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

КАФЕДРА КОНСТРУЮВАННЯ ЕОА

Звіт

з лабораторної роботи №1

по курсу

«Цифрове оброблення сигналів»

на тему

«Основи роботи в системі MATLAB»

Київ 2022

Лабораторна робота №1

Тема. Основи роботи в системі MATLAB.

Мета: знайомство з основними командами системи MATLAB.

Хід роботи

Введення Матриць.

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

A =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

Операції додавання елементів, транспортування та діагоналізації матриці.

Сума стовпців:

```
>> sum(A)

ans =

    34    34    34    34
```

Транспонування:

```
>> A'

ans =

    16     5     9     4
     3    10     6    15
     2    11     7    14
    13     8    12     1
```

Сума в рядах:

```
>> sum(A')'

ans =

    34
    34
    34
    34
```

Діагональ:

```
>> diag(A)

ans =

    16
    10
     7
     1
```

Сума по діагоналі:

```
>> sum(diag(A))

ans =

    34
```

Індекси.

```
>> A(1,4) + A(2,4) + A(3,4) + A(4,4)

ans =

    34

>> X = A;
>> X(4, 5) = 17

X =

    16     3     2    13     0
     5    10    11     8     0
     9     6     7    12     0
     4    15    14     1    17
```

Оператор двокрапка.

Інтервал від 1 до 10:

```
>> 1:10

ans =

     1     2     3     4     5     6     7     8     9    10
```

Зворотній інтервал з приростом:

```
>> 100:-7:50

ans =

   100    93    86    79    72    65    58    51
```

```
>> 0:pi/4:pi

ans =

         0    0.7854    1.5708    2.3562    3.1416
```

Сума елементів в останньому стовпці матриці:

```
>> sum(A(:,end))

ans =

    34
```

Функція Magic.

Створення магічного квадрату:

```
>> B = magic(5)

B =

    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

Змінні.

Створення змінної зі значенням:

```
>> num_students = 50

num_students =

    50
```

Функції.

Список всіх елементарних математичних функцій: *help elfun*

Список складніших математичних та матричних функцій: *help specfun, help elmat*

Зміна значення функції та встановлення її початкового значення:

```
>> eps = 2.e-6

eps =

   2.0000e-06

>> clear eps
```

Вирази.

```
>> rho = (1+sqrt(5))/2

rho =

    1.6180

>> a = abs(3+4i)

a =

     5

>> z = sqrt(besselk(4/3,rho-i))

z =

   0.3730 + 0.3214i

>> huge = exp(log(realmax))

huge =

  1.7977e+308

>> toobig = pi*huge

toobig =

   Inf
```

Генерування матриць.

```
>> Z = zeros(2,4)

Z =

     0     0     0     0
     0     0     0     0

>> F = 5*ones(3,3)

F =

     5     5     5
     5     5     5
     5     5     5

>>  N = fix(10*rand(1,10))

N =

     8     9     1     9     6     0     2     5     9     9
```

```
>> R = randn(4,4)

R =

   -1.3499    0.7147    1.4090    0.7172
    3.0349   -0.2050    1.4172    1.6302
    0.7254   -0.1241    0.6715    0.4889
   -0.0631    1.4897   -1.2075    1.0347
```

Завантаження матриць.

Завантаження з файлу:

```
>> load magik.dat
>> magik

magik =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

Об’єднання.

Об’єднання матриць:

```
>> B = [A A+32; A+48 A+16]

B =

    16     3     2    13    48    35    34    45
     5    10    11     8    37    42    43    40
     9     6     7    12    41    38    39    44
     4    15    14     1    36    47    46    33
    64    51    50    61    32    19    18    29
    53    58    59    56    21    26    27    24
    57    54    55    60    25    22    23    28
    52    63    62    49    20    31    30    17
```

Видалення рядків та стовпчиків.

```
>> X  =  A;
>> X(:,2) = []

X =

    16     2    13
     5    11     8
     9     7    12
     4    14     1
```

```
>> X(2:2:10) = []

X =

    16     9     2     7    13    12     1
```

Перемноження матриць.

```
>> A

A =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1

>> B

B =

    16     4     4    13
     5    -7     2    12
     0    -9    23    65
   -10     4    14     9

>> C = A * B

C =

         141          77         298         491
          50        -117         405         972
          54         -21         377         752
         129        -211         382        1151
```

Поелементне перемноження:

```
>> C = A.*B

C =

   256    12     8   169
    25   -70    22    96
     0   -54   161   780
   -40    60   196     9
```

Створення М-файлів

Зміст файлу average.m з функцією обчислення середнього значення:

```
function y = average (x)
% AVERAGE Середнє значення елементів вектору.
% AVERAGE(X), де X - вектор. Обчислює середнє значення елементів
% вектору.
% Якщо вхідний аргумент не є вектором, генерується помилка.
[m, n] = size(x);
   if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
    error('Вхідний масив має бути вектором')
   end
        y = sum(x) / length(x); %  Власне обчислення

```

Виклик функції з М-файлу:

```
>> z = 1:99;
>> average(z)

ans =

    50
```

Порядок виконання роботи

1. Відкрийте програму, яка генерує синусоїдальний сигнал з частотою 6 Гц (Sin_Statistic.m). Програма генерує синусоїдальний сигнал з частотою дискретизації 240 Гц на інтервалі 1с. Амплітуда генерується сигналу задається змінною A, постійний зсув - змінною OffSet. Для заданого сигналу розраховується середнє значення mean і середньоквадратичне відхилення - теоретичне і справжнє.

```
fs = 240;        % Sample frequency
t = 0:1/fs:1;    % Time interval 1s
A = 1.25;        % Amplitude
OffSet = 1.33;   % Offset
x = A*sin(2 * pi * t * 6) + OffSet;     % Generating Sin signal
m = mean(x);                            % Calculating mean
SD_Teor = (2 * A) / (2 * sqrt(2))       % Theoretical stanard deviation
SD_Real = std(x)                        % Real stanard deviation
plot(t, x);
grid;
title('Sin Function');
xlabel('Time [S]');
ylabel('Amplitude');
legend(sprintf('mean = %.2f', m));
```

Результатом роботи даної програми буде сигнал, представлений на рисунку:

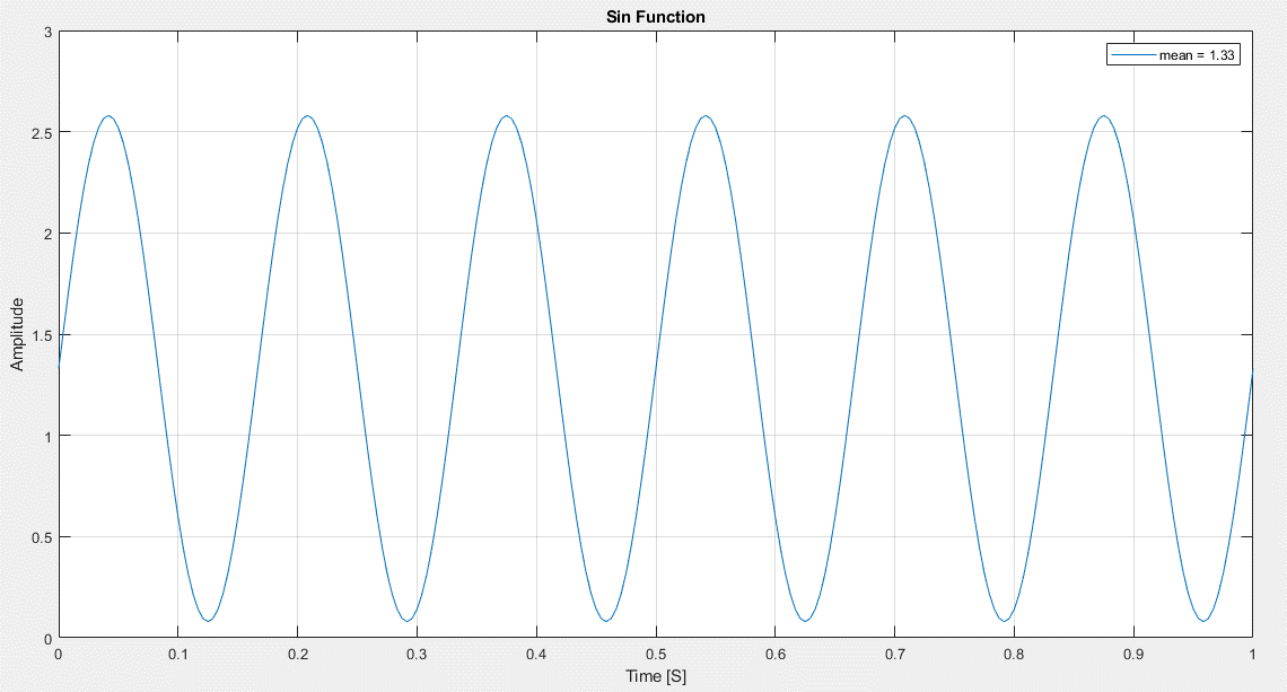


Рисунок 1. Розраховане середнє значення відображається на графіку, а СКВ - в командному вікні середовища MatLab

```
>> Sin_Statistic

SD_Teor =

    0.8839


SD_Real =

    0.8839
```

Завдання

1) Змініть значення амплітуди, частоти сигналу і постійного зміщення. Перевірте отримані нові результати середнього значення і СКВ.

   fs = 960

   A = 2

   OffSet = 3.05

Результат роботи даної програми:

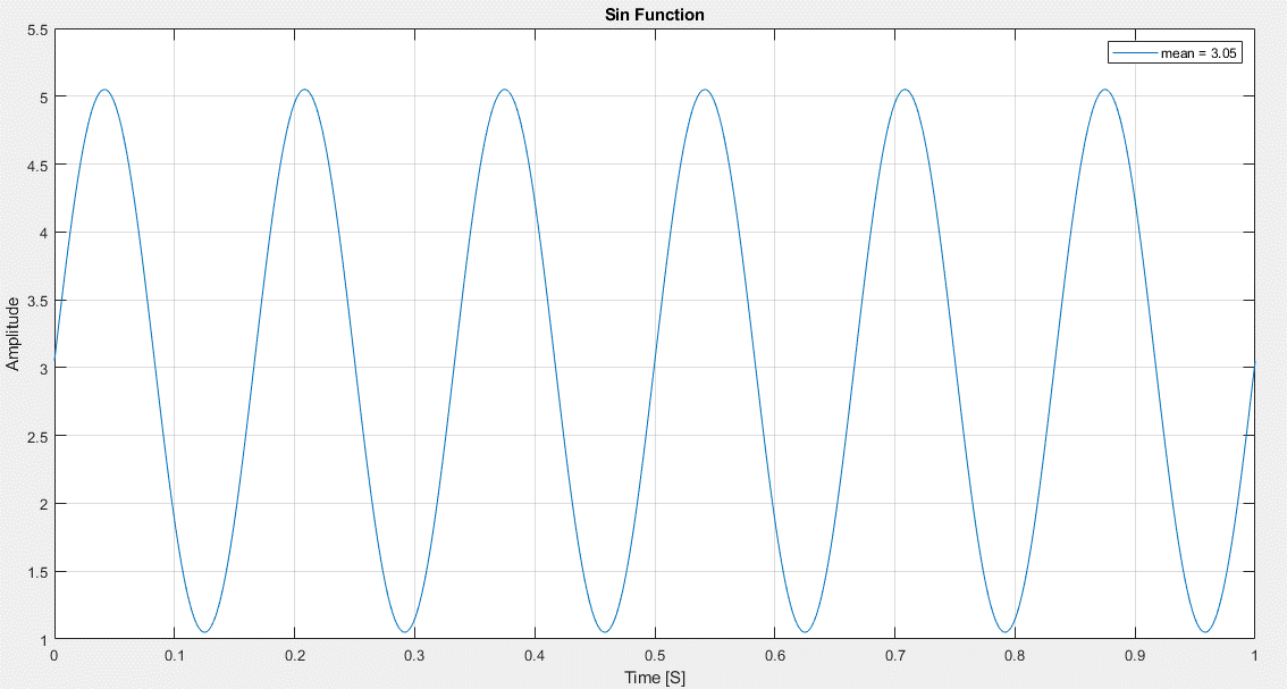


Рисунок 1.1

```
>> Sin_Statistic

SD_Teor =

    1.4142


SD_Real =

    1.4142
```

2) Згенеруйте послідовність прямокутних імпульсів (використовуючи функцію square). Задайте для неї амплітуду, частоту і зміщення. Розрахуйте середнє значення і СКВ.

x = A * square(2 * pi * t * 6) + OffSet

СКВ розраховується за формулою:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x_i - \mu)^2 \quad (1.1)$$

$$\mu = \frac{1}{N}\sum_{i=0}^{N-1} x_i \quad (1.2)$$

Результат роботи даної програми:

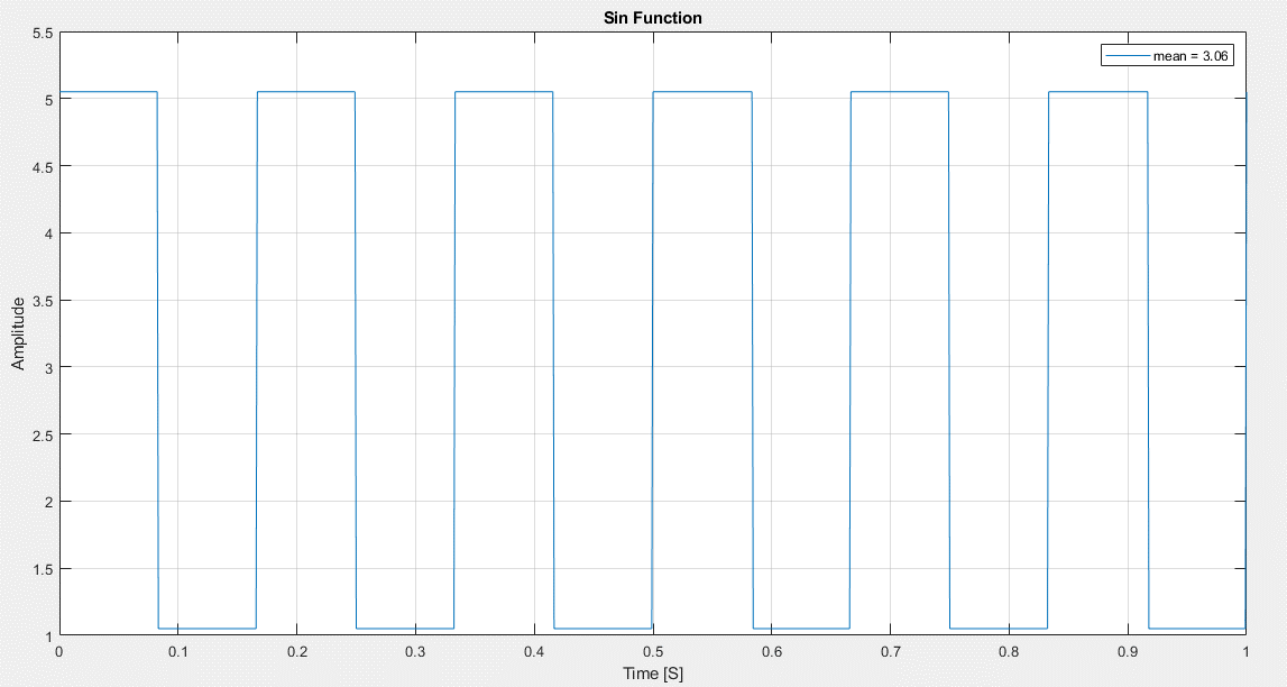


Рисунок 1.2

```
>> Sin_Statistic

SD_Teor =

    2.0021


SD_Real =

    2.0010
```

3) Згенеруйте послідовність трикутних імпульсів (використовуючи функцію sawtooth). Задайте для неї амплітуду, частоту і зміщення. Розрахуйте середнє значення і СКВ.

   x = A * sawtooth (2 * pi * t * 6) + OffSet

   СКВ розраховується за формулою (1.1).

Результат роботи даної програми:

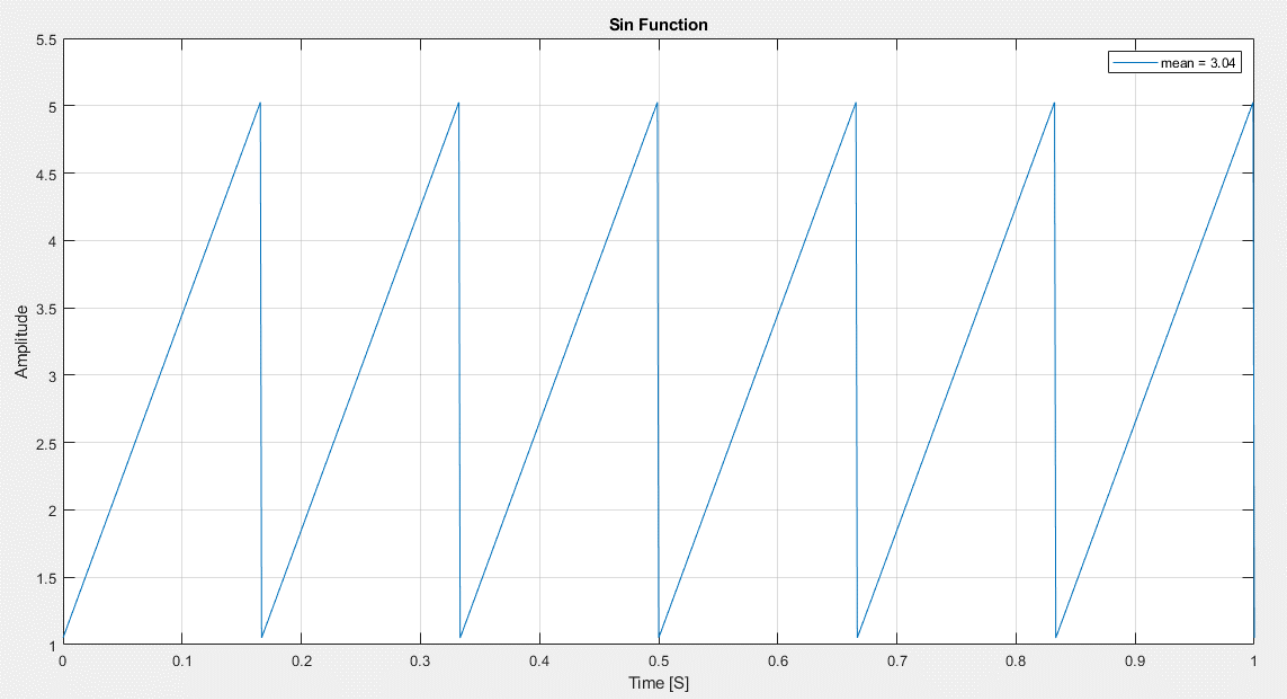


Рисунок 1.3

```
>> Sin_Statistic

SD_Teor =

    1.1571


SD_Real =

    1.1565
```

2. Відкрийте програму, яка генерує дві псевдовипадкові послідовності з нормальним і рівномірним розподілом **(Noise.m)**, які містять 1024 відліку. Для заданих послідовностей, за допомогою функції hist, формується графік розподілу значень в послідовності.

```
% Generates Uniformly and Normally Distributed random signals
N = 1024;            % Define Number of samples
R1 = randn(1, N);    % Generate Normal Random Numbers

R2 = rand(1, N);     % Generate Uniformly Random Numbers
figure(1);           % Select the figure
subplot(2, 2, 1);    % Subdivide the figure into 4 quadrants
plot(R1);            % Plot R1 in the first quadrant
grid;
title('Normal [Gaussian] Distributed Random Signal');
xlabel('Sample Number');
ylabel('Amplitude');
subplot(2, 2, 2);    % Select the second qudrant
hist(R1);            % Plot the histogram of R1
grid;
title('Histogram [Pdf] of a normal Random Signal');
xlabel('Sample Number');
ylabel('Total');
subplot(2, 2, 3);
plot(R2);
grid;
title('Uniformly Distributed Random Signal');
xlabel('Sample Number');
ylabel('Amplitude');
subplot(2, 2, 4);
hist(R2);
grid;
title('Histogram [Pdf] of a uniformly Random Signal');
xlabel('Sample Number');
ylabel('Total');
```

Результатом роботи даної програми будуть сигнали, представлені на малюнку:

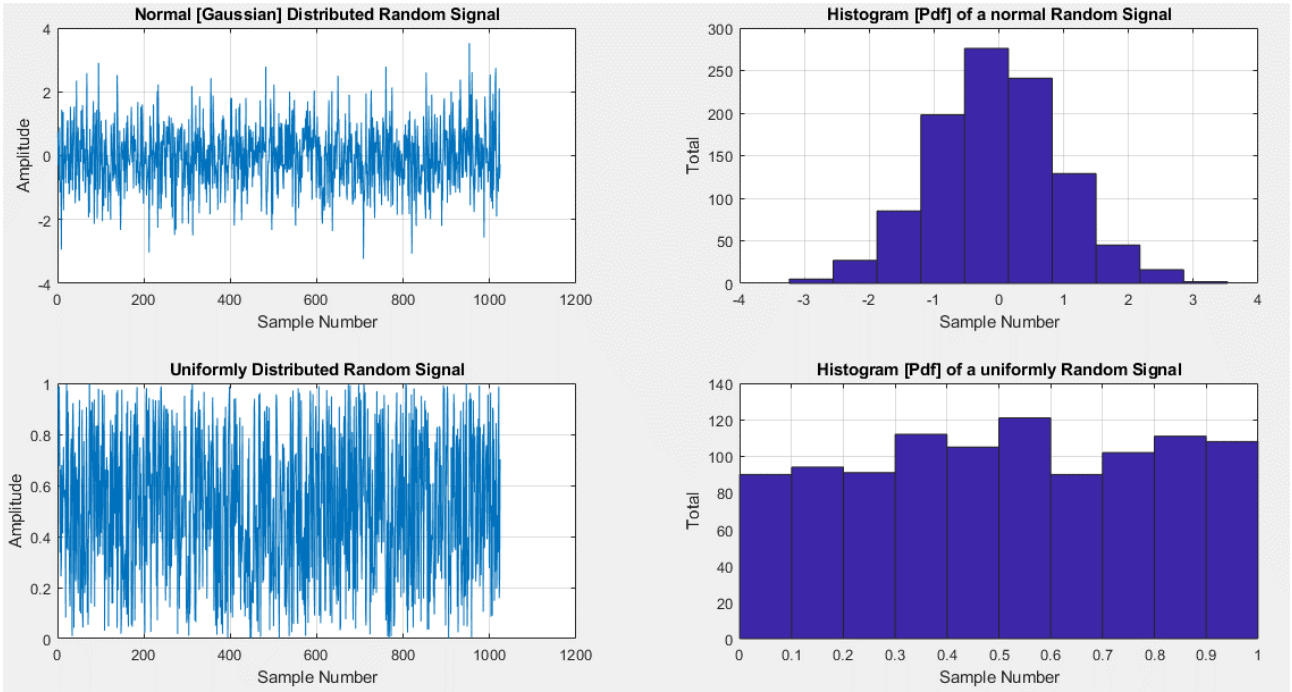


Рисунок 2

Завдання

1) Визначте для випадкових послідовностей середнє значення і СКВ.

   m1 = sum(R1) / N

   SD1 = sqrt(sum((R1 - m).^2) / (N - 1))

   m2 = sum(R2) / N

   SD2 = sqrt(sum((R2 - m).^2) / (N - 1))

```
>> Noise

m1 =

    0.0175


SD1 =

    1.0210


m2 =

    0.4913


SD2 =

    0.3931
```

2) Згенеруйте псевдовипадкову послідовність для 128 точок з рівномірним розподілом в діапазоні від -π до π. Визначте для цієї послідовності середнє значення і СКВ.

```
N = 128;
R = (2 * pi).*rand(1, N) - pi;
m = sum(R)/N;
SD = sqrt(sum((x - m).^2) / (size(x, 2) - 1));

figure(1);
subplot(1, 2, 1);
plot(R);
grid;
title('Uniformly Distributed Random Signal');
xlabel('Sample Number');
ylabel('Amplitude');
subplot(1, 2, 2);
hist(R);
title('Histogram [Pdf] of uniformly random signal');
xlabel('Sample Number');
ylabel('Total');
```

Результат роботи даної програми:

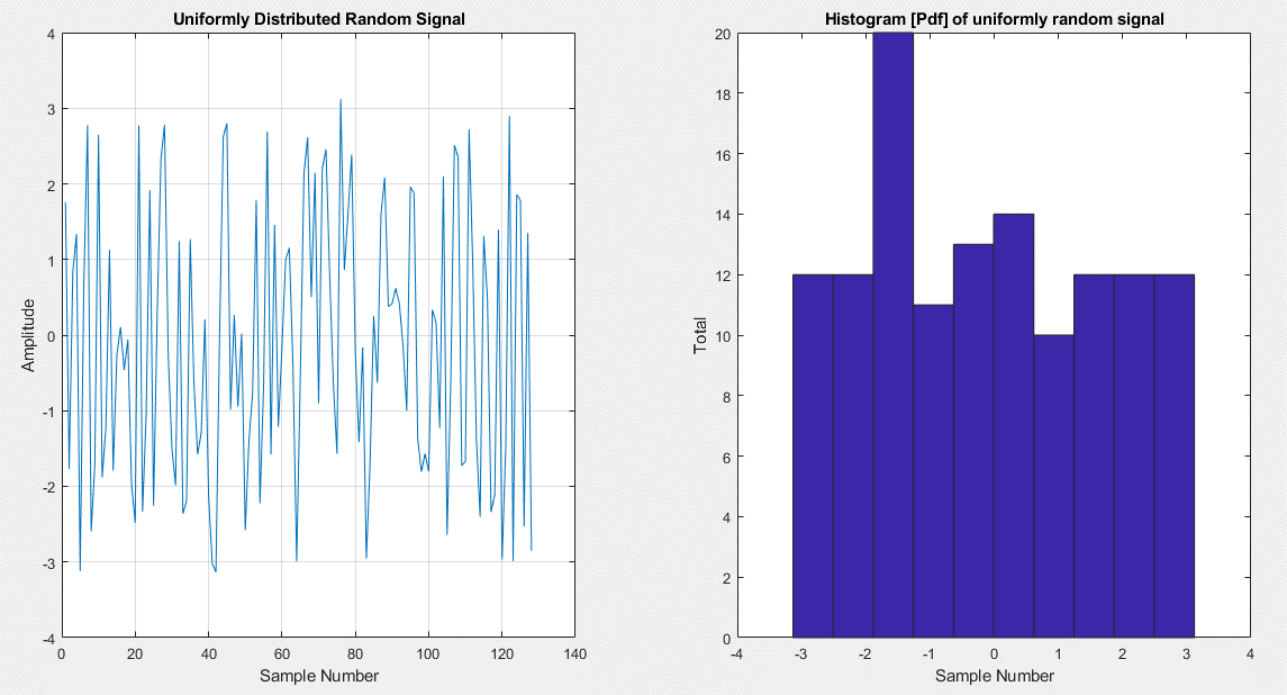


Рисунок 2.2

```
>> Noise

m =

   -0.1394


SD =

    1.7915
```

3. Відкрийте програму, яка демонструвала б базові операції над сигналами - затримку, додавання і множення сигналів **(Sig_Man.m)**. У даній програмі генеруються три синусоїдальних сигнали - частотою 150, 450 і 1500 ГЦ (128 відліків при частоті дискретизації 8 КГц). Потім, формуються три нових сигнали. Перший сигнал - затриманий на 20 відліків сигнал з частотою 150 Гц. Другий сигнал - сума двох сигналів, 150 і 450 Гц. Третій сигнал - множення двох сигналів - 150 і 1500 Гц (амплітудна модуляція). Вихідні і отримані сигнали відображаються в окремих вікнах.

```
% Program demonstrating Basic Signal Manipulation
N = 128;
f1 = 150;
f2 = 450;
f3 = 1500;
fs = 8000;
n = 0 : N - 1;
x1 = sin(2 * pi * (f1 / fs) * n);
x2 = (1 / 3) * sin(2* pi * (f2 / fs) * n);
x3 = sin(2 * pi * (f3 / fs) * n);
figure(1);
subplot(1, 1, 1);
subplot(2, 3, 1);
plot(n, x1);
grid;
title('Signal, x1(n)');
subplot(2, 3, 2);
plot(n, x2);
grid;
title('Signal, x2(n)');
subplot(2, 3, 3);
plot(n, x3);
grid;
title('Signal, x3(n)');
% Signal Delay
x1d = [zeros(1, 20), x1(1 : N - 20)];
subplot(2, 3, 4);
plot(n, x1d);
grid;
title('Delayed x(n), [x1(n - 20)]');
% Signal Addition
xadd = x1 + x2;
subplot(2, 3, 5);
plot(n, xadd);
grid;
title('x1(n) + x2(n)');
% Signal Multiplication
xmult = x1 .* x3;
subplot(2, 3, 6);
plot(xmult);
grid;
title('x1 * x3');
```

Результатом роботи даної програми будуть сигнали, представлені на малюнку:

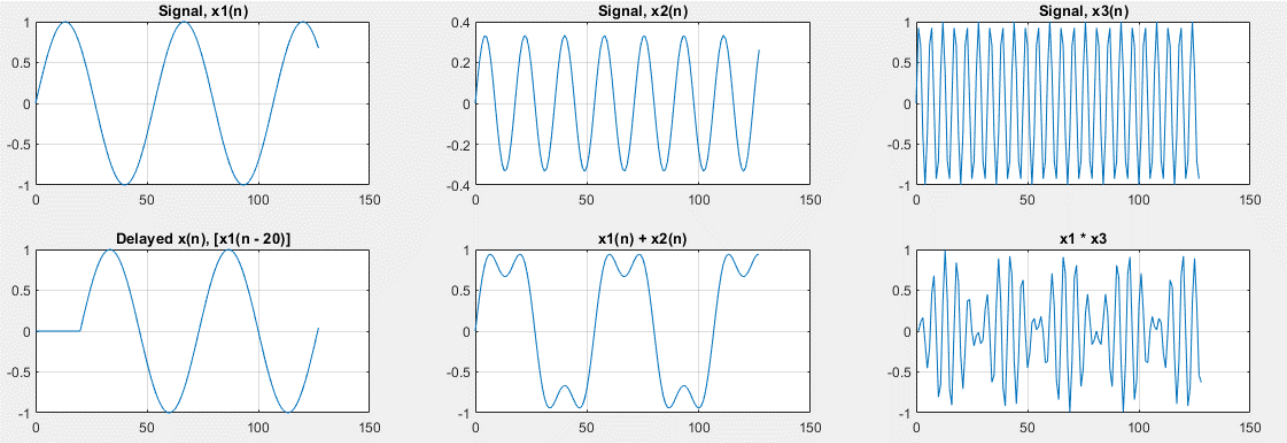


Рисунок 3

Завдання

Згенеруйте нові сигнали, що представляють собою:

1) Затриманий на 35 відліків сигнал з частотою 450 Гц;

```
% Signal Delay
x2d = [zeros(1, 35), x2(1 : N - 35)];
subplot(3, 3, 7);
plot(n, x2d);
grid;
title('Delayed x(n), [x2(n - 35)]');
```

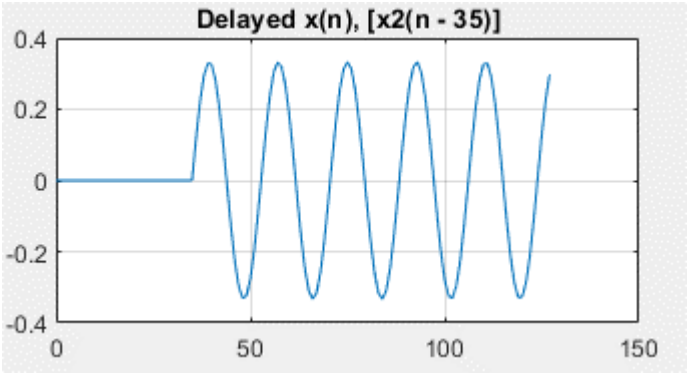


Рисунок 3.1

2) Суму сигналів частотою 450 і 1500 Гц;

```
% Signal Addition
xadd = x2 + x3;
subplot(3, 3, 8);
plot(n, xadd);
grid;
title('x2(n) + x3(n)');
```

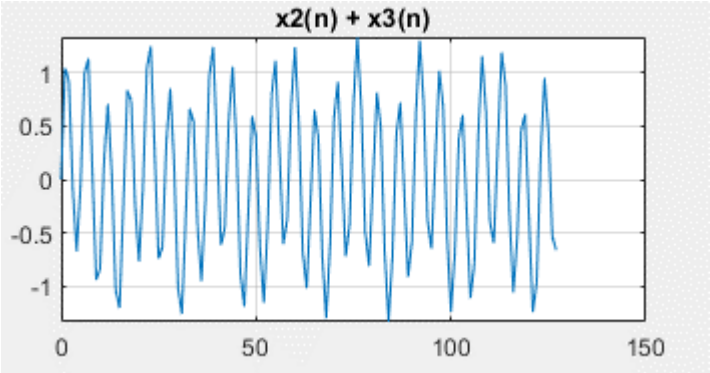


Рисунок 3.2

3) Добуток сигналів 150 і 450 Гц.

```
% Signal Multiplication
xmult = x1 .* x2;
subplot(3, 3, 9);
plot(xmult);
grid;
title('x1 * x2');
```

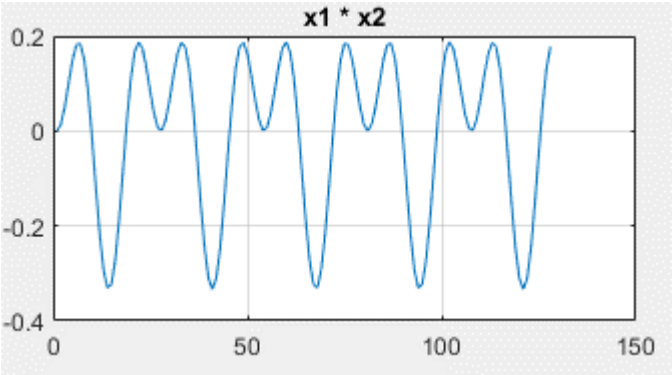


Рисунок 3.3

4. Відкрийте програму, яка демонструвала б додавання до вихідного гармонічного сигналу білого шуму (Sig_Noise.m). Відношення сигнал\шум задається рівним 0,1 дБ. Потужність вихідного сигналу вимірюється функцією awgn для розрахунку потужності шуму. Для отриманої послідовності визначається середнє значення і СКВ.

```
t = 0:.01:2;                        % Time vector for 2s
x = sin(2 * pi * t);                % Generation sin wave
snr = 0.1;                          % SNR = 0.1 dB
y = awgn(x, snr, 'measured');       % Add white noise
m = mean(y);
dev = std(y);
plot(t, x, t, y);                   %plot both signals
legend('Original signal', 'Signal with Noise');
```

Результатом роботи даної програми будуть сигнали, представлені на малюнку:

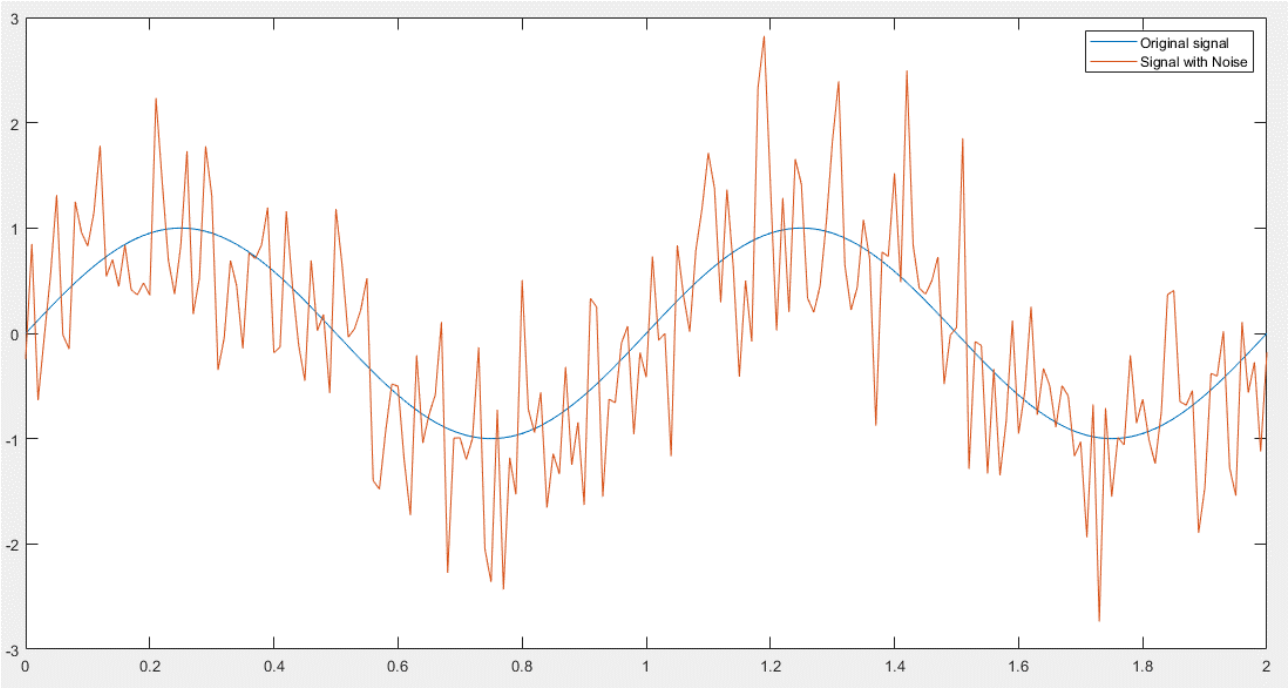


Рисунок 4

Завдання

1) Для даної програми - змініть співвідношення сигнал/шум і проаналізуйте результат.

snr = 20

Результат роботи даної програми:

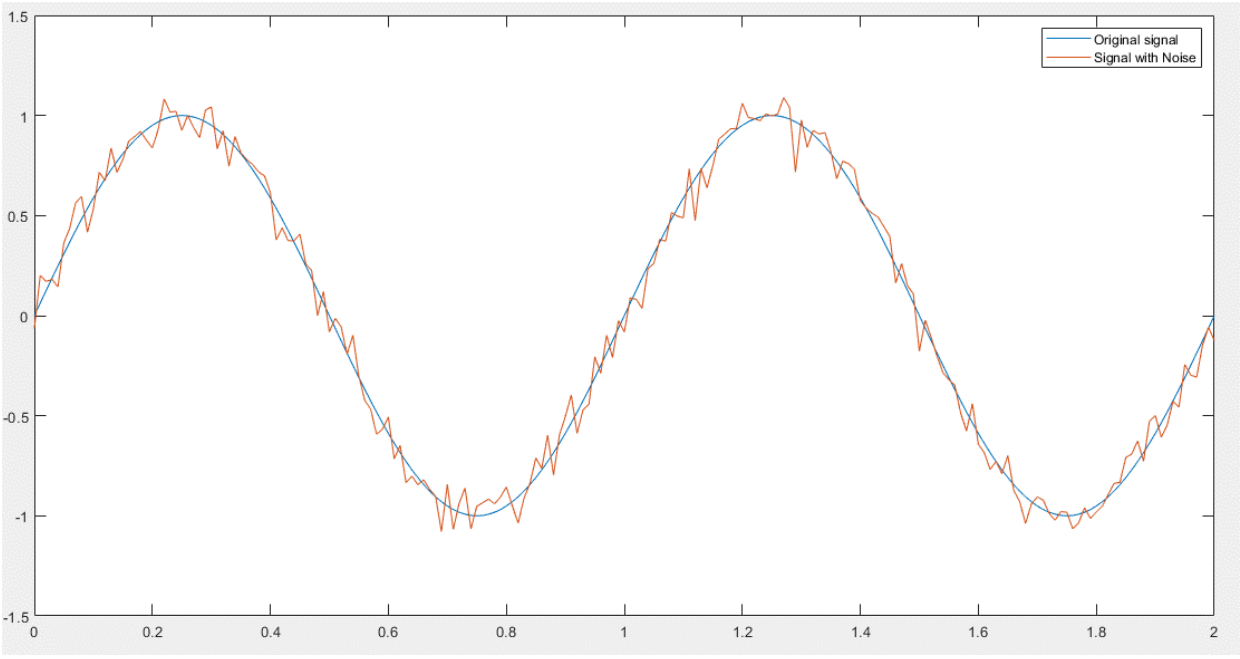


Рисунок 4.1

При співвідношенні сигнал/шум рівному 20 дб рівень зашумлення менший, тобто він став більш близьким до чистого сигналу.

2) Згенеруйте послідовність трикутних імпульсів (використовуючи функцію sawtooth). Задайте для неї амплітуду, частоту і зміщення. Додайте до неї білий шум з SNR = -3дБ. Розрахуйте середнє значення і СКО отриманого сигналу.

```
A = 6;
fs = 350;
OffSet = 5.05;
t = 0:.01:2;                                % Time vector for 2s
x = A * sawtooth(2 * pi * t) + OffSet;      % Generation sin wave
snr = -3;                                   % SNR = -3 dB
y = awgn(x, snr, 'measured');               % Add white noise
m = mean(y);
dev = std(y);
plot(t, x, t, y);                           % plot both signals
legend('Original signal', 'Signal with Noise');
```

Результат роботи даної програми:

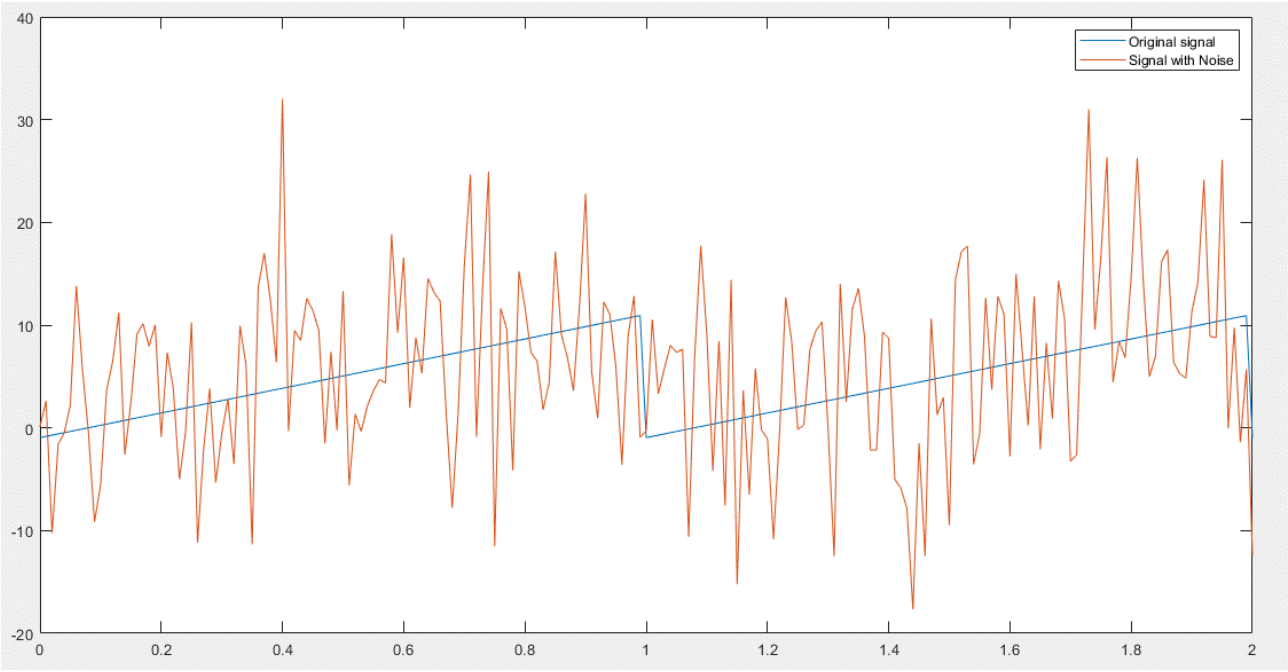


Рисунок 4.2

Завдання

1. **Опрацюйте основні команди, викладені вище, в системі MATLAB.**
2. **Створіть М-функцію, яка на вході отримує вектор довільної розмірності з даними і повертає:**
   a. **середнє значення, обчислене відповідно до формули (1.1), а також отримане в результаті застосування функції mean;**
   b. **середньоквадратичне відхилення, обчислене відповідно до формули (1.2), а також отримане в результаті застосування функції std.**

```
function y = task2 (x)

N = size(x, 2);
[m, n] = size(x);
   if (~((m == 1) | (n == 1)) | (m == 1 && n == 1))
    error('Вхідний масив має бути вектором')
   end
        y = sum(x) / length(x);
        m_Teor = sum(x) / N
        m_Real = mean(x)
        SD_Teor = sqrt(sum((x - m_Teor).^2) / (N - 1))
        SD_Real = std(x)
```

Результат роботи даної програми:

```
>> z = 1:78;
>> task2 (z)

m_Teor =

   39.5000


m_Real =

   39.5000


SD_Teor =

   22.6605


SD_Real =

   22.6605


ans =

   39.5000
```

3. **Створіть М-функцію, яка на вході отримує вектор довільної розмірності з даними і повертає значення статистичної похибки TE відповідно до формули (1.3).**

```
function y = task3 (x)

N = size(x, 2);
[m, n] = size(x);
   if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
    error('Вхідний масив має бути вектором')
   end
TE = (sqrt(1 / (N - 1) * (sum(x.^2) - ((sum(x).^2) / N))) / sqrt(N))
```

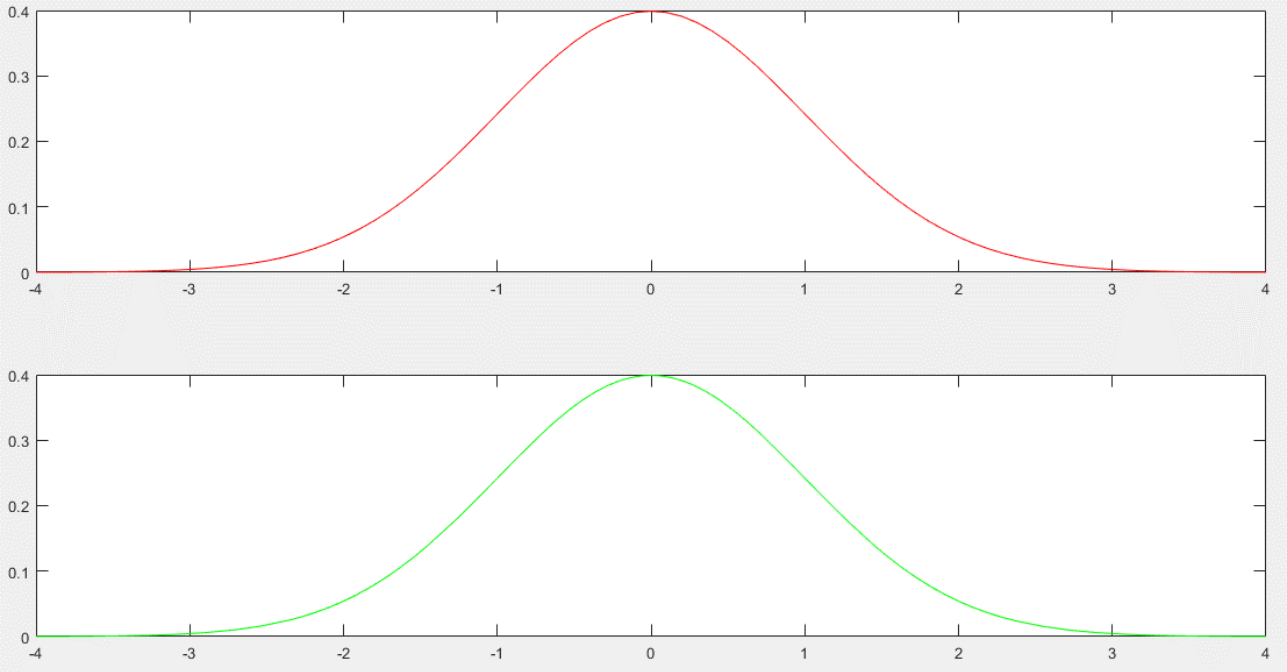
Результат роботи даної програми:

```
>> z = 1:78;
>> task3 (z)

TE =

    2.5658
```

4. **Самостійно досліджуйте функцію побудови гістограми hist (виклик довідки по даній функції - doc hist).**
5. **Побудуйте графік функції нормального розподілу відповідно до формули (1.4) за допомогою функцій plot і fplot.**

```
m = 0;
sd = 1;
x = -4:.1:4;
subplot(2, 1, 1)
P = 1 / (sqrt(2 * pi) * sd) * exp(-(x - m).^2 / 2 * sd.^2);
plot (x, P, 'r');
subplot(2,1,2)
P = @(x) 1 / (sqrt(2 * pi) * sd) * exp(-(x - m).^2 / 2 * sd.^2);
fplot (P, [-4, 4], 'g');
```

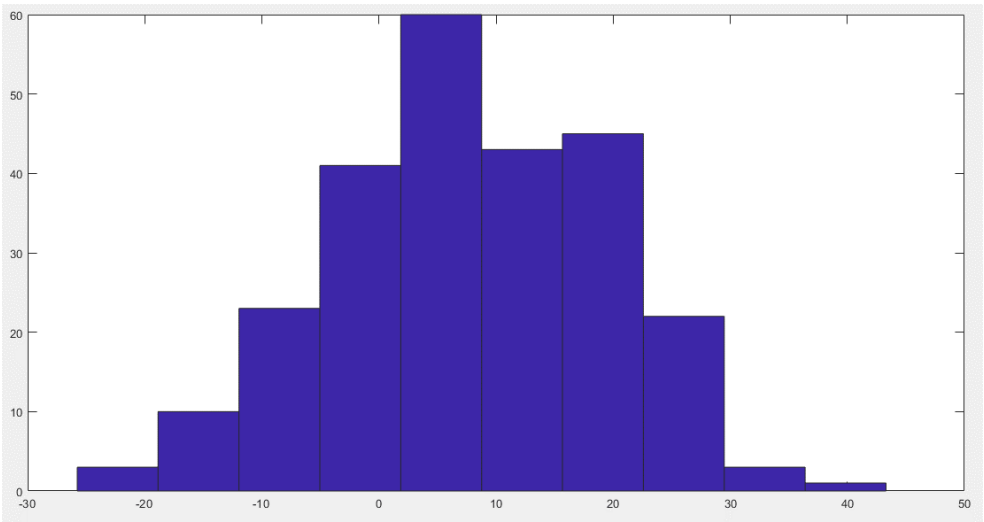
Результат роботи даної програми:



6. Створіть М-функцію на основі команди randn, яка генерує випадковий шум з нормальним законом розподілу з заданим середнім значенням і середньоквадратичним відхиленням.

```
mean = 7;
sd = 12;
x = -18:.1:7;
size1 = length(x);

y = mean + sd.*randn(1, length(x));
plot (x, y, 'y');
hist(y)
```

Результат роботи даної програми:



```
>> task6

mean =

     7


sd =

    12
```