^{e} \text{combin}(n, i)}\right)$$

Для даних значень: $N0 = 9 \quad d = 3$

4. Для наглядності будуємо матрицю відстаней кодових посилань (матриця для n-розрядних посилок):

$$\text{errM}(n) := \left| \begin{array}{l} \text{for } i \in 0..2^n - 1 \\ \quad \text{for } j \in 0..2^n - 1 \\ \quad\quad \text{res}_{i,j} \leftarrow \text{codeDistance}(i, j, n) \\ \text{res} \end{array} \right.$$

5. Функція визначення множини можливих кодових комбінацій, що передаються працює за таким алгоритмом:
- задаємо k - індекс матриці кодів, що передаються;
- першим кодом є код 00..0;
- наступне значення коду, який передається;
- організовуємо цикл, що буде шукати наступні коди;
- якщо кодова відстань не відповідає вимозі - тоді беремо наступний елемент;

- задаємося змінною IsCross, яка є прапорцем для знайденої кодової, що перетинається з множиною попередніх комбінації;
- надаємо тимчасовій змінній значення наступної комбінації та повертаємось на початок циклу.

$errM(n) =$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
0	0	1	1	2	1	2	2	3	1	2	2	3	2	3	3	4	1	2	2	3	2	3	3
1	1	0	2	1	2	1	3	2	2	1	3	2	3	2	4	3	2	1	3	2	3	2	4
2	1	2	0	1	2	3	1	2	2	3	1	2	3	4	2	3	2	3	1	2	3	4	2
3	2	1	1	0	3	2	2	1	3	2	2	1	4	3	3	2	3	2	2	1	4	3	3
4	1	2	2	3	0	1	1	2	2	3	3	4	1	2	2	3	2	3	3	4	1	2	2
5	2	1	3	2	1	0	2	1	3	2	4	3	2	1	3	2	3	2	4	3	2	1	3
6	2	3	1	2	1	2	0	1	3	4	2	3	2	3	1	2	3	4	2	3	2	3	1
7	3	2	2	1	2	1	1	0	4	3	3	2	3	2	2	1	4	3	3	2	3	2	2
8	1	2	2	3	2	3	3	4	0	1	1	2	1	2	2	3	2	3	3	4	3	4	4
9	2	1	3	2	3	2	4	3	1	0	2	1	2	1	3	2	3	2	4	3	4	3	5
10	2	3	1	2	3	4	2	3	1	2	0	1	2	3	1	2	3	4	2	3	4	5	3
11	3	2	2	1	4	3	3	2	2	1	1	0	3	2	2	1	4	3	3	2	5	4	4
12	2	3	3	4	1	2	2	3	1	2	2	3	0	1	1	2	3	4	4	5	2	3	3
13	3	2	4	3	2	1	3	2	2	1	3	2	1	0	2	1	4	3	5	4	3	2	4
14	3	4	2	3	2	3	1	2	2	3	1	2	1	2	0	1	4	5	3	4	3	4	2
15	4	3	3	2	3	2	2	1	3	2	2	1	2	1	1	0	5	4	4	3	4	3	3
16	1	2	2	3	2	3	3	4	2	3	3	4	3	4	4	5	0	1	1	2	1	2	2
17	2	1	3	2	3	2	4	3	3	2	4	3	4	3	5	4	1	0	2	1	2	1	3
18	2	3	1	2	3	4	2	3	3	4	2	3	4	5	3	4	1	2	0	1	2	3	1
19	3	2	2	1	4	3	3	2	4	3	3	2	5	4	4	3	2	1	1	0	3	2	2
20	2	3	3	4	1	2	2	3	3	4	4	5	2	3	3	4	1	2	2	3	0	1	1
21	3	2	4	3	2	1	3	2	4	3	5	4	3	2	4	3	2	1	3	2	1	0	2

$$transCodes = \begin{pmatrix} 0 \\ 7 \\ 25 \\ 30 \\ 42 \\ 45 \\ 51 \\ 52 \end{pmatrix}$$

```
transCodes :=  k ← 0
               M_k ← 0
               newM ← 1
               while newM ≤ 2^n − 1
                   while codeDistance(M_k, newM, n) < d
                     newM ← newM + 1
                   isCross ← 0
                   i ← 0
                   while (i ≤ k) ∧ (isCross = 0)
                       isCross ← 1 if codeDistance(M_i, newM, n) < d
                       i ← i + 1
                   if isCross = 0
                       k ← k + 1
                       M_k ← newM
                   newM ← newM + 1
               M
```

6. Функція для визначення множини комбінацій, які будуть сприйматися на приймальній частині, якщо передана комбінація mainCode (знаходження множини відбувається шляхом знаходження тих комбінацій, які містять e - кількість помилок і менше)

```
resCods(mainCode) :=  k ← 0
                      for newCode ∈ 0.. 2^n − 1
                        if codeDistance(mainCode, newCode, n) ≤ e
                            M_k ← newCode
                            k ← k + 1
                      M
```

$$\text{resCods}(45) = \begin{pmatrix} 13 \\ 37 \\ 41 \\ 44 \\ 45 \\ 47 \\ 61 \end{pmatrix}$$

2.6 Методи стиснення даних

2.6.1 Код Шеннона-Фано

При передачі повідомлень, закодованих двійковим рівномірним кодом, не враховують статистичну структуру повідомлень, які незалежно від імовірності їх появи є кодовими комбінаціями однакової довжини, тобто кількість двійкових символів не одне постійне повідомлення. Такі коди мають надлишковість.

Найбільш ефективним способом зменшення надлишковості повідомлення є побудова оптимальних кодів, які мають мінімальну довжину кодових слів.

Враховуючи статистичні властивості джерела повідомлення, можна мінімізувати середнє число символів, які потрібні для вираження одного знаку повідомлення, що при відсутності шуму дозволяє зменшити час передачі або об'єм запам'ятовувального пристрою.

Ефективне кодування повідомлення для передачі їх по дискретному каналу без перешкод базується на теоремі Шеннона, яку можна сформулювати так:

Повідомлення джерела з ентропією $H(Z)$ завжди можна закодувати послідовностями символів з об'ємом алфавіту так, що середнє число символів на знак повідомлення $L_{сер}$ буде якомога ближче до величини $\frac{H(Z)}{\log(m)}$, але менше за неї.

Для одержання оптимального коду, що має мінімальну довжину кодової комбінації, необхідно домагатися найменшої надмірності кожного з кодів слова, що у свою чергу повинні будуватися з рівноймовірнісних і взаємонезалежних символів. При цьому кожен кодовий елемент повинен приймати значення 0 чи 1 по можливості з рівними ймовірностями, а вибір наступного елемента повинен бути незалежним від попереднього. Алгоритм побудови такого коду вперше був запропонований К. Шенноном у 1948 році і трохи пізніше модифікований Р. Фано, у зв'язку з чим він одержав назву Шеннона-Фано.

Відповідно до алгоритму на початку процедури кодування всі символи алфавіту джерела заносяться в таблицю в порядку зменшення ймовірностей. На першому етапі кодування символи розбиваються на дві групи таким чином, щоб суми ймовірностей символів у кожній з них були по можливості однакові. Усім символам верхньої групи присвоюється елемент кодової комбінації 0, а всім нижнім – 1. На другому етапі кодування кожна з груп знову розбивається на дві рівноймовірнісні підгрупи. Другому елементу кодових комбінацій для верхньої підгрупи присвоюється значення 0, а нижній – 1. Процес кодування продовжується до тих пір, поки в кожній підгрупі не залишиться по одному символу. Аналогічно може бути побудований альтернативний варіант оптимального префіксного коду Шеннона-Фано, у якому в процесі кодування верхнім підгрупам символів присвоюється кодовий елемент 1, а нижнім – 0.

Цей код відрізняється від попереднього тим, що його кодові комбінації для відповідних символів будуть інверсними.

Розглянемо алгоритм Шеннона-Фано на прикладі кодування джерела, алфавіт якого складається з 8 символів a_i, $(i=1,2,...,8)$, а імовірності появи символів у повідомленні дорівнюють від'ємним степеням двійки, тобто $P(a)=\left(\frac{1}{2}\right)^i$. Щоб ансамбль повідомлень джерела представляв повну групу подій, у прикладі прийнято $P(a_8)=P(a_7)=\left(\frac{1}{2}\right)^7$. Процедура розбивання символів на групи й підгрупи та утворення кодових слів показані у таблиці 2.7.

Середнє число бітів на символ при кодуванні в даному випадку кодом Шеннона-Фано дорівнює

$$L_{сер}=\sum_{i=1}^{8}P(a_i)l(a_i)=\frac{1}{2}+\frac{1}{4}\cdot 2+\frac{1}{8}\cdot 3+\frac{1}{16}\cdot 4+\frac{1}{32}\cdot 5+\frac{1}{64}\cdot 6+\frac{1}{128}\cdot 7+\frac{1}{128}\cdot 8=1\frac{63}{64},$$

а ентропія джерела

$$H(A)=\sum_{i=1}^{8}P(a_i)\log(a_i)=\frac{1}{2}\cdot\log\left(\frac{1}{4}\right)+\frac{1}{4}\cdot\log\left(\frac{1}{4}\right)+\frac{1}{8}\cdot\log\left(\frac{1}{8}\right)+\frac{1}{16}\cdot\log\left(\frac{1}{16}\right)+$$

$$\frac{1}{32}\cdot\log\left(\frac{1}{32}\right)+\frac{1}{64}\cdot\log\left(\frac{1}{64}\right)+\frac{1}{128}\cdot\log\left(\frac{1}{128}\right)+\frac{1}{128}\cdot\log\left(\frac{1}{128}\right)=1\frac{63}{64}.$$

Таким чином, код Шеннона-Фано для заданого розподілу ймовірностей символів є оптимальним, оскільки середнє число бітів на символ у точності дорівнює ентропії джерела. При звичайному кодуванні для представлення кожного символу необхідно 3 біта. Отже, коефіцієнт стиснення по-

відомлення за рахунок нерівномірного кодування Шеннона-Фано дорівнює

$$\frac{1\frac{63}{64}}{3} = 0,66 = 66\%.$$

Таблиця 2.8- Розбивання символів на групи й підгрупи

Символи a_i	Імовірності $P(a_i)$	Етапи кодування							Кодові комбінації
		I	II	III	IV	V	VI	VII	
a_1	$\frac{1}{2}$	0							0
a_2	$\frac{1}{4}$	1	0						10
a_3	$\frac{1}{8}$	1	1	0					110
a_4	$\frac{1}{16}$	1	1	1	0				1110
a_5	$\frac{1}{32}$	1	1	1	1	0			11110
a_6	$\frac{1}{64}$	1	1	1	1	1	0		111110
a_7	$\frac{1}{128}$	1	1	1	1	1	1	0	1111110
a_8	$\frac{1}{128}$	1	1	1	1	1	1	1	1111111

Приклад кодування за алгоритмом Шеннона-Фано для ансамбля символів з довільним розподілом імовірностей приведено у таблиці 2.9.

Таблиця 2.9 – Кодування за алгоритмом Шеннона-Фано

Символи a_i	Імовірності $P(a_i)$	Етапи кодування				Кодові комбінації
		I	II	III	IV	
a_1	0,22	0	0			00
a_2	0,20	0	1	0		010
a_3	0,16	0	1	1		011
a_4	0,16	1	0	0		100
a_5	0,10	1	0	1		101
a_6	0,10	1	1	0		110
a_7	0,04	1	1	1	0	1110
a_8	0,02	1	1	1	1	1111

2.6.2 Лабораторна робота №6

Мета роботи. Засвоїти методику побудови коду Шеннона-Фано та за допомогою математичного пакета MathCAD виконати побудову коду.

Хід роботи

1. На основі виданого варіанта (таблиця 2.10) виконати стиснення вхідного повідомлення, використовуючи алгоритм побудови кода Шеннона-Фано.
2. Обчислити імовірність кожного символа повідомлення.
3. Обчислити середнє значення кількості бітів на один символ.
4. Визначити коефіцієнт стиснення.
5. Отримати кодові комбінації кожного символа.
6. Зробити висновки по роботі.

Контрольні запитання

1. За рахунок чого зменшується середня довжина кодових слів оптимального коду?
2. Який принцип побудови кодів Шеннона-Фано?
3. Вкажіть переваги та недоліки методики Шеннона-Фано.
4. Завдяки чому збільшується ефективність кодування?
5. В чому полягає теорема Шеннона?
6. Як визначається коефіцієнт стиснення вхідного повідомлення?

Таблиця 2.10 - Варіанти завдання

Варіант	Вхідне повідомлення
1	“The world is big the life is long”
2	“tentomentoentetento”
3	“In vina veritas”
4	“pacta sunt servanda”
5	“Homo sapiens?????????”
6	““”Homo sapiens – golota sapiens"
7	“Nil desperandum”
8	“It’s a to good day to die”
9	“Nemo me impune lacessit”

Приклад. Стиснення вхідного повідомлення методом Шеннона-Фано.

1. Записуємо вхідне повідомлення як символьну величину.

S := "aaaaaaaabbbbsadrgsddfsgsdfg"

2. Знаходимо кількість символів в цій величині, використовуючи функцію *strlen(string)*

countsimbol:= strlen(S) countsimbol= 27

3. Записуємо функцію видалення однакових знаків, використовуючи вбудовану функцію *concat*

$S_2 :=$ Stemp ← substr(S, 0, 1)
for i ∈ 1 .. countsimbol − 1
Stemp ← concat(Stemp, substr(S, i, 1)) if search(Stemp, substr(S, i, 1), 0) = −1
Stemp

S_2 = "absdrgf"

4. Записуємо функцію знаходження однакових знаків, використовуючи вбудовану функцію *search(string,s,k)*, де *string* - символьна величина, *s* - шуканий символ, *k* - індекс, з якого починається пошук.

find_s(s) :=
j ← 0
k ← 0
while (search(S, s, k) > −1)
  v_j ← search(S, s, k)
  k ← v_j + 1
  j ← j + 1
j

5. Створюємо вектор символів і вектор кількості їх повторень.

$k := 0 .. \text{strlen}(S_2) - 1$

$S\_end(k) := \text{substr}(S_2, k, 1)$ $P\_s(k) := \text{find\_s}(\text{substr}(S_2, k, 1))$

S_end (k) =

"a"
"b"
"s"
"d"
"r"
"g"
"f"

P_s (k) =

9
4
4
4
1
3
2

6. Об'єднуємо створені вектори в одну матрицю та впорядковуємо її (розмістимо повідомлення в порядку зменшення ймовірностей).

$$S\_end := \begin{pmatrix} "a" \\ "b" \\ "s" \\ "d" \\ "r" \\ "g" \\ "f" \end{pmatrix} \quad P\_s := \begin{pmatrix} 9 \\ 4 \\ 4 \\ 4 \\ 1 \\ 3 \\ 2 \end{pmatrix}$$

$Matrix^{\langle 0 \rangle} := S\_end$ $Matrix^{\langle 1 \rangle} := P\_s$

$$Matrix = \begin{pmatrix} "a" & 9 \\ "b" & 4 \\ "s" & 4 \\ "d" & 4 \\ "r" & 1 \\ "g" & 3 \\ "f" & 2 \end{pmatrix}$$

Matrix_sort := reverse(csort(Matrix,1))

$$Matrix\_sort = \begin{pmatrix} "a" & 9 \\ "b" & 4 \\ "d" & 4 \\ "s" & 4 \\ "g" & 3 \\ "f" & 2 \\ "r" & 1 \end{pmatrix}$$

7. Записуємо кодові комбінації кожного символа.

$$Main := \begin{pmatrix} 1 & 1 & "*" & "*" & "*" \\ 1 & 0 & "*" & "*" & "*" \\ 0 & 1 & "*" & "*" & "*" \\ 0 & 0 & 1 & "*" & "*" \\ 0 & 0 & 0 & 1 & "*" \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

8. Знаходимо середнє число бітів на символ та коефіцієнт стиснення

Середнє число бітів на символ:

$$L := \left| \begin{array}{l} \text{for } i \in 0..(\text{rows}(Main) - 1) \\ \quad \left| \begin{array}{l} L_i \leftarrow 0 \\ \text{for } j \in 0..(\text{cols}(Main) - 1) \\ \quad \left| \begin{array}{l} v_j \leftarrow \text{submatrix}(Main, i, j, 0, \text{cols}(Main) - 1)^T \\ L_i \leftarrow L_i + 1 \quad \text{if } v_j \neq "*" \end{array} \right. \end{array} \right. \\ L_i \end{array} \right. \qquad L := \begin{pmatrix} 2 \\ 2 \\ 2 \\ 3 \\ 4 \\ 5 \\ 5 \end{pmatrix}$$

$$n := \text{rows}(L) - 1 \qquad Ser\_number := \sum_{i=0} \frac{P\_s\_sort}{countsimbol} L$$

Коефіцієнт стиснення $Ser\_number = 18.926$

$$K := \frac{Ser\_number}{3} \cdot 100 \qquad K = 630.864$$

2.6.3 Код Хаффмена

Алгоритм Шеннона-Фано не завжди приводить до однозначної побудови коду. Це викликано тим, що при розбиванні m символів джерела на підгрупи можна зробити більшою за імовірністю як верхню, так і нижню підгрупи. При різному розбиванні на підгрупи середнє число бітів на символ може бути різним. Більш ефективний є алгоритм, запропонований Хаффменом у 1952 р., що дозволяє побудувати оптимальний код із найменшим для даного розподілу імовірностей середнім числом бітів на символ. Тобто

$$l_{cp} = \min\left[\sum_{i=1}^{m} P(a_i) l(a_i)\right]. \qquad (2.72)$$

Для двійкового коду алгоритм Хаффмена зводиться до такого. Символи повідомлення упорядковуються за зменшенням ймовірностей і розташовуються в основний стовпець таблиці таким чином, що $P(a_i) \geq P(a_j)$ для усіх $i<y$ (табл. 2.11).

Таблиця 2.11 - Алгоритм Хаффмена

Символи a_i	Імовірності $P(a_j)$	Додаткові стовпці						
a_1	0,22	0,22	0.22	0,26	0,32	0,42	0,58	1,0
a_2	0,20	0,20	0,20	0,22	0,26	0,32	0,42	
a_3	0,16	0,16	0,16	0,20	0,22	0,26		
a_4	0,16	0,16	0,16	0,16	0,20			
a_5	0,10	0,10	0,16	0,16				
a_6	0,10	0,10	0,10					
a_7	0,04	0,06						
a_8	0,02							

Два останніх символи поєднуються в один допоміжний, імовірність якого дорівнює сумарній імовірності складових його символів. Усі символи, що залишилися, разом з утвореним допоміжним символом, знову розташовуються за зменшенням імовірностей у додатковому стовпці. Два останніх елементи стовпця поєднуються в другий допоміжний символ, і утворюється наступний додатковий стовпець, у якому всі елементи, розташовані в порядку спадання ймовірностей. Процедура продовжується доти, поки не вийде єдиний допоміжний символ, що має імовірність, рівну одиниці. Для формування кодових комбінацій, що відповідають символам даного повідомлення, необхідно простежити шлях переходу символів по рядках і стовпцях таблиці. При побудові кодів Хаффмена найчастіше використовуються кодові дерева. З однієї сторони: це дозволяє більш наочно відобразити процедури кодування і декодування, а, з іншого боку –

програмну реалізацію цих процедур (рис. 2.20).

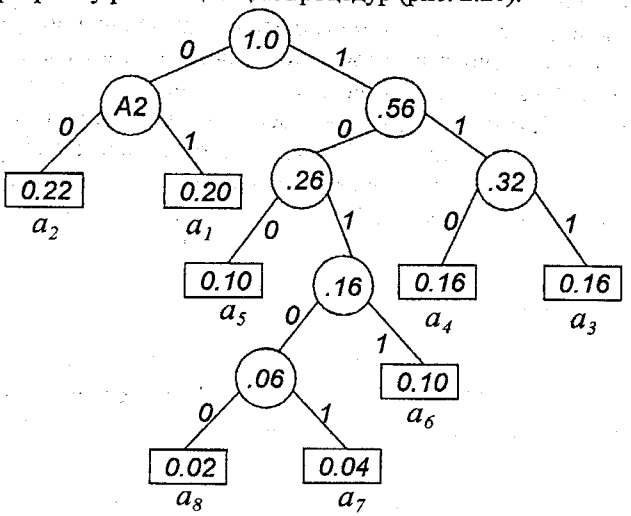


Рисунок 2.20 – Кодове дерево (а) і таблиця коду Хаффмена (б)

Побудова дерева (рис. 2.20,а) починається з кореневого вузла, імовірність якого дорівнює 1. З кореня проводяться дві вітки, причому вітці з більшою імовірністю присвоюється значення *(біт)* 1, а з меншою імовірністю - 0. Знову утворені вузли можуть відображати одиночний чи допоміжний символи. В останньому випадку вузол є проміжним і з кожного з них, у свою чергу, знову проводяться по дві вітки. Таке послідовне розгалуження продовжується доти, поки не буде побудований вузол, що відповідає імовірності символу алфавіту *(вузол листа)*. Рухаючись по кодовому дереву від кореня зверху вниз, можна записати для кожного символу відповідну йому кодову комбінацію (рис. 2.20 ,б). Середнє число бітів на символ при такій побудові коду складає

$$L_{cp} = \sum P(a_i) l(a_i) = 0.22 \times 2 \times 0.2 \times 2 \times 0.16 \times 3 \times 0.16 \times 3 \times 0.1 \times \\ \times 3 \times 4 \times 0.04 \times 5 = 2.8 \text{ біта}$$

Ентропія джерела повідомлення дорівнює:

$H(A) = -\sum P(a_i) \log P(a_i) =$ - (0,221og0,22 + 0,21og0,2 + 0,16log0,16 + 0,16log0,16 + 0,1log0,1 + 0,11og0,1 + 0,041og0,04 + 0,02tog0,02 = 2,754 біта

Як видно з розглянутого прикладу, середня довжина кодової комбінації й ентропія джерела практично збігаються, тобто отриманий код є оптимальним.

Алгоритм є двопрохідний, тому що при його реалізації потрібно двічі переглядати кодоване повідомлення. При першому проході обчислюються імовірності *(частоти)* появи символів у повідомленні і будується хаффменівське дерево.

При другому проході здійснюється кодування символів, що надійшли від джерела. У цьому випадку визначається значення віток дерева при русі від *листка,* що відповідає кодованому символу, до *кореня.* Очевидно, що для прискорення процедури декодування біти кодової комбінації на вихід кодера повинні видаватися, починаючи зі старшого розряду, тобто з вітки дерева, що виходить від кореня. На практиці замість імовірностей символів використовують абсолютні значення кількості *(вага)* символів у переданому повідомленні, тому що кількість символів пропорційно імовірності їхньої появи. Для однозначного декодування таблиця імовірностей символів повідомляється декодеру.

2.6.4 Лабораторна робота №7

Мета робота. Засвоїти методику побудови коду Хаффмена та за допомогою математичного пакета MathCAD 2000 виконати кодування .

Хід роботи

1. На основі виданого варіанта (лаб. робота №6) виконати стиснення вхідного повідомлення, використовуючи алгоритм побудови коду Хаффмена.
2. Обчислити імовірність кожного символу повідомлення.
3. Обчислити середнє значення кількості бітів на один символ.
4. Визначити коефіцієнт стиснення.
5. Отримати кодові комбінації кожного символу.
6. Зробити висновок по роботі.

Контрольні запитання

1. За рахунок чого зменшується середня довжина кодових слів оптимального коду?
2. Який принцип побудови кодів Хаффмена?
3. Вкажіть переваги та недоліки методики Хаффмена.
4. Яким чином будується дерево Хаффмена?
5. Як визначається коефіцієнт стиснення вхідного повідомлення?
6. Дайте порівняльну характеристику методик кодування Хаффмена й Шеннона-Фано.

Приклад. Стиснення вхідного повідомлення методом Хаффмена.

1. Записуємо вхідне повідомлення як символьну величину.

S := "aaaaaaaabbbbsadsgrttfsgfdsg"

2. Знайти кількість символів у цій величині, використовуючи функцію *strlen(string).*

countsimbol := strlen(S) countsimbol = 27

3. Записуємо функцію видалення однакових символів, використовуючи вбудовану функцію concat та, знаходимо кількість символів, що не повторюються.

```
S2 := | Stemp ← substr(S,0,1)                          S2 = "absdrgf"
      | for i ∈ (1.. countsimbol- 1)
      |   Stemp ← concat(Stemp,substr(S,i,1))  if search(Stemp,substr(S,i,1),0) = -1
      | Stemp
```

CountNsimbol := (strlen(S2)) CountNsimbol = 7

4. Записуємо функцію знаходження однакових знаків, використовуючи вбудовану функцію *search(string,s,k)*, де *string* – символьна величина, *s – шуканий символ*, *k* – індекс, із якого починається пошук.

```
find_s(s) := | j ← 0
             | k ← 0
             | while (search(S,s,k) > -1)
             |   | v_j ← search(S,s,k)
             |   | k ← v_j + 1
             |   | j ← j + 1
             | j
```

5. Створюємо вектор символів та векторів кількості їх повторень:

$k := 0 .. strlen(S2) - 1$ $S\_end_k := substr(S2, k, 1)$

$$S\_end = \begin{pmatrix} "a" \\ "b" \\ "s" \\ "d" \\ "r" \\ "g" \\ "f" \end{pmatrix}$$

$P\_s_k := find\_s(substr(S2, k, 1))$

$$P\_s = \begin{pmatrix} 9 \\ 4 \\ 4 \\ 4 \\ 1 \\ 3 \\ 2 \end{pmatrix}$$

6. Об'єднуємо створені вектори в одну матрицю та впорядковуємо її (розміщуємо повідомлення в порядку зменшення ймовірностей).

$Matrix^{\langle 0 \rangle} := S\_end$

$Matrix^{\langle 1 \rangle} := P\_s$

$$Matrix = \begin{pmatrix} "a" & 9 \\ "b" & 4 \\ "s" & 4 \\ "d" & 4 \\ "r" & 1 \\ "g" & 3 \\ "f" & 2 \end{pmatrix}$$

$Matrix\_soft := reverse(csort(Matrix, 1))$

$$Matrix\_soft = \begin{pmatrix} "a" & 9 \\ "b" & 4 \\ "d" & 4 \\ "s" & 4 \\ "g" & 3 \\ "f" & 2 \\ "r" & 1 \end{pmatrix}$$

Таким чином, отримуємо таблицю, в якій розміщені кількості повторень певного символа (можна представити і як імовірності, розділивши кожне значення на кількість символів) у порядку спадання.

7. Записуємо функцію знаходження двох символів у стовпці *col*:

$f\_sum2last(col) :=$
- $v \leftarrow submatrix(Matrix\_soft, 0, countNsimbol - 1, col, col)$
- $v_{countNsimbol+2} + v_{countNsimbol+1}$

8. Створюємо матрицю, аналогічну таблиці 1:

$M\_Hafm :=$
- $M\_Hafm \leftarrow Matrix\_sort$
- $v \leftarrow submatrix(Matrix\_sort, 0, countNsimbol - 1, 1, 1)$
- for $i \in 2 .. countNsimbol$
  - $v2 \leftarrow reverse(sort(v))$
  - $sum \leftarrow v2_{countNsimbol+1} + v2_{countNsimbol+1}$
  - $v2_{countNsimbol+1} \leftarrow sum$
  - for $k \in (countNsimbol - i + 1) .. (countNsimbol - 1)$
    - $v2_k \leftarrow 0$
  - $M\_Hafm^{\langle i \rangle} \leftarrow reverse(sort(v2))$
  - $v \leftarrow v2$
- M_Hafm

2.6.5 Арифметичне кодування

При арифметичному стисненні повідомлень алфавіту джерела ставиться у відповідність числовий, відкритий праворуч, інтервал [0,1), а кожен символ алфавіту зіставляється з різними ділянками цієї числової осі. Ширина інтервалу (*діапазон*) кожної ділянки залежить від імовірності (*частоти*) появи символа в повідомленні.

Розглянемо, як приклад, алфавіт, що складається із шести символів A, B, C, D, E, ! з імовірностями появи відповідно 0,1; 0,1; 0,3; 0,2; 0,2 та 0,1. Тоді інтервал [0,1) може бути розділений на ділянки таким чином:

A: [0, 0,1), B: [0,1, 0,2), C: [0,2, 0,5), D: [0,5, 0,7), E: [0,7, 0,9), !: [0,9, 1,0).

Розподіл числової осі ілюструється рис. 2.21. Як видно з прикладу, поділ одиничного інтервалу зручно здійснюється підсумовуванням імовірності кожного з границею попереднього.

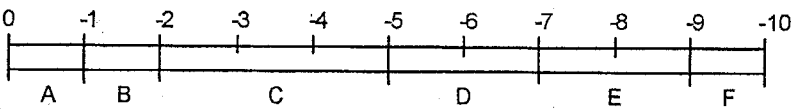


Рисунок 2.21 – Розподіл відрізків числової осі між символами при арифметичному кодуванні

У процесі стиснення рядок текстового повідомлення представляється двома дробовими числами, що відображають деякий інтервал числової осі [0, 1). До початку кодування допустимою областю для повідомлення є інтервал [0, 1). При надходженні від джерела першого символу йому виділяється інтервал, що відповідає розташуванню цього символ на осі [0, 1) відповідно до імовірності появи його в повідомлені. Надходження наступного символу звужує виділений інтервал. При цьому границі нового інтервалу визначаються числовим діапазоном, що відповідає черговому символу, тобто ширина діапазону пропорційна імовірності символа. З приходом наступного символа виробляється подальше звуження інтервалу. Таким чином, із збільшенням довжини рядка, що кодується, відбувається постійне звуження інтервалу. Теоретично його можна звужувати як завгодно довго, а на практиці цей інтервал обмежується технічними можливостями пристроїв, що кодують (розрядністю слова).

Число бітів, необхідне для представлення інтервалу шириною s, дорівнює $-\log(s)$. Основа логарифму тут і в наступних виразах дорівнює 2. Ширина s останнього інтервалу рядка повідомлення, що складається з N_c символів визначається добутком ймовірностей символів P_{jC} повідомлення

$$S = \prod_{i=1}^{N_C} P_{jC}.$$

В зв'язку з цим можна записати, що

$$-\log(s) = -\sum \log P_i = -\sum_{i=1}^{m} P(a_i)\log P(a_i), \qquad (2.73)$$

де m – кількість різних символів повідомлення a_i, $(i = 1,2,...,m)$. Таким чином, число бітів, затрачуване на кодування повідомлення, при арифметичному кодуванні в точності дорівнює ентропії джерела.

Розглянемо процедуру арифметичного стиснення на прикладі кодування рядка повідомлення (CADA!). Тут символ ! є ознакою закінчення рядка. Імовірності появи символів і відповідні їм числові діапазони візьмемо з попереднього прикладу. Початковий числовий діапазон, який виділяється для кодера, дорівнює одиниці [0,1). Після надходження від джерела першого символа C числовий інтервал звужується до величини [0,2, 0,3). З приходом наступної букви A границі діапазону стають рівними [0,2, 0,23). Процедура кодування наочно ілюструється на рис. 2.22. Вертикальні лінії відображають початкову числову вісь [0, 1) і її наступні ділянки в збільшеному розмірі (промасштабовані на одиничну вісь) на різних етапах кодування. Який би не був довгий рядок, кінцевий діапазон завжди буде розташований в інтервалі першого кодованого символа. Підрахунок інтервалів здійснюється за рекурентною формулою

$$\begin{aligned} (X_{\max} - X_{\min})X'_{\min} + X_{\min} \\ (X_{\max} - X_{\min})X'_{\max} + X_{\min}, \end{aligned} \qquad (2.74)$$

де $X_{\max}$ і $X_{\min}$ – нижня і верхня границі для попередньої ітерації; $X'_{\min}$ і $X'_{\max}$ – границі інтервалу нового символа.

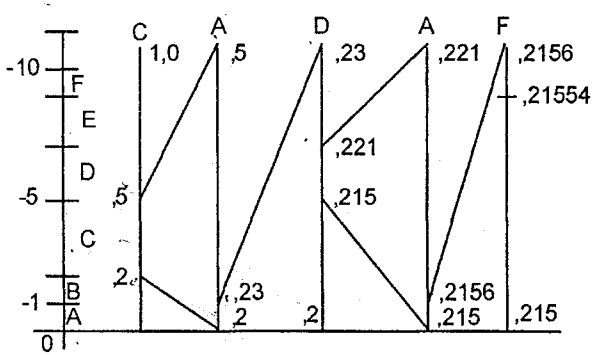


Рисунок 2.22 – Ілюстрація процедури арифметичного кодування

2.6.6 Алгоритм арифметичного кодування в загальному вигляді

ПОЧАТОК
Нижня_границі = 0,0;
Верхня_границя = 1,0;
ПОКИ існує вхідний символ ВИКОНУВАТИ
Увести черговий символ С;
Інтервал = Верхня_границя – Нижня_границя;
Верхня_границя = Нижня_границя + Інтервал· Нижня_границя(С);
Нижня_границя = Нижня_границя + Інтервал· Верхня_границя(С);
КІНЕЦЬ ПОКИ
ВИВЕСТИ Верхня_Границя, Нижня_границя;
КІНЕЦЬ

Границі інтервалів для всіх етапів кодування приведені у таблиці 2.12. Таким чином, числовий інтервал із границями [0,21554, 0,2156) відображає рядок повідомлення (CADA!). Для кодування цього рядка може бути виконане будь-яке число з отриманого діапазону. Нехай таким числом є нижня границя інтервалу, тобто $X = 0,21554$.

Таблиця 2.12 - Границі інтервалів

Символи	Нижня границя інтервалу	Верхня границя інтервалу
Вихідний стан	0,0	1,0
C	0,2	0,5
A	0,2	0,23
D	0,215	0,221
A	0,215	0,2156
!	0,21554	0,2156

2.6.7 Лабораторна робота №8

Мета роботи. Засвоїти методику арифметичного кодування за допомогою математичного пакета MathCAD.

Хід роботи

1. На основі виданого варіанта (таблиця 2.13) виконати стиснення вхідного повідомлення, використовуючи алгоритм арифметичного кодування.
2. Обчислити імовірність кожного символа повідомлення.
3. Виконати числовий розподіл осі між символами.

4. Визначити нижню та верхню границю кожного символа.
5. Визначити кінцеву нижню та верхню границю повідомлення.
6. Зробити висновок по роботі.

Таблиця 2.13 – Варіанти завдань

Варіант	Вхідний алфавіт	Імовірність появи	Вхідне повідомлення
1	"abcdefgh"	0,1; 0,2; 0,1; 0,3; 0,2; 0,2; 0,2; 0,3	"cadaf"
2			"fefehfa"
3			"bebehh"
4			"fefecf"
5			"hehecc"
6	"klnmopqae"	0,2; 0,3; 0,1; 0,3; 0,1; 0,1; 0,2; 0,2	"onopan"
7			"kalanaq"
8			"pelenop"
9			"aoaoao"
10			"nopaepa"

Контрольні запитання

1. Який принцип побудови арифметичних кодів?
2. Укажіть переваги та недоліки арифметичних кодів?
3. В чому полягає основний алгоритм побудови арифметичного коду?
4. Як визначається верхня та нижня границя ітерації при кодуванні?
5. Як визначається верхня та нижня границя ітерації при розкодуванні?
6. Як відбувається розподіл відрізків числової осі між символами при арифметичному кодуванні?
7. Дайте порівняльну характеристику методик кодування Хаффмена, Шеннона-Фано, арифметичних кодів.

Приклад . Стиснення повідомлення методом арифметичного кодування.

1. Записуємо вказаний алфавіт згідно з прикладом табл. 2.12 як символьну величину, знаходимо кількість символів, створюємо масив символів, а також їх імовірності.

S := "ABCDE!"

countsimbol := strlen(S) countsimbol = 6

i := 0.. countsimbol – 1 S_end(i) := substr(S ,i,1)

$$S\_end(i) = \begin{pmatrix} "A" \\ "B" \\ "C" \\ "D" \\ "E" \\ "!" \end{pmatrix} \qquad P\_s := \begin{pmatrix} 0.3 \\ 0.1 \\ 0.3 \\ 0.2 \\ 0.2 \\ 0.1 \end{pmatrix}$$

–ймовірність символів

2. Виконаємо розподіл числової осі між символами

gran :=
interm ← 0
for i ∈ 0.. countsimbol– 1
  $n_i \leftarrow$ interm
  $v_i \leftarrow n_i + P\_s_i$
  interm ← v_i
$gran^{\langle 0 \rangle} \leftarrow n$
$gran^{\langle 1 \rangle} \leftarrow v$

$$gran = \begin{pmatrix} 0 & 0.1 \\ 0.1 & 0.2 \\ 0.2 & 0.5 \\ 0.5 & 0.7 \\ 0.7 & 0.9 \\ 0.9 & 1 \end{pmatrix}$$

3. Задаємо вхідне повідомлення (як відповідні індекси вхідного алфавіту). В даному випадку: "CADA!"

$$index := \begin{pmatrix} 2 \\ 0 \\ 3 \\ 0 \\ 5 \end{pmatrix}$$

4. Знаходимо нижню й верхню границі кожного символа вхідного алфавіту на осі

```
gran := | interm ← 0
        | for i ∈ 0 .. countsimbol − 1
        |     n_i ← interm
        |     v_i ← n_i + P_s_i
        |     interm ← v_i
        | gran^⟨0⟩ ← n
        | gran^⟨1⟩ ← v
        | gran
```

$$gran = \begin{pmatrix} 0 & 0.1 \\ 0.1 & 0.2 \\ 0.2 & 0.5 \\ 0.5 & 0.7 \\ 0.7 & 0.9 \\ 0.9 & 1 \end{pmatrix}$$

5. Згідно з вказаним алгоритмом знаходимо кінцеві нижню й верхню границі кожного символа повідомлення

```
| interm_n ← 0
| old_ng ← gran2^⟨0⟩
| old_vg ← gran2^⟨1⟩
| for i ∈ 0 .. rows(index) − 1
|     n_i ← interm_n + interval · old_ng_i
|     v_i ← interm_n + interval · old_vg_i
|     interm_n ← n_i
|     interval ← v_i − n_i
| m^⟨0⟩ ← n
| m^⟨1⟩ ← v
| m
```

$$m = \begin{pmatrix} 0.2 & 0.5 \\ 0.2 & 0.23 \\ 0.215 & 0.221 \\ 0.215 & 0.2156 \\ 0.21554 & 0.2156 \end{pmatrix}$$

2.7 Динамічні методи стиснення даних

2.7.1 Динамічне кодування методом Хаффмена

Класичний метод кодування Хаффмена визначає від початку перетворення вірогідності появи символів на виході джерела інформації, причому символи впорядковується за спаданням вірогідностей їх виникнення. Для впорядкованого списку складається кодова таблиця, у якій довжина вихідної комбінації визначається вірогідністю вихідного символа. Природно, що на передавальній та приймальній сторонах повинні бути відомі таблиці (кодові дерева) для кожного повідомлення, що стискається. Цей метод має два вагомих недоліки. Перший полягає в тому, що для його реалізації потрібно два проходи масиву, що кодується. При першому перегляді обчислюються імовірності появи кожного знака в повідомленні і складається таблиця коду Хаффмена. На наступному етапі здійснюється кодування на підставі статичної структури дерева Хаффмена і передача символів у стиснутому вигляді. Таким чином, виграш, отриманий за рахунок стискання даних, може помітно знижуватись, особливо при передачі щодо короткого повідомлення, у зв'язку з необхідністю передавати декодеру додаткову інформацію про кодове дерево. Другий недолік – наявність затримки від моменту надходження даних від джерела до видачі відповідних кодових комбінацій, що обмежує використання нерівномірного кодування в синхронних мережах передачі інформації, а також у системах, що функціонують у реальному часі.

На початку 70-х років були розроблені однопрохідні методи стискання інформації, засновані на класичній процедурі кодування Хаффмена. Усі ці методи незначно відрізняються один від одного і їхня суть полягає в тому, що передавач будує дерево Хаффмена в темпі надходження даних від джерела, тобто "*на ходу*". У процесі кодування відбувається "*навчання*" кодера на основі статичних характеристик джерела повідомлень, у ході якого обчислюється оцінка вихідних імовірностей повідомлення і виробляється відповідна модифікація кодового дерева, цей процес отримав назву *динамічного кодування Хаффмена*. Очевидно, що для правильного відновлення стиснутих даних, декодер також повинен безупинно "*вчитися*" поряд із кодером, здійснюючи синхронну зміну кодової таблиці на приймальній стороні. Для забезпечення синхронності процесів кодування і декодування кодер видає символ у нестиснутому вигляді, якщо він уперше з'явився на виході джерела, і визначає його на кодовому дереві. З повторною появою символа на вході кодера він передається нерівномірною кодовою комбінацією, обумовленою позицією символа на поточному кодовому дереві. Кодер корегує дерево Хаффмена збільшенням частоти передачі си-

мволів, що уже введені в дерево, чи нарощує дерево, додаючи в нього нові вузли.

Найважливішою умовою, якої потрібно дотримуватись при модифікації кодового дерева, є збереження властивостей хаффменівського дерева. Для формулювання цих властивостей звернемося ще раз до алгоритму побудови оптимального коду Хаффмена. При статичному кодуванні символи розміщуються в списку в порядку спадання ваг (ймовірностей). Потім проводиться об'єднання двох вузлів найменшої ваги W_i, W_j і заміна їх внутрішнім вузлом з вагою, рівною сумі вихідних ваг $W_i + W_j$. Знову утворений вузол розміщується в списку таким чином, щоб не порушувався порядок розташування вузлів за вагами. Цей процес повторюється доти , поки в списку не залишиться один, так званий *кореневий* вузол.

Розглянемо приклади побудови дерев оптимального коду Хаффмена (рис. 2.23 та рис. 2.24) і проаналізуємо їх властивості. Як видно з рисунків, вузли дерева розташовані в порядку зростання їхніх ваг при обході дерева від крайнього нижнього вузла до кореня ліворуч, праворуч і знизу вгору. У зв'язку з тим, що вузли з вагами W_i, W_j поєднуються парно, то на одному рівні не може бути менше двох вузлів, причому пари вузлів є дочірні в загальному батьківському вузлі, вага якого дорівнює сумі ваг дочірніх вузлів. Неважко переконатися, якщо при побудові дерева припустити, що дочірній вузол із великою вагою з'єднаний нульовою віткою, а з меншою вагою – одиничною, то хаффменівське дерево залишається упорядкованим за зростанням ваг при русі від нижнього вузла до кореня по рівнях праворуч ліворуч.

При побудові кодових дерев і їхньої модифікації проводиться нумерація вузлів із першого до $(2m-1)$-го в порядку збільшення їхніх ваг. Перший номер присвоюється вузлу з мінімальною вагою. Тут m - число символів алфавіту джерела. На рис. 2.23 показано дерево Хаффмена, побудоване для повідомлення $18A$, $10B$, $2C$, $2D$, $1E$, $1F$, $1G$ та $1H$, приведеного в прикладі 1 за умови, що вага символа $3I$ збільшиться і стане рівною 3. Зробимо нумерацію вузлів дерева, починаючи з листка з мінімальною вагою (нижнім крайнім лівим вузлом). Як видно з рисунка, властивості хаффменівського дерева, побудованого для статичного коду, зберігаються, хоча воно і придбало іншу конфігурацію. При цьому ваги усіх вузлів на шляху від листка із символом 3 до кореня збільшиться на 1.

При динамічному кодуванні Хаффмена після одержання від джерела наступного символа (наприклад, "C") повинен збільшитись на 1 вагу відповідного йому листка (рис. 2.24). Але це приведе до порушення порядку розташування вузлів за вагами, властивому оптимальному кодовому дереву. Тому при одержанні чергового символа від джерела необхідно здійсни-

ти модифікацію кодового дерева, щоб воно залишалося оптимальним, так званим хаффменівським деревом.

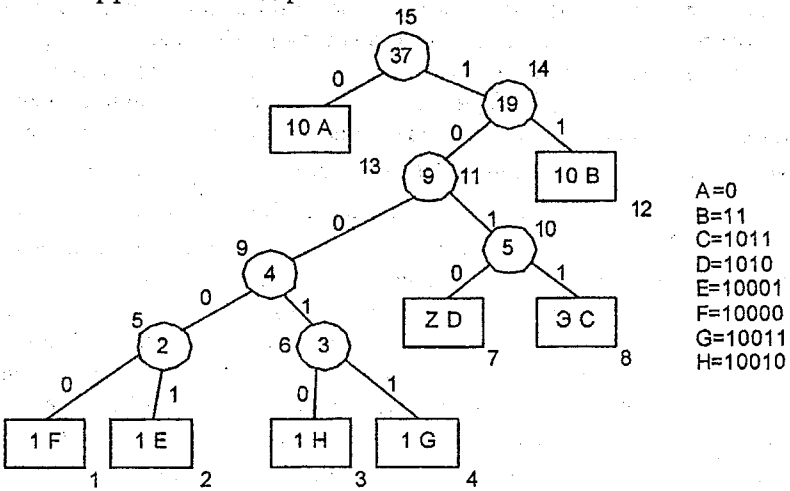


Рисунок 2.23 – Модифіковане дерево Хаффмена для прикладу 1, після надходження чергового символа "C"

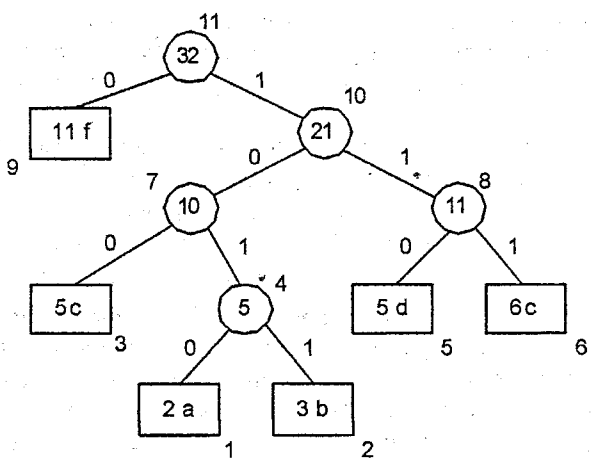


Рисунок 2.24 – Дерево Хаффмена для повідомлення, що складається з 32-х символів

Таким чином, на підставі аналізу властивостей оптимальних кодових дерев можна зробити висновок, що кодове дерево, яке має m зовнішніх вузлів, є хаффменівським, якщо воно має такі властивості:

а) зовнішні вузли (листки) хаффменівського кодового дерева мають вагу $W > 0$; кожен внутрішній (*батьківський*) вузол має підлеглі (дочірні) вузли, а його вага дорівнює сумі підлеглих (*дочірніх*) ваг;

б) на кожному рівні дерева (за винятком кореневого) повинно бути не менш однієї пари вузлів, що мають загальний батьківський вузол;

в) усі вузли нумеруються в зростаючому порядку таким чином, що вузли з номерами $(2j-1)$ та $2j$ є вузлами одного рівня для $l < j < m-l$, а їх загальний батьківський вузол має більш високий рівень;

г) нумерація вузлів відповідає тому порядку, у якому вузли поєднуються у відповідності зі статичним алгоритмом Хаффмена.

2.7.2 Лабораторна робота №9

Мета роботи. Ознайомитися з динамічним кодуванням стиснення даних методом Хаффмена, отримати навики побудови кодового дерева та здійснити стиснення даних .

Хід роботи

1. Побудувати нульову вітку дерева Хаффмена згідно з варіантом завдання (таблиця 2.14).

2. При надходженні першого символа побудувати перший родинний вузол, при цьому вліво йде вітка з меншою вагою, тобто нульова.

3. Для кожного наступного символа зробити звірення із символами, які вже представлені на дереві. Якщо символ вже є, то він передається в стиснутому вигляді. При надходженні кожного наступного символа необхідно перевіряти умову, щоб уліво йшла вітка з меншою вагою. Якщо ця умова не виконується, то необхідно вузли поміняти місцями.

4. Виконувати пункти 2 і 3 поки не закінчиться кодова комбінація.

5. По дереву Хаффмена отримати кодову комбінацію для кожного символа.

Таблиця 2. 14 – Варіанти завдання

Варіант	Кодова комбінація
1	SDYTRGYUHSTRNRG
2	FGJKRKJGKGTKFGFR
3	YTUIRYIUERTUYTUT
4	ZXXCVDFZCXXDFZD
5	MNBVNMGNHMNJN
6	LKKJLUKJYLHJKLLL
7	QWASDREWQDSAFD
8	POIULKIOPULKJPOU
9	GHJJFYGHJJHGFYRTJ
10	HGJHGYJHGTHJGYFH

Приклад . Кодуванням стиснення даних методом Хаффмена

1.Функція знаходження ймовірностей символів в рядку

```
cpmat(str) := | S ← str2vec(str)                    перетворюємо вхідний рядок у вектор
              | C_0 ← S_0
              | P_0 ← 1
              | for k ∈ 1.. last(S)                 -0-му елементу масиву кодів присво-
              |   | isAdd ← 0                       юємо значення коду першого симво-
              |   | for i ∈ 0.. last(C)             ла в рядку;
              |   |   if C_i = S_k                  -в масив P заносимо початкову кіль -
              |   |     | P_i ← P_i + 1             кість символів, що містяться в ма-
              |   |     | isAdd ← 1                 сиві C;
              |   | if isAdd = 0                    -isAdd - змінна, яка показує чи еле-
              |   |   | C_last(C)+1 ← S_k           мент доданий до масиву;
              |   |   | P_last(P)+1 ← 1             -перевіряємо, чи є в масиві C черговий
              | msum ← Σ P                          символ із вхідного масиву S;
              | for i ∈ 0.. last(P)                 -якщо є, тоді збільшуємо на 1 значен-
              |   P_i ← P_i / msum                  ня відповідного значення масиву P, в
              | augment(C,P)                        якому містяться кількість однакових
                                                    символів в рядку;
                                                    -якщо елемент не був доданий, то
                                                    необхідно створити новий елемент
                                                    масиву C і відповідне значення в
                                                    P установити 1;
                                                    -знаходимо суму елементів масиву P;
                                                    -знаходимо імовірність появи кожної
                                                    літери і записуємо їх в масив P.
```

2. Функція пошуку числа n у масиві M, повертає індекс елементу

```
search(n,M) := for i ∈ 0.. rows(M) − 1
                 return i if M_i = n
```

3. Функція кодування

```
encoder(CP) := | for i ∈ 0.. rows(CP) − 1
               |   | Haff_{i,0} ← CP_{i,0}
               |   | (CP_{i,0})_0 ← CP_{i,0}   початкові установки
               | Haff_{0,1} ← 0
               | for i ∈ 0.. rows(Haff) − 1
               |   (Haff_{i,1})_0 ← −1
               | while rows(CP) ≥ 2
               |   | for i ∈ 0.. last(CP_{0,0})
               |   |   | j ← search[(CP_{0,0})_i, Haff^⟨0⟩]
               |   |   | (Haff_{j,1})_{last(Haff_{j,1})+1} ← 0  if (Haff_{j,1})_{last(Haff_{j,1})} ≠ −1
               |   |   | (Haff_{j,1})_{last(Haff_{j,1})} ← 0  otherwise
               |   | for i ∈ 0.. last(CP_{1,0})
               |   |   | j ← search[(CP_{1,0})_i, Haff^⟨0⟩]
               |   |   | (Haff_{j,1})_{last(Haff_{j,1})+1} ← 1  if (Haff_{j,1})_{last(Haff_{j,1})} ≠ −1
               |   |   | (Haff_{j,1})_{last(Haff_{j,1})} ← 1  otherwise
               |   | CP_{1,1} ← CP_{1,1} + CP_{0,1}
               |   | CP_{1,0} ← stack(CP_{1,0}, CP_{0,0})
               |   | CP ← submatrix(CP, 1, rows(CP) − 1, 0, cols(CP) − 1)
               |   | CP ← csort(CP, 1)
               | Haff
```

str := "abrakadabra"

CP := cpmat(str)

$$CP = \begin{pmatrix} 97 & 0.455 \\ 98 & 0.182 \\ 114 & 0.182 \\ 107 & 0.091 \\ 100 & 0.091 \end{pmatrix}$$

CP := csort(CP,1)

$$CP = \begin{pmatrix} 100 & 0.091 \\ 107 & 0.091 \\ 114 & 0.182 \\ 98 & 0.182 \\ 97 & 0.455 \end{pmatrix}$$

code := encoder(CP)

$$code = \begin{pmatrix} 100 & \{4,1\} \\ 107 & \{4,1\} \\ 114 & \{2,1\} \\ 98 & \{3,1\} \\ 97 & \{1,1\} \end{pmatrix}$$

$$code_{0,1} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad code_{1,1} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \quad code_{2,1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad code_{3,1} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad code_{4,1} = (0)$$

2.7.3 Динамічне кодування методом FGK

Уперше алгоритм синтезу динамічного коду Хаффмена був запропонований Н. Феллером у 1973 році, а потім модифікований Р. Галлагером і Д. Батогом. У зв'язку з цим він одержав назву "*алгоритм FGK*".

Введемо деякі позначення, що будуть використовуватись при аналізі й синтезі динамічного кодового дерева Хаффмена за алгоритмом *FGK*: m - розмір алфавіту джерела повідомлень; z_i – $j-$й символ алфавіту; $M(k) = z(1), z(2),..., z(k)$ - перші k символів у повідомленні; k - число символів у повідомленні, оброблених до поточного моменту часу; $z(k)$ - k-й символ у повідомленні; К - кількість різних символів, оброблених на даний момент часу; W_j - число (вага) символів z_j, що надійшли на момент обробки повідомлення; l_j - відстань від кореня дерева до z_j-го листка.

Суть алгоритму синтезу динамічного кодового дерева Хаффмена полягає в процедурі обчислення листків і побудові бінарного дерева з міні-

льною вагою шляху $\sum_j W_j \cdot l_j$. Процедуру можна умовно розбити на два етапи, хоча при реалізації алгоритму вони можуть легко бути об'єднані в один. На першому етапі дерево Хаффмена, побудоване після обробки повідомлення $M(k)$, перетвориться в інше, еквівалентне вихідному, яке потім простим збільшенням ваг може бути перетворене в хаффменівське дерево для $M(k+l)$.

Перший етап починається після одержання від джерела символа $z(k+l)$ із присвоєння статусу *поточного* вузла листка $z(k+l)$. Потім проходить обмін поточного вузла (включаючи утворене ним піддерево), із вузлом, що має *найбільший порядковий номер та вагу*. Після цього в якості нового поточного вузла ініціюється батьківський вузол останнього поточного вузла. Обмін вузлами в разі потреби багаторазово повторюється, поки не буде досягнутий корінь дерева. Неважко переконатися, що максимальна кількість перестановок, які можуть знадобитися при модифікації кодового дерева, дорівнює висоті дерева $l_{\max}$.

На другому етапі формується листок дерева, що відповідає оброблюваному символу і наступні проміжні вузли, розташовані на шляху руху від листка до кореня дерева.

Оригінальний спосіб був запропонований Д. Кнутом. Його суть полягає в такому. Усі символи використовуваного алфавіту, що ще не з'явилися на виході джерела, відмічаються на дереві *нульовим вузлом*. Тому, коли символ, що підлягає стисненню, генерується вперше, кодер "відзначає" його, виробляючи *префіксну комбінацію*. Слідом за нею на вихід надходить код нестиснутого символа. У якості префіксної комбінації використовується код, що відповідає листку *нульової ваги*. Для кодування символів, що надійшли від джерела вперше, використовується не 8-ми розрядна комбінація ASCII-коду, а мінімальний префіксний код, побудований на підставі таких міркувань: якщо є повідомлення, що складається із m символів $a_i,\ldots, a_m$, то число m можна представити у вигляді цілого ступеня двійки і цілого числа m

$$m = 2^e + r, \tag{2.75}$$

де $0 < m < 2^e$.

Тоді a_k-й символ кодується $(e+1)$-ю бітовою комбінацією числа $k-1$, якщо $1 \le k \le 2r$. У протилежному випадку – e-ю бітовою комбінацією числа $k-r-1$. Наприклад, якщо $m=6$, то $e=2$ та $r=2$, а символи джерела будуть відображатися комбінаціями:

$$a_1 \Rightarrow 000;\ a_2 \Rightarrow 001;\ a_3 \Rightarrow 010;\ a_4 \Rightarrow 001;\ a_5 \Rightarrow 10;\ a_6 \Rightarrow 11.$$

Неважко помітити, що отриманий код має властивість префікса. Цей код є оптимальним, якщо символи мають однакову імовірність.

Неважко помітити, що отриманий код має властивість префікса. Цей код є оптимальним, якщо символи мають однакову імовірність.

2.7.4 Лабораторна робота №10

Мета роботи. Ознайомитися з динамічним кодуванням методом FGK стиснення даних, отримати навики побудови кодового дерева та здійснити стиснення даних .

Хід роботи

1. Визначити кількість символів в повідомленні згідно з варіантом (таблиця 2.15).
2. Визначити комбінацію префіксного коду, що відображає перший символ і відобразити символ на кодовому дереві листком із вагою 1, що з'єднаний одиничною віткою з коренем дерева.
3. З надходженням від джерела наступного символа на вихід кодера видати *0-код нульового вузла*, а за ним кодову комбінацію, що відображає символ, що визначається за формулою (2.75).
4. Продовжуючи процедуру кодування до завершення рядка, одержати набір кодових комбінацій, що відповідають символам, які надійшли.
5. По дереву Хаффмена отримати кодову комбінацію для кожного символа.

Таблиця 2. 15 – Варіанти завдання

Номер варіанта	Кодова комбінація
1	Baracudauda
2	Herculessles
3	Colibrivibru
4	Canzascity
5	Bangkokhilton
6	Carnavalbal
7	Bazukamuka
8	Mashinashina
9	Maraburayon
10	Banzayzay

Приклад. Кодування повідомлення методом FGK

На прикладі слова ABRACADABRA! Символ "!" – означає кінець рядка.

1. Оскільки ми використовуємо тільки літери латинського алфавіту й знак "!", то $m = 2^e + r = 2^4 + 11 = 27$;

2. Оскільки A є першою буквою алфавіту, то $k = 1$. Отже, комбінація префіксного коду, що відображає символ A, буде 00000. Ця комбінація поступає на вихід, а символ A відмічається на кодовому дереві листком з вагою 1, який з'єднаний одиничною віткою з коренем дерева.

3. З надходженням від джерела символа B на виході видається *0-код нульового вузла*, а за ним кодова комбінація, що відповідає B і визначається за формулою (2.75). Таким чином $B \Rightarrow 00001$.

4. Фіксується перший нульовий біт – біт нульової ваги. Далі поступає символ R, який ще не піддавався стисненню. Буква R є 18-ю в латинському алфавіті, тобто k=18, а код, що відповідає – 1000.

5. У зв'язку з тим, що код нульового вузла дерева дорівнює 00, то символ R буде закодований семирозрядною комбінацією 0010001. Кодове дерево при цьому модифікується .

6. Продовжуючи процедуру кодування до завершення рядка, отримаємо набір кодових комбінацій :

$A \Rightarrow 0$; $C \Rightarrow 10000010$; $A \Rightarrow 0$; $D \Rightarrow 110000011$;

$A \Rightarrow 0$; $B \Rightarrow 110$; $R \Rightarrow 110$; $A \Rightarrow 0$.

2.7.5 Динамічне кодування методом Віттера

Подальше удосконалення алгоритму динамічного кодування даних нерівномірними кодами *FGK* було запропоноване Д. Віттером. Цей алгоритм одержав назву *алгоритму V*. При пошуку шляхів оптимізації процедури кодування даних кодом Хаффмена автор виходив із того, що в новому алгоритмі число обмінів вузлів у процесі модифікації кодового дерева повинна обмежуватись деяким малим числом (у кращому випадку одиницею), а динамічне хаффменівське дерево повинна будуватися таким чином, щоб мінімізувати не тільки сумарну довжину зовнішнього шляху $\sum W_j \cdot l_j$, але і величина $\sum l_j$, $\max\{l_j\}$. Мінімізація висоти дерева $h = \max\{l_j\}$ дозволить запобігати утворенню довгих кодових комбінацій при кодуванні чергового символа в повідомленні. Віттеру в значній мірі вдалося вирішити поставлену задачу. Розроблений ним алгоритм має в порівнянні з алгоритмом FGK такі дві переваги.

1. Кількість обмінів вузлами, при яких поточний вузол переміщується вгору по кодовому дереві в процесі його модифікації, обмежується одиницею. В алгоритмі FGK верхня границя числа обмінів складає $\frac{l_j}{2}$, де l_j - довжина кодового слова для $Z_j(k+l)$-го символа до початку процедури модифікації.

2. Алгоритм *V* мінімізує довжину зовнішнього шляху дерева l_j, і гарантує дерево мінімальної висоти $h = \max\{l_j\}$ за умови мінімізації сумарної довжини зовнішнього шляху дерева $\sum W_j \cdot l_j$.

Суть удосконалення алгоритму *V* полягає у введені нової системи нумерації вузлів кодового дерева, що одержала назву *неявної нумерації (Implicit numbering)*. При неявній нумерації вузли хаффменівського дерева нумеруються в порядку збільшення по рівнях ліворуч, праворуч і знизу вгору, тобто, вузли більш низького рівня мають номери менші ніж вузли наступного рівня. Найважливішою особливістю неявної нумерації є дотримання необхідної умови побудови дерева.

Для кожної ваги *W* всі зовнішні вузли (листи) дерева з вагою *W* повинні передавати всім внутрішнім вузлам ваги *W*.

Неважко помітити, що ця умова є однією з відмінних рис неявної нумерації у порівнянні з іншими алгоритмами.

Тому, якщо передача продовжиться із символів *d, c, g, e, f* чи з ще не використаних символів, то формовані кодові слова в алгоритмі *V* будуть коротші, ніж у *FGK*, тобто стратегія мінімізації зовнішнього шляху й висоти дерева оптимальна при припущенні, що будь-який символ, який з'являється, рівноімовірний.

Другою відмінною рисою алгоритму *V* є введення поняття блоку еквівалентних вузлів. При цьому вузли *v* та *в* еквівалентні, якщо вони мають однакову вагу і обоє є або внутрішніми, або зовнішніми. Вузол блоку, що має (при неявній нумерації) найвищий номер, називається *лідером блоку.* Блоки впорядковуються за збільшенням ваги, причому блок листків ваги *W* повинен дорівнювати блоку внутрішніх вузлів тієї ж самої ваги.

Третя відмінна риса алгоритму *V* полягає в способі модифікації дерева після одержання чергового символа. Головною операцією алгоритму за підтримкою умови неявної нумерації (*) є ковзання й збільшення (*Slide and Increment*). Суть цієї операції полягає в тому, що вузол, оголошений поточним, обмінюється з лідером свого блоку і потім сковзає в напрямку кореня дерева по сусідньому блоці, що безпосередньо примикає до блоку поточного вузла. Ковзання продовжується доти, поки поточний вузол не пройде весь блок і буде установлений у початок цього блоку. Потім здійснюється збільшення ваги поточного вузла і новим поточним вузлом призначається

батько старого поточного вузла. Операція ковзання із збільшенням продовжується до досягнення кореня дерева. При цьому вибір батьківського вузла залежить від того чи був поточний вузол листком, чи внутрішнім вузлом. Якщо поточний вузол був листком, то новим поточним вузлом стає батько, із яким виявився зв'язаний поточний вузол після завершення ковзання. А у випадку, якщо поточним вузлом був внутрішній вузол, то новим поточним вузлом призначається його батьківський вузол, із яким був зв'язаний поточний до початку ковзання.

2.7.6 Лабораторна робота №11

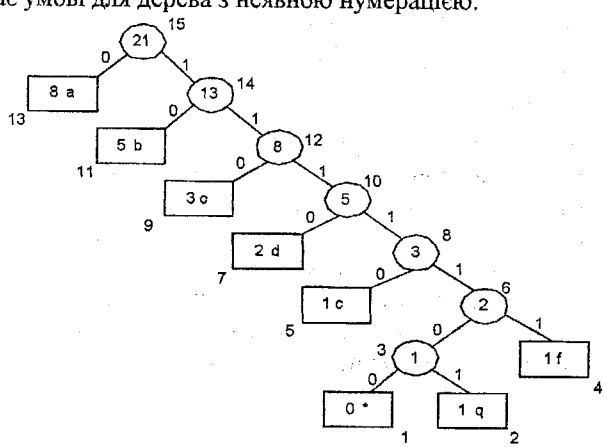
Мета роботи: Ознайомитися з динамічним кодуванням стиснення даних методом Віттера, отримати навики побудови кодового дерева та здійснити стиснення даних.

Хід роботи

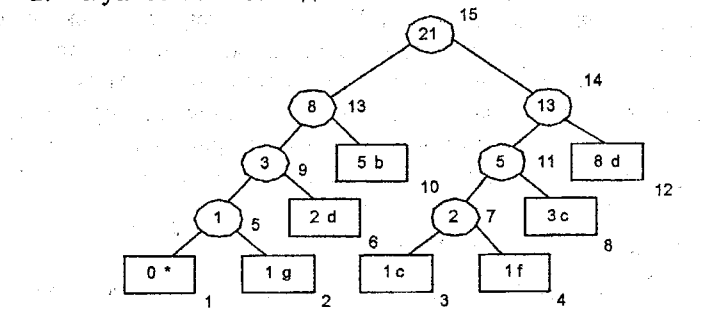
1. Побудувати хаффменівське дерево згідно з варіантом завдання (таблиця 2.15).
2. Модифікувати дерево Хаффмена за методом Віттера з неявною нумерацією. Внутрішній вузол повинен переміщатися з усіма утвореними ним піддеревами, тому що його вага включає ваги дочірніх вузлів.
3. Провести покроково модифікацію й отримати остаточне модифіковане дерево.
4. Згідно з модифікованим деревом Хаффмена отримати кодові комбінації для кожного символа.

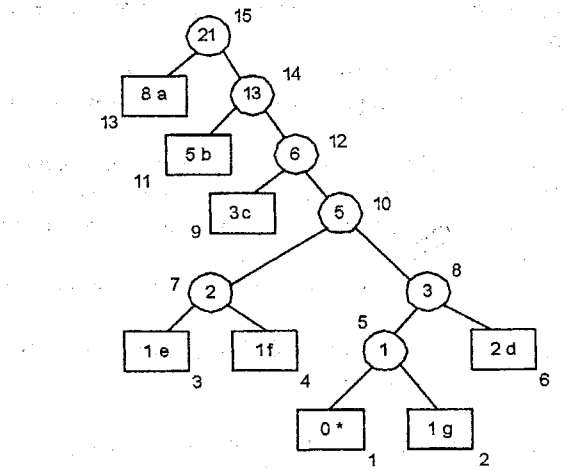
Приклад. Кодування повідомлення методом Віттера

1. Дане кодове дерево відповідає всім вимогам оптимального хаффменівського дерева, але порядок розташування вузлів однієї ваги не відповідає умові для дерева з неявною нумерацією.



2. Вузли 3 та 5 необхідно поміняти місцями.





3. Вузли 6 та 7 необхідно поміняти місцями і т д.
4. Остаточні кодові комбінації символів:

$a \Rightarrow 11;\ b \Rightarrow 101;\ c \Rightarrow 101;\ d \Rightarrow 001;\ f \Rightarrow 1001;\ g \Rightarrow 0001.$

Контрольні запитання

1. В чому суть динамічного кодування?
2. Як будується Хаффменівське дерево при динамічному кодуванні?
3. Яка найважливіша умова якої потрібно дотримуватись при модифікації кодового дерева?
4. Які властивості має Хаффменівське дерево?
5. В чому суть синтезу динамічного кодування методом FGK?
6. В чому суть синтезу динамічного кодування методом Віттера?
7. Дайте порівняльну характеристику методам динамічного кодування.

ЛІТЕРАТУРА

1. Кузьмин И.В., Кедрус В.А. Основы теории информации и кодирования. – Киев.: Вища школа, 1986. – 238с.
2. Бэрри Нанс. Компьютерные сети. – М.: «Бином», 1995. – 396с.
3. В.С. Чернега. Сжатие информации в компьютерных сетях. – Севастополь: «СевГТУ», 1997. – 216с.
4. Й.Й. Білинський. Основи телекомунікацій: Навчальний посібник. – Вінниця: ВДТУ, 1999. – 110с.
5. М. Гук. Аппаратные средства локальных сетей. – СПб.: «Питер КОМ», 2002. – 424с.
6. Ю.И. Рыжиков. Решение научно-технических задач на персональном компьютере. – СПб.: КОРОНА принт, 2000. – 272 с.
7. М.Херхагель, Х. Партоль. MathCAD 2000: Полное руководство. – Киев: «Ирина», BHV, 2000. – 416с.
8. Мартынов Н.Н., Іванов А.П. MATLAB 5.x. Вычисления, визуализация, программирование. – М.: КУДИЦ-ОБРАЗ, 2000. – 336 с.
9. Дьяконов В. Maple 6: Учебный курс. – СПб.: Питер, 2001. – 608 с.
10. Дьяконов В.П. Компьютерная математика. Теория и практика. – М.: Нолидж, 2001. – 1296 с.
11. Рычков В., Дьяконов В., Новиков Ю. Компьютер для студента. – СПб.: Питер, 2000. – 592 с.

Навчальне видання

Йосип Йосипович Білинський
Володимир Вячеславович Мотигін

ОБРОБКА ІНФОРМАЦІЇ ЗАСОБАМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ

Лабораторний практикум

Оригінал-макет підготовлено Мотигіним В.В.
Редактор О.Д. Скалоцька

Навчально-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку 16.09.2005 р.
Формат 29,7×42 1/4.
Друк різографічний
Тираж 75 прим.
Зам. № 2005 – 153

Гарнітура Times New Roman
Папір офсетний
Ум. друк. арк. 6.92

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ