МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙЙИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

КАФЕДРА КОНСТРУЮВАННЯ ЕОА

Звіт

з лабораторної роботи №4

по курсу

«Цифрове оброблення сигналів»

на тему

«Лінійне перетворення сигналів»

Київ 2022

Лабораторна робота №4

Тема. Лінійне перетворення сигналів.

Мета: навчитися створювати сигнали складної форми, використовуючи властивості лінійності.

Хід роботи

Формування складних послідовностей сигналів можна здійснювати за допомогою лінійних математичних операцій - множення на константу, складання і тимчасового зсуву. Як приклад розглянемо формування послідовності прямокутних імпульсів з відповідних гармонійних складових.

Формула, що описує розкладання імпульсного періодичного сигналу на гармонійні складові виглядає наступним чином:

$$S(t) = \frac{A}{2} + \frac{2A}{\pi}(Cos(\frac{2\pi}{T}t) - \frac{1}{3}Cos(3\frac{2\pi}{T}t) + \frac{1}{5}Cos(5\frac{2\pi}{T}t) - ...)$$

Програма, що реалізує цю функцію для восьми гармонійних складових, виглядає наступним чином:

Приклад № 1 (meandr.m)

```
N = 8;                  % Число ненульових гармонік
t = -1 : 0.01 : 1;      % Вектор моментів часу
A = 1;                  % Амплітуда
T = 1;                  % Період
nh = (1 : N) * 2 - 1;   % Номери ненульових гармонік

% Обчислення рядків-гармонік
harmonics = cos(2 * pi * nh' * t / T);
Am = 2 / pi ./ nh;                  % Амплітуди гармонік
Am(2 : 2 : end)= -Am(2 : 2 : end);  % Чергування знаків
s1 = harmonics .* repmat(Am', 1, length(t));

% Формування рядків - часткових сум гармонік
s2 = cumsum(s1);

for k = 1 : N,
    subplot(4, 2, k);
    plot(t, s2(k, :))
end
```

Результат роботи даної програми зображений на рисунку 1.

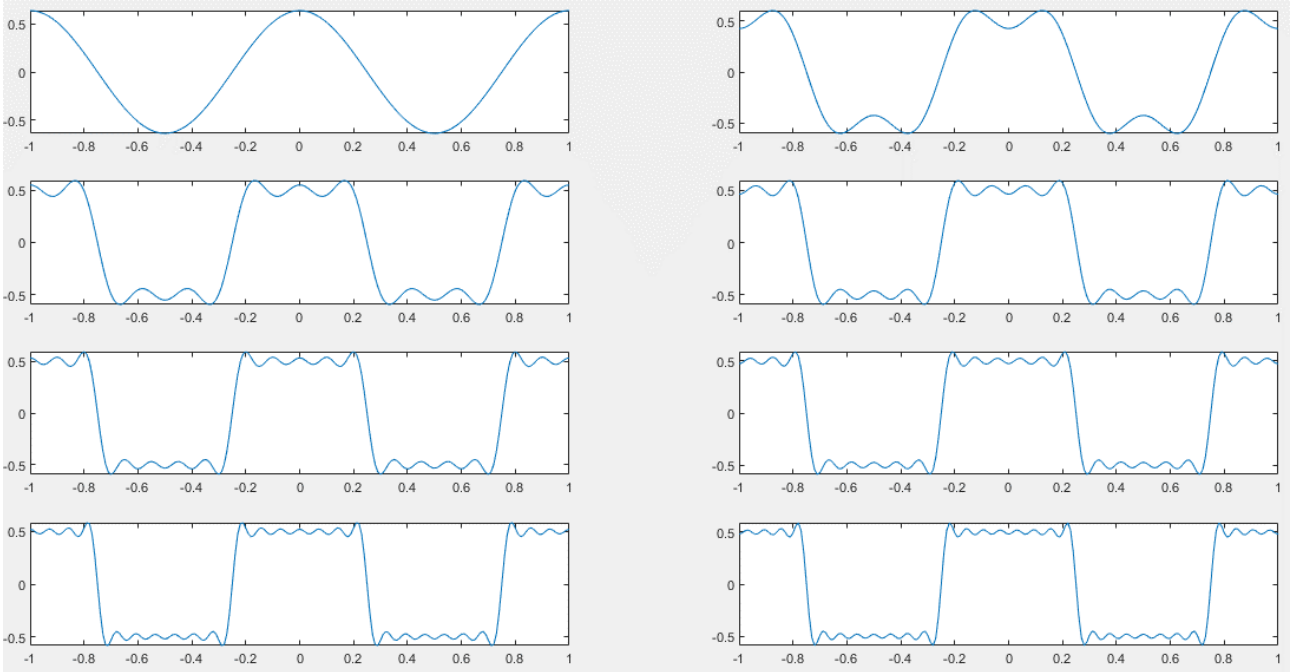

Рисунок 1. Формування імпульсного сигналу з восьми гармонійних складових.

Для повного розуміння принципу роботи програми – подивіться документацію для наступних функцій – repmat і cumsum.

Генерувати періодичні сигнали різної форми можна за допомогою вбудованих функцій середовища MatLab:

square – послідовність прямокутних імпульсів;

sawtooth – послідовність трикутних імпульсів;

diric – функція Діріхле (періодична sinc-функція);

chirp – генерація коливань з мінливою частотою.

Для генерації дискретного білого шуму з нормальним розподілом можна використовувати функцію randn (m, n). В результаті виклику цієї функції генерується масив, що містить m рядків і n стовпців псевдовипадкових чисел, що мають нормальний розподіл з нульовим математичним очікуванням і одиничною дисперсією. Приклад роботи цієї функції показаний нижче:

Приклад № 2 (example2.m)

```
x = randn(1, 10000);
plot(x);
```

Результат генерації випадкового сигналу зображений на рисунку 2.

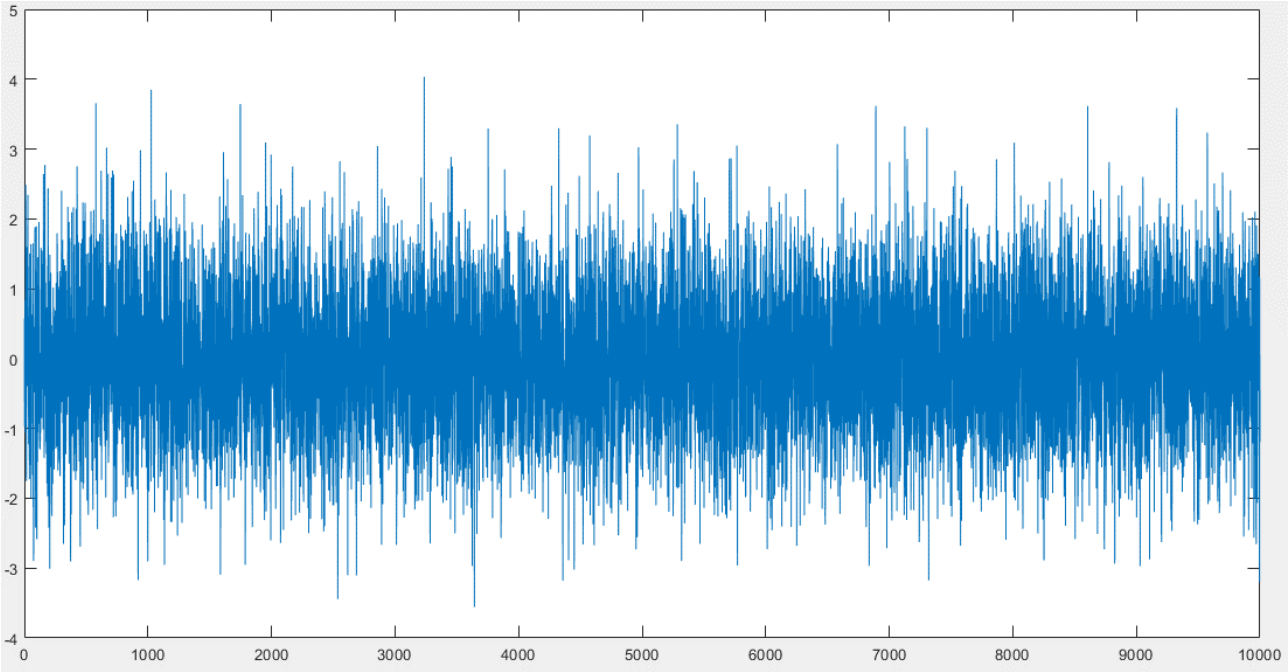

Рисунок 2. Генерація випадкового сигналу.

У пакеті MatLab є ще одна корисна функція – awgn. Вона дозволяє додавати до сигналу білий шум із заданим рівнем. Формат виклику даної функції наступний: y = awgn (x, snr, sigpower, state, 'powertype'), де x – вектор відліку сигналу, snr – скаляр, що задає співвідношення сигнал/шум в одиницях, визначених параметром 'powertype' (за замовчуванням - в децибелах), sigpower – потужність сигналу, state – примусова установка генератора випадкових чисел (останні три параметра не є обов'язковими).

Завдання

1. **Опрацюйте основні приклади, викладені вище, в системі MATLAB.**
2. **Створіть М-файл, який реалізує наступні сигнали з відповідних гармонійних складових:**

– Напишіть програму, що синтезує пилкоподібний сигнал з 10 гармонійних складових. Функція для синтезу описується наступним виразом:

$$S(t) = \frac{2A}{\pi}(Sin(\frac{2\pi}{T}t) - \frac{1}{2}Sin(2\frac{2\pi}{T}t) + \frac{1}{3}Sin(3\frac{2\pi}{T}t) - \frac{1}{4}Sin(4\frac{2\pi}{T}t) + ...)$$

(task2.m)

```
N = 10;              % Число ненульових гармонік
t = -1 : 0.01 : 1;   % Вектор моментів часу
A = 1;               % Амплітуда
T = 1;               % Період
nh = (1 : N);        % Номери ненульових гармонік

% Обчислення рядків-гармонік
harmonics = sin(2 * pi * nh' * t / T);
Am = 2 * A / pi ./ nh;               % Амплітуди гармонік
Am(1 : 1 : end) = -Am(1 : 1 : end); % Чергування знаків
s1 = harmonics .* repmat(Am', 1, length(t));

% Формування рядків - часткових сум гармонік
s2 = cumsum(s1);
for k = 1 : N,
    subplot(N / 2, 2, k);
    plot(t, s2(k, :))
end
```

Результат роботи програми:

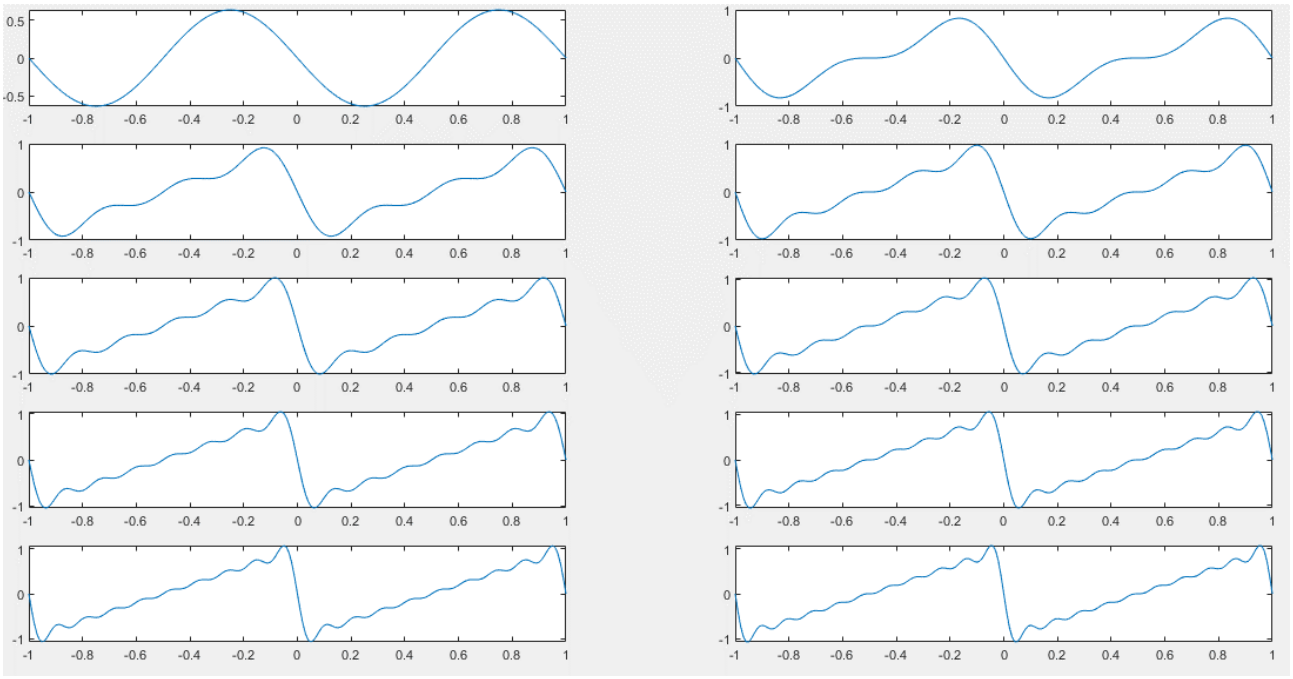

Рисунок 2.1

– Збільшіть кількість гармонійних складових. Як це вплинуло на сигнал, що синтезується?

$N = 16$

Результат роботи програми:

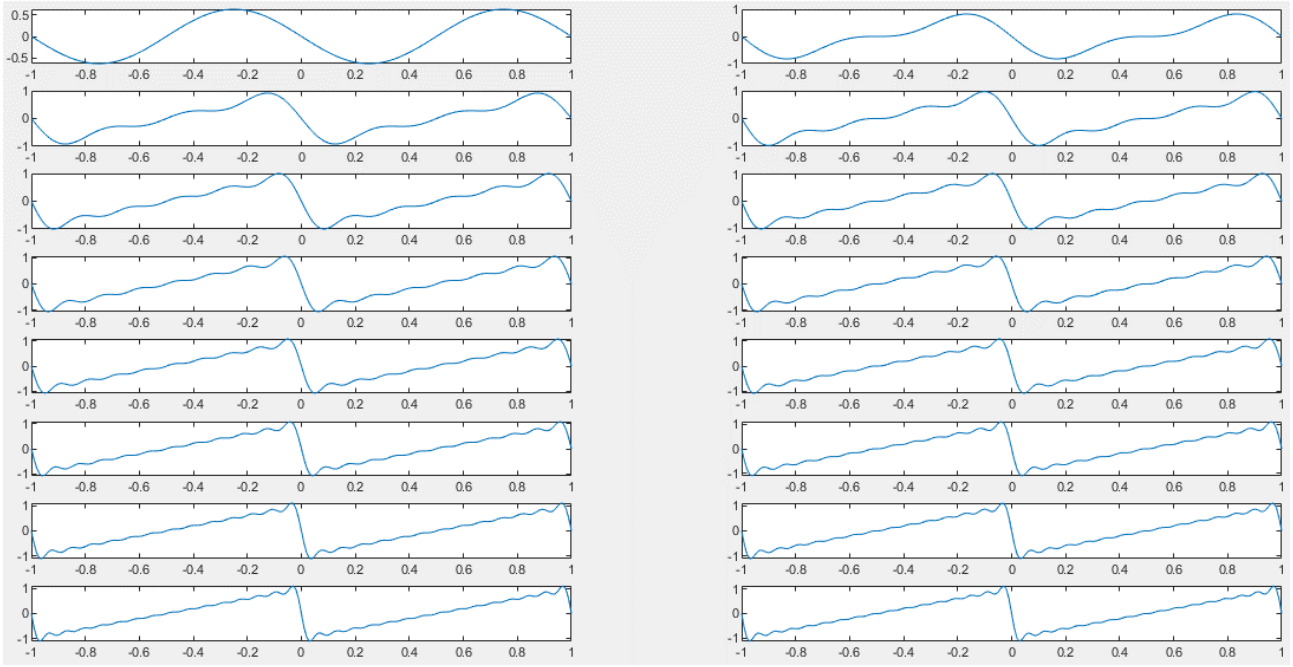

Рисунок 2.2

Бачимо, що при збільшенні кількості гармонійних складових хвилі стають майже непомітні.

– Напишіть програму, що синтезує трикутний сигнал з 8 гармонійних складових. Функція для синтезу описується наступним виразом:

$$S(t) = \frac{8A}{\pi^2}(Cos(\frac{2\pi}{T}t) + \frac{1}{3^2}Cos(3\frac{2\pi}{T}t) + \frac{1}{5^2}Cos(5\frac{2\pi}{T}t) + ...)$$

```
N = 8;                      % Число ненульових гармонік
t = -1 : 0.01 : 1;          % Вектор моментів часу
A = 1;                      % Амплітуда
T = 1;                      % Період
nh = (1 : N) * 2 - 1;       % Номери ненульових гармонік

% Обчислення рядків-гармонік
harmonics = cos(2 * pi * nh' * t / T);
Am = 8 * A / (pi .^ 2) ./ (nh .^ 2);    % Амплітуди гармонік
s1 = harmonics .* repmat(Am', 1, size(t, 2));

% Формування рядків - часткових сум гармонік
s2 = cumsum(s1);
for k = 1 : N,
    subplot(N / 2, 2, k);
    plot(t, s2(k, :))
end
```

Результат роботи програми:

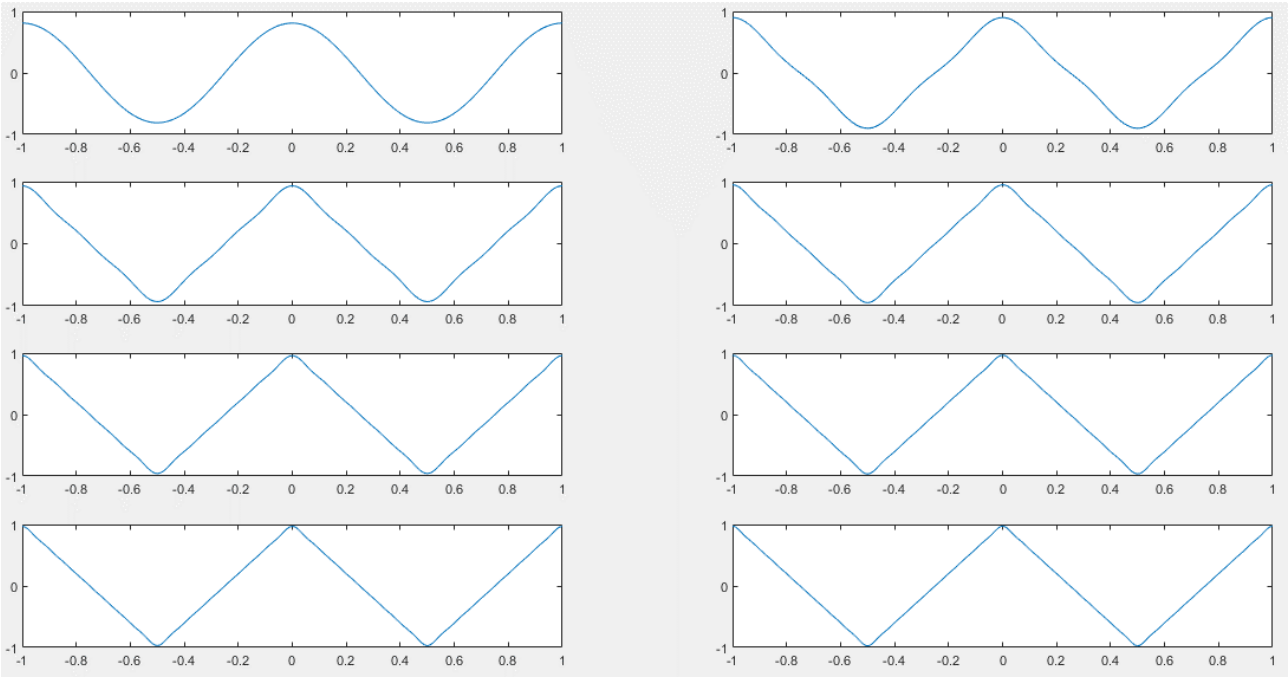

Рисунок 2.3

– Збільште кількість гармонійних складових. Як це вплинуло на сигнал, що синтезується?

$N = 16$

Результат роботи програми:

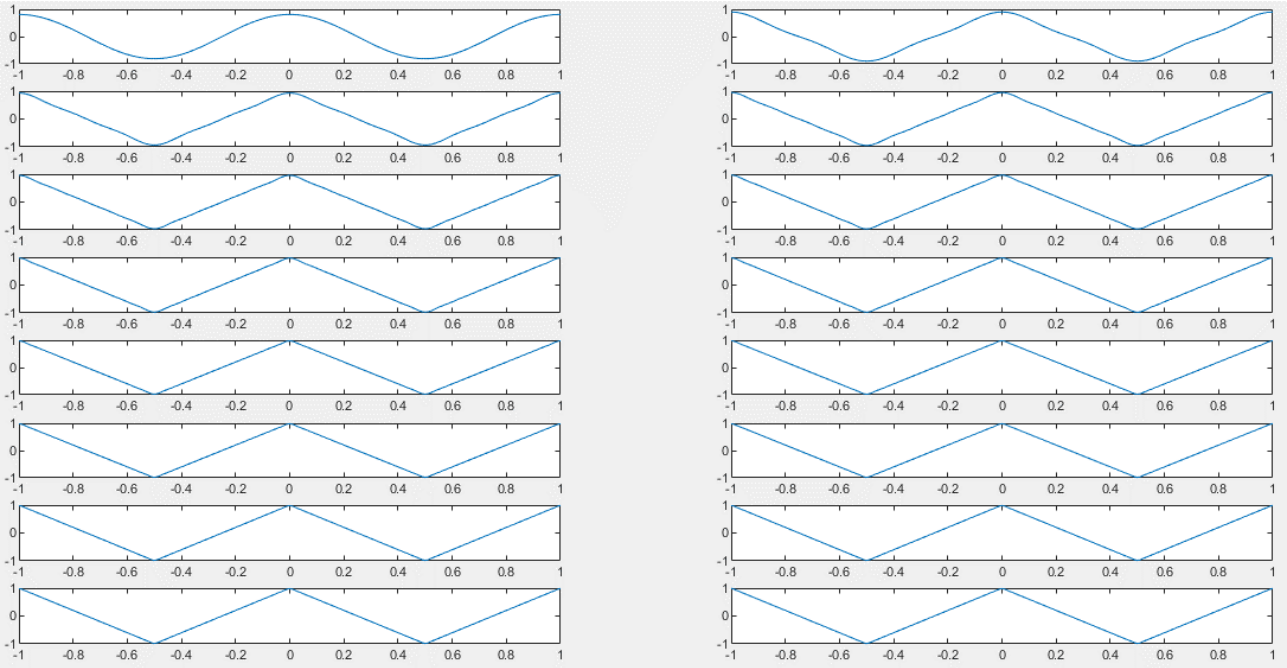

Рисунок 2.4

Бачимо, що при збільшенні кількості гармонійних складових хвилі стають майже непомітні.

4. **Для випадкового сигналу, що генерується за допомогою функції randn (m, n) створіть М-файл, в якому: генерувався б сигнал, для нього розраховувалося середнє значення, СКО і будувалася гістограма.**

   (task4.m)

```
x = randn(1, 10000);

m = mean(x)
s = std(x)

subplot(1, 2, 1);
plot(x);
subplot(1, 2, 2);
hist(x);
```

Результат роботи програми:

```
>> task4

m =

   -0.0077


SD =

    1.0040
```

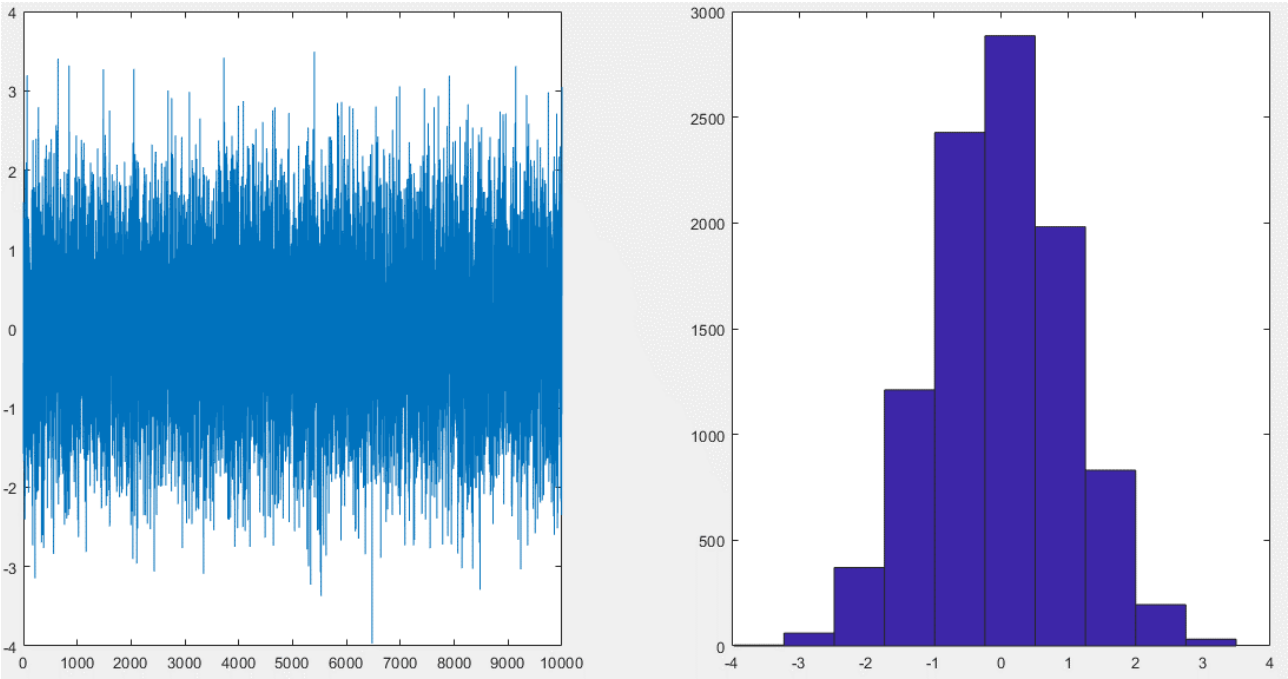

Рисунок 4.1

5. **Створіть М-файл, в якому до згенерованого періодичного сигналу додавався випадковий шум. Обчисліть статистичні характеристики такого сигналу.**

(task5.m)

```
function [y, xi] = task5()
xi = 0 : 0.2 : 16;
x = @sin;
y = x(xi) + randn(1, length(xi)) / 5;
plot(y)
end
```

Результат роботи програми:

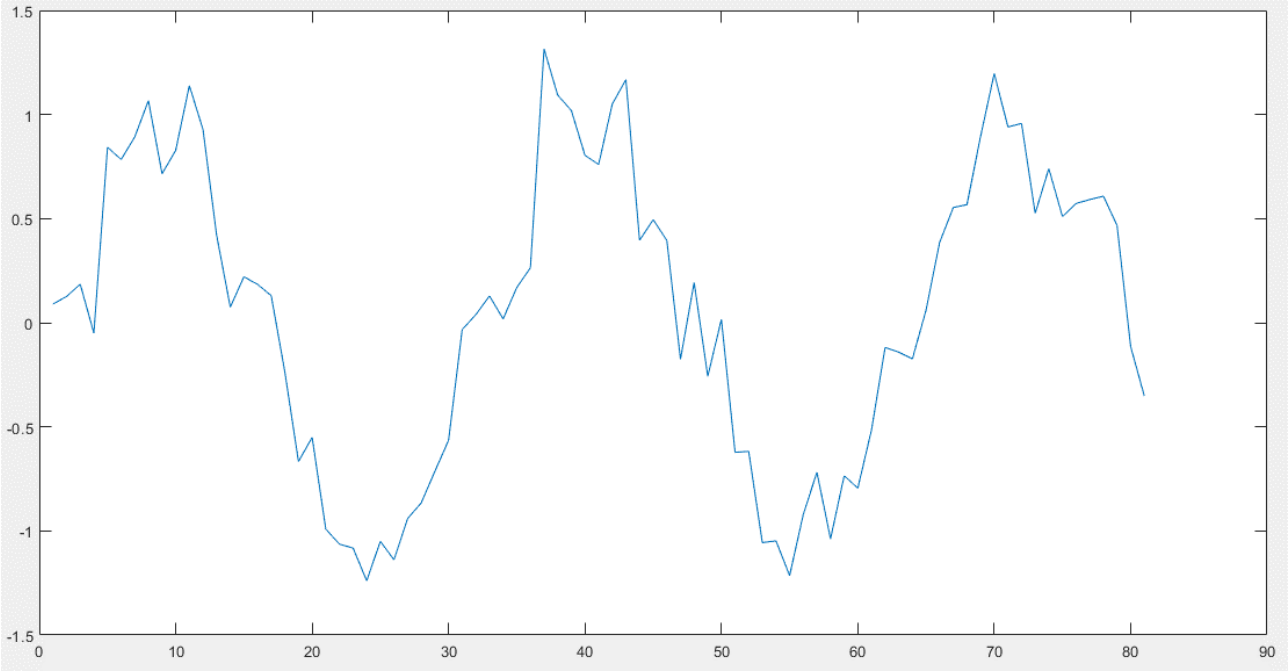

Рисунок 5.1

6. **Створіть М-файл, в якому до гармонійного сигналу додається білий шум в заданому співвідношенні (використовуйте для цього функцію awgn).**

(task6.m)

```
t = (0 : 0.1 : 10)';
x = sin(t);
y = awgn(x, 5, 'measured');
plot(t, [x y])
```

Результат роботи програми:

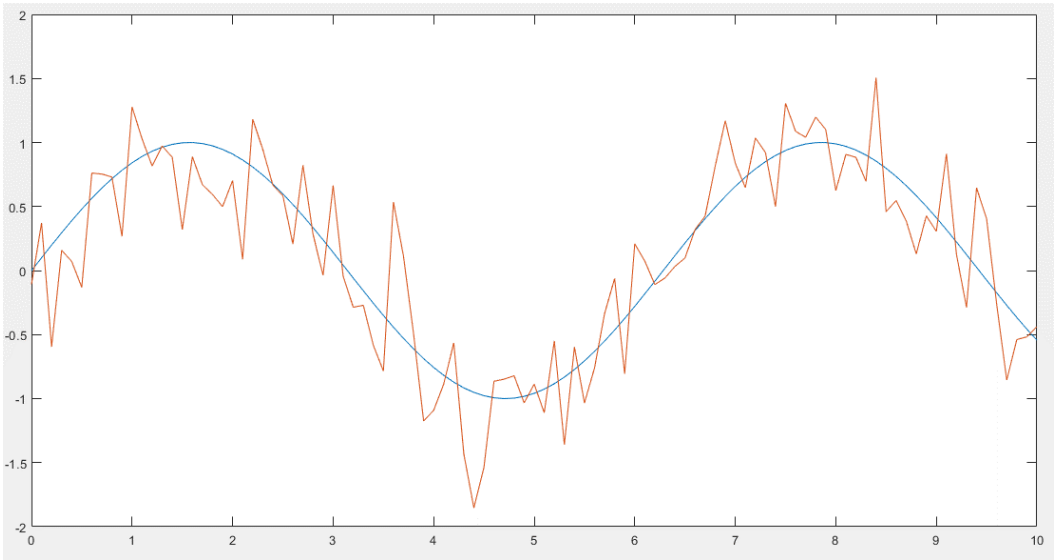

Рисунок 6.1

7. Відкрийте програму, що реалізовує імпульсне розкладання сигналу (**decomp.m**). Вихідний сигнал - один період синусоїдального сигналу, представлений шістнадцятьма відліками. Програма виконує розкладання вихідного сигналу на шістнадцять імпульсних послідовностей (амплітуда і положення імпульсу в кожній послідовності залежить від значення відповідного відліку в вихідному сигналі).

```
t = 0 : pi / 8 : 2 * pi;     % Period - 16 points
y = sin(t);                  % Ouer signal
N = length(y);               % Namber of pulse

for IC = 1 : N               % Pulse decomposition
    decY{IC} = y;
    decY{IC}(1 : IC - 1) = 0;
    decY{IC}(IC + 1 : N) = 0;
end
stem(t, y);
grid;
figure
for k = 1 : N - 1
    subplot(4, 4, k);
    stem(decY{1, k})
end
```

Результати роботи даної програми зображені на рисунках нижче.

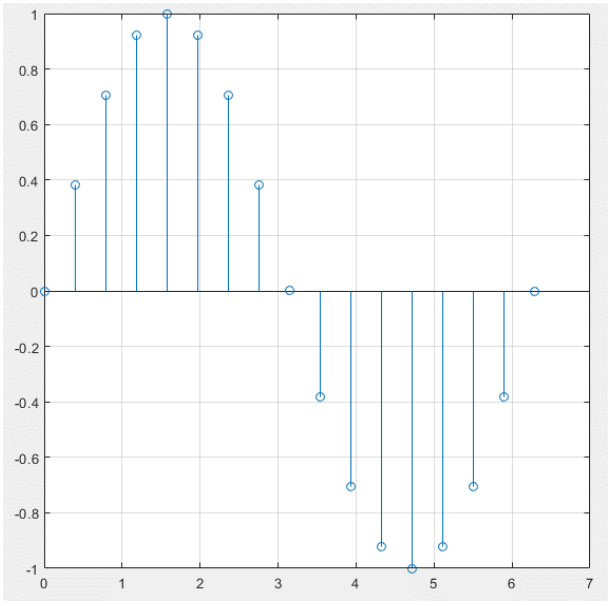

Рисунок 7.1a

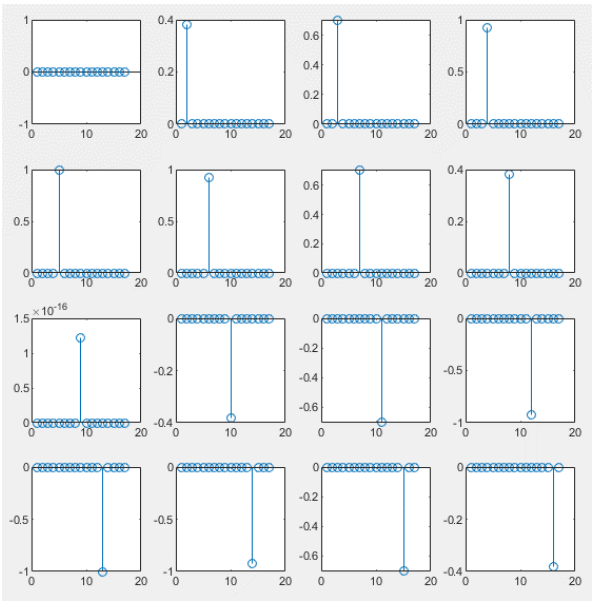

Рисунок 7.1b

На першому рисунку – вихідний сигнал. На другому – результат розкладання сигналу на імпульсну послідовність.

Завдання

1) Розробіть програму, що виконує для вихідного сигналу розкладання на ступінчасті функції (сигнал типу «скачок»). Перевірте коректність роботи програми.

(task7_1.m)

```
mas = cos(0 : 3 * pi / 15 : 3 * pi)
d = zeros(16, 16);
t = 0 : 1 : 15;

subplot(6, 3, 1);
plot(t, mas);
xlim([0; 15]);
ylim([-1; 1]);

for i = 1 : 16
    for j = 1 : 16
        if(i ~= 1)
            if(i < j)
                d(i, j) = mas(i) - mas(i - 1);
            else
                d(i, j) = 0;
            end
        else
            d(i, j) = mas(i);
        end

    end

    subplot(6, 3, i + 1);
    plot(t, d(i, :));
    xlim([0; 15]);
    ylim([-1; 1]);
end
```

Результат роботи програми:

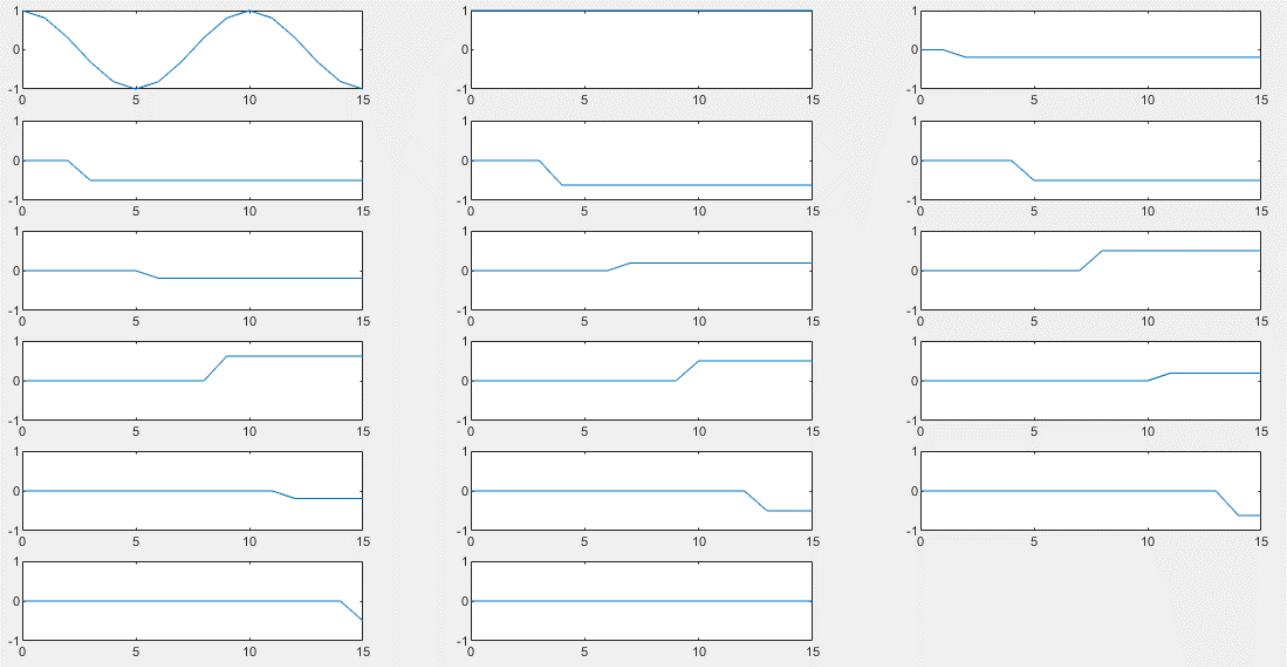

Рисунок 1

2) Розробіть програму, що виконує для вихідного сигналу розкладання на дві послідовності – з парною і непарною симетрією. Перевірте коректність роботи програми.

(task7_2.m)

```
t = 0 : 1 / 15 : 1;

sig = cos(2 * pi * 2 * t)
symmetry_even = zeros(1, 16);
symmetry_odd  = zeros(1, 16);
Ymin = min(sig) - 1;
Ymax = max(sig) + 1;

for i = 1 : 16
   even_symmetry(i) = (sig(i) + sig(17 - i)) / 2;
   odd_symmetry(i)  = (sig(i) - sig(17 - i)) / 2;
end

subplot(3, 1, 1);
plot(t, sig);
axis([0 1 Ymin Ymax]);
grid on;

subplot(3, 1, 2);
plot(t, even_symmetry);
xlabel('Even Symmetry')
axis([0 1 Ymin Ymax]);
grid on;

subplot(3, 1, 3);
plot(t, odd_symmetry);
xlabel('Odd Symmetry')
axis([0 1 Ymin Ymax]);
grid on;
```

Результат роботи програми:

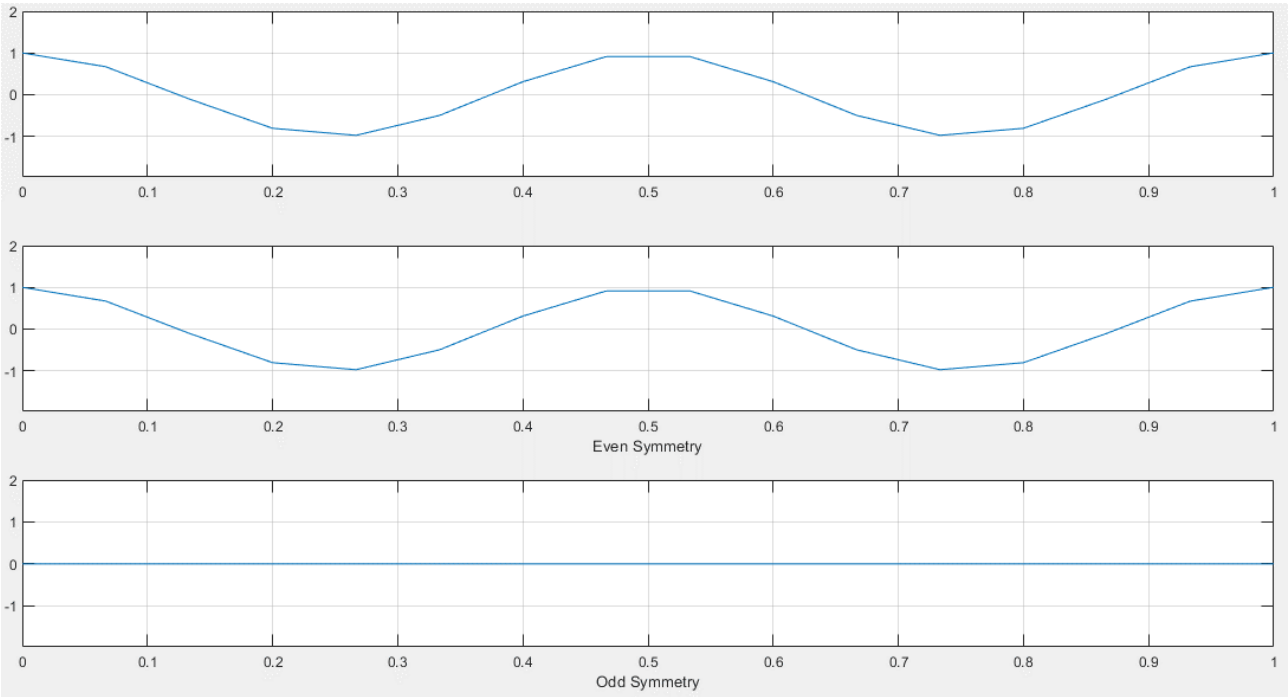

Рисунок 2

3) Розробіть програму, що виконує для вихідного сигналу розкладання на дві послідовності, що чергуються – з парними і непарними відліками. Перевірте коректність роботи програми.

(task7_3.m)

```
mas = sin(0 : 3 * pi / 15 : 3 * pi)
x = 0 : 1 : 15;
subplot(3, 1, 1);
plot(x, mas);
xlim([0; 15]);
ylim([-1; 1]);
grid on;
even = zeros(1, 16);
odd = zeros(1, 16);
for i = 1 : 16
    if(mod(i, 2) ~= 0)
        odd(i) = mas(i);
    else
        even(i) = mas(i);
    end
end

subplot(3, 1, 2);
plot(x, even);
xlabel('Even')
xlim([0; 15]);
ylim([-1; 1]);
grid on;

subplot(3, 1, 3);
plot(x, odd);
xlabel('Odd')
xlim([0; 15]);
ylim([-1; 1]);
grid on;
```

Результат роботи програми:

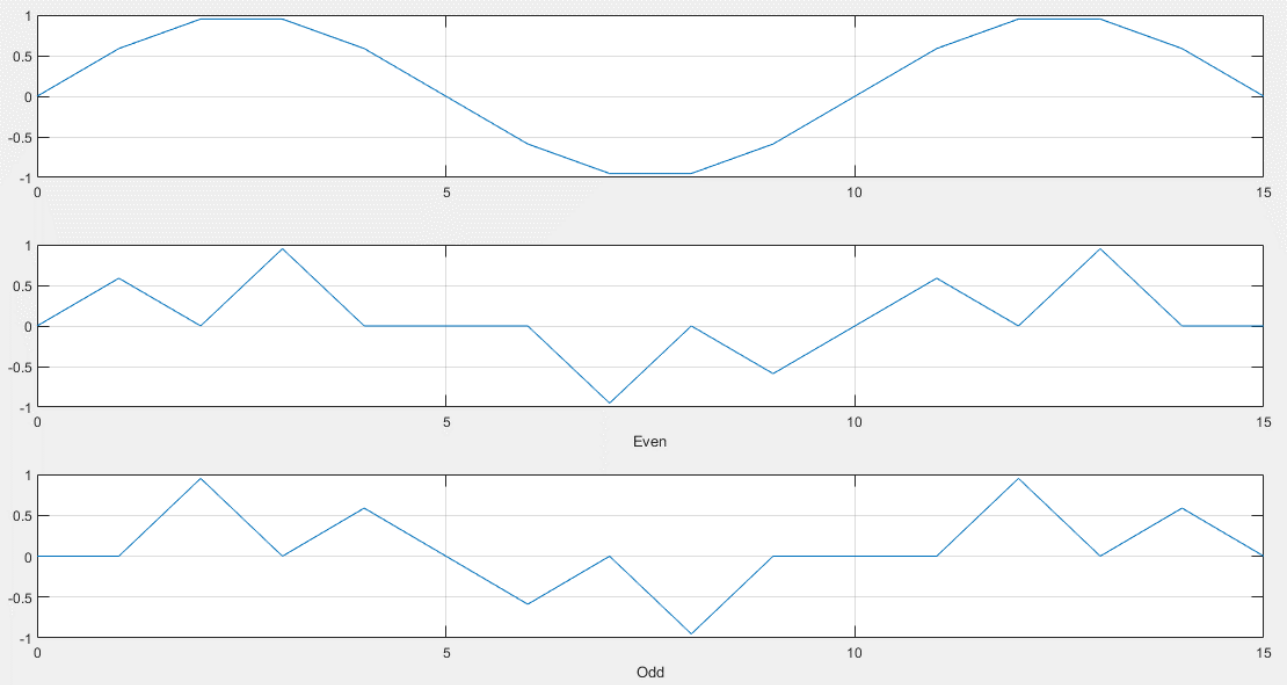

Рисунок 3