Лабораторна робота №1

Тема: ***«Основи роботи в системі MATLAB»***

Ціль роботи: **знайомство з основними командами системи MATLAB**

Посібник з лабораторної роботи

Що таке MATLAB?

MATLAB – це високопродуктивна мова для технічних розрахунків. Вона складається з обчислення, візуалізації та програмування в зручному середовищі, де завдання та розв'язання виражаються у формі, близької до математичної. Типове використання **MATLAB** – це:

- Математичне обчислення;
- Створення алгоритмів;
- Моделювання;
- Аналіз даних, дослідження та візуалізація;
- Наукова та інженерна графіка;
- Розробка додатків, включаючи створення графічного інтерфейсу.

MATLAB – це інтерактивна система, в якій основним елементом даних є масив. Це дозволяє вирішувати різноманітні завдання, пов'язані з технічними обчисленнями, особливо в яких використовуються матриці та вектори, в кілька разів швидше, ніж під час написання програм з використанням «скалярних» мов програмування, таких як **Сі** або **Фортран**.

Слово **MATLAB** означає матричну лабораторію (**matrix laboratory**). **MATLAB** був спеціально написаний для забезпечення легкого доступу до **LINPACK** і **EISPACK**, які є сучасними програмними засобами для матричних обчислень.

MATLAB розвивався протягом декількох років, орієнтуючись на різноманітних користувачів. В університетському середовищі, він був стандартним інструментом для роботи в різних областях математики, машинобудування та науки. В промисловості **MATLAB** – це інструмент для високопродуктивних досліджень, розробок та аналізу даних.

В **MATLAB** важлива роль надається спеціалізованим групам програм, які називаються toolboxes. Вони дуже важливі для більшості користувачів **MATLAB**, так як дозволяють вивчати та використовувати спеціалізовані методи.

Toolboxes – це всебічна колекція функцій **MATLAB** (M-файлів), які дозволяють вирішувати конкретні класи завдань. Toolboxes використовуються для оброби сигналів, систем контролю, нейронних сіток, нечіткої логіки, вейлетів, моделювання і т.д.

СИСТЕМА MATLAB

Система MATLAB складається з п’яти основних частин:

- Мова **MATLAB.** Ця мова матриць та масивів високого рівня з керуванням потоками, функціями, структурами даних, введенням-виведенням і особливостями об’єктно-орієнтованого програмування. Це дозволяє як програмувати в «невеликому масштабі» для швидкого створення чорнових програм, так і у «великому» для створення великих та складних додатків.
- Середовище **MATLAB.** Це набір інструментів та пристосувань, з якими працює користувач або програміст **MATLAB.** Воно складається з засобів керування змінними в робочому просторі **MATLAB**, введенням та виведенням даних, а також створення, контроль та налагодження M-файлів і додатків **MATLAB.**
- Управління графікою. Це графічна система **MATLAB**, яка має команди високого рівня для візуалізації дво-і тривимірних даних, обробки зображень, анімації і ілюстрованої графіки. Вона також має команди низького рівня, які дозволяють повністю редагувати зовнішній вигляд графіки, так само, як у створенні Графічного Інтерфейсу Користувачів (GUI) для **MATLAB** додатків.
- Бібліотека математичних функцій. Це велика колекція обчислювальних алгоритмів від елементарних функцій, таких як сума, синус, косинус, комплексна арифметика, до більш складних, таких як звернення матриць, знаходження власних значень, функції Бесселя, швидке перетворення Фур'є.

Програмний інтерфейс. Це бібліотека, яка дозволяє писати програми на **Ci** та **Фортрані**, які взаємодіють з **MATLAB**. Вона містить засоби для виклику програм з **MATLAB** (динамічний зв'язок), викликаючи **MATLAB** як обчислювальний інструмент і для читання-запису MAT-файлів.

Про SIMULINK

Simulink, супутня **MATLAB** програма, - це інтерактивна система для моделювання нелінійних динамічних систем. Вона являє собою середовище,

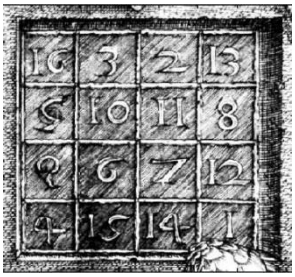
кероване мишею, яка дозволяє моделювати процес шляхом перетягування блоків діаграм на екрані і їх маніпуляцією. **Simulink** працює з лінійними, нелінійними, безперервними, дискретними, багатовимірними системами.

Blocksets - це доповнення до **Simulink**, які забезпечують бібліотеки блоків для спеціалізованих додатків, таких як зв'язок, обробка сигналів, енергетичні системи.

Real-Time Workshop - це програма, яка дозволяє генерувати **C** код з блоків діаграм і запускати їх на виконання на різних системах реального часу.

МАТРИЦІ І МАГІЧНІ КВАДРАТИ

Кращий спосіб почати роботу з **MATLAB** - це навчитися поводитись з матрицями. В **MATLAB** матриця - це прямокутний масив чисел. Особливе значення надається матрицям 1x1, які є скалярами, і матриць, які мають один стовпчик або один рядок, - векторам. **MATLAB** використовує різні способи для зберігання числових та не числових даних, проте спочатку найкраще розглядати всі дані як матриці. **MATLAB** організований так, щоб всі операції в ньому були якомога більш природніми. У той час як інші програмні мови працюють з числами як елементами мови, **MATLAB** дозволяє вам швидко і легко оперувати з цілими матрицями.



Хороший приклад матриці можна знайти на гравюрі часів Ренесансу художника і любителя математики ***Альбрехта Дюрера***. Це зображення містить багато математичних символів, і якщо добре придивитися, то в верхньому правому куті можна помітити квадратну матрицю. Це матриця відома як магічний квадрат і за часів Дюрера вважалося, що вона володіє магічними властивостями. Вона і насправді володіє чудовими властивостями, які варті дослідження.

ВВЕДЕННЯ МАТРИЦЬ

Ви можете вводити матриці в **MATLAB** кількома способами:
- вводити повний список елементів;
- завантажувати матриці із зовнішніх файлів;
- генерувати матриці, використовуючи вбудовані функції;
- створювати матриці за допомогою ваших власних функцій в M-файлах.

Почнемо з введення матриці Дюрера як списку елементів. Ви повинні виконати декілька основних умов:

• відокремлювати елементи рядка пробілами або комами
• використовувати крапку з комою,; , для позначення закінчення кожного рядка
• весь список елементів поставити в квадратні дужки, [].

Щоб ввести матрицю ***Дюрера*** просто напишіть (рис. 1.1):

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

MATLAB відобразить матрицю, яку ми ввели,

```
A =

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

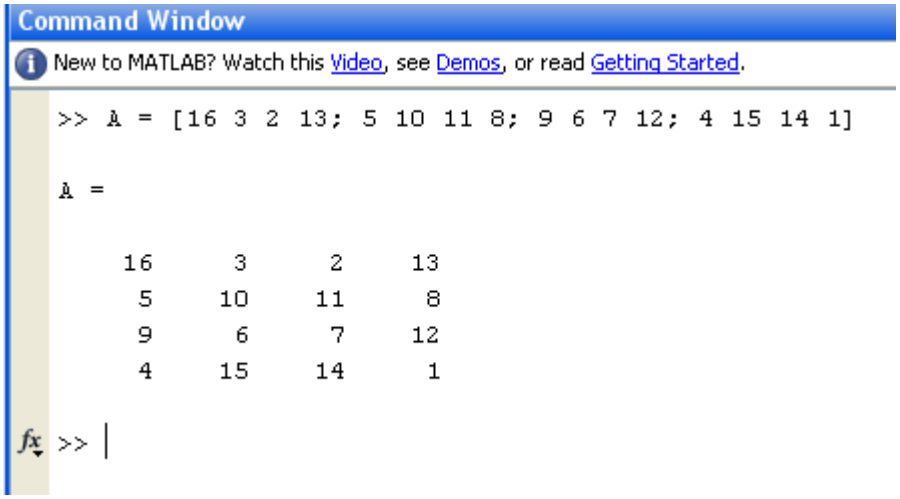


Рис. 1.1 Приклад фрагменту командного вікна MATLAB

Це точно відповідає числам на гравюрі. Якщо ми ввели матрицю, то вона автоматично запам'ятовується середовищем **MATLAB**. І ми можемо до неї легко звернутися як до **A**. Зараз ми маємо **A** в робочому просторі **MATLAB** (рис. 1.2)

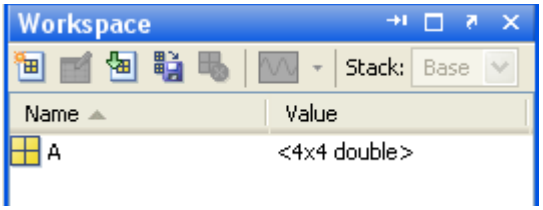


Рис. 1.2 Приклад фрагменту робочого простору MATLAB

ОПЕРАЦІЇ ДОДАВАННЯ ЕЛЕМЕНТІВ, ТРАНСПОРТУВАННЯ ТА ДІАГОНАЛІЗАЦІЇ МАТРИЦІ

Ви можливо вже знаєте, що особливі властивості магічного квадрату пов'язані з різними способами додавання його елементів. Якщо ви берете суму елементів вздовж якого-небудь рядка або стовпця, або уздовж будь-якої з двох головних діагоналей, ви завжди отримаєте одне і теж число. Давайте перевіримо це, використовуючи **MATLAB**. Перше твердження, яке ми перевіримо -

```
sum(A)
```

MATLAB дасть відповідь

```
ans =

    34    34    34    34
```

Коли вихідна змінна не визначена, **MATLAB** використовує змінну **ans**, коротко від answer - відповідь, для зберігання результатів обчислення. Ми рахували вектор-рядок, що містить суму елементів стовпців матриці **A**. Дійсно, кожен стовпець має однакову суму, магічну суму, рівну 34.

А як щодо сум в рядках? **MATLAB** надає перевагу працювати зі стовпцями матриці, таким чином, кращий спосіб отримати суму в рядках - це транспонувати нашу матрицю, підрахувати суму в стовпці, а потім транспонувати результат. Операція транспонування позначається апострофом або одинарними лапками. Вона дзеркально відображає матрицю щодо головної діагоналі і змінює рядки на стовпці. Таким чином

```
A'
```

викликає

```
ans =

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

А вираз

```
sum(A')'
```

викликає результат вектор-стовпчик, який містить суми в рядках

```
ans =
```

```
    34
    34
    34
    34
```

Суму елементів на головній діагоналі можна легко отримати за допомогою функції *diag*, котра вибирає цю діагональ

```
diag(A)
ans =
    16
    10
     7
     1
```

А функція

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

викликає

```
ans =
    34
```

Таким чином, ми перевірили, що матриця на гравюрі ***Дюрера*** дійсно магічна, і навчились використовувати деяка матричні операції **MATLAB**. В наступних розділах ми продовжимо використовувати цю матрицю для демонстрації додаткових можливостей и **MATLAB**.

ІНДЕКСИ

Елемент в рядку *i* и стовпці *j* матриці **A** зображується **A**(*i*,*j*). Наприклад, **A**(4,2) - це число в четвертому ряду і другому стовпці. Для нашого магічного квадрату **A**(4,2) = 15. Таким чином, можна обчислити суму елементів в четвертому стовпці матриці **A**, набрав

```
A(1,4) + A(2,4) + A(3,4) + A(4,4)
ans =
   34
```

Проте, це не найкращий спосіб додавання окремого рядку.

Також можливо звертатися до елементів матриці через один індекс, **A**(k). Це звичайний спосіб посилатися на рядки і стовпці матриці. Але його можна використовувати тільки з двовимірними матрицями. В цьому випадку масив розглядається як довгий вектор, сформований із стовпців початкової матриці.

Так, для нашого магічного квадрату, **A**(8) - це інший спосіб посилатися на значення 15, що зберігається в **A**(4,2).

Якщо ви намагаєтеся використовувати значення елемента поза матрицею, **MATLAB** видасть помилку:

```
t=A(4,5)
??? Index exceeds matrix dimensions.
```

З іншого боку, якщо ви зберігаєте значення поза матрицею, то розмір матриці збільшується.

```
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

Двокрапка,:, - це один з найбільш важливих операторів **MATLAB**. Він проявляється в різних формах. Вираз

```
1:10
ans =
     1     2     3     4     5     6     7     8     9    10
```

Для отримання зворотного інтервалу, опишемо приріст. Наприклад

```
100:-7:50
ans =

   100    93    86    79    72    65    58    51
```

Або

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

що призведе

```
ans =

         0    0.7854    1.5708    2.3562    3.1416
```

Індексний вираз, включаючи двокрапку, відноситься до частини матриці. **A**(1: k, j) - це перші до елементів j-го стовпця матриці **A**.

Так `sum (A(4, 1: 4))` обчислює суму четвертого рядка. Але є і кращий спосіб. Двокрапка, сама по собі, звертається до всіх елементів в рядку і стовпці матриці, а слово **end** - до останнього рядка або стовпця. Так

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

обчислює суму елементів в останньому стовпці матриці **A**

```
ans =

    34
```

Чому магічна сума квадрату 4x4 рівна 34? Якщо цілі числа від 1 до 16 відсортовані в чотири групи з рівними сумами, ця сума повинна бути

```
sum(1:16)/4
```

котра, звичайно, рівна

```
ans =

    34
```

ФУНКЦІЯ MAGIC

MATLAB насправді володіє вбудованою функцією, яка створює магічний квадрат майже будь-якого розміру. Тож не дивно, що ця функція називається **magic**.

```
B=magic(4)

B =

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

Ця матриця майже та сама матриця, що і на гравюрі ***Дюрера***, і вона має такі самі магічні властивості. Єдина відмінність полягає в тому, що два середніх стовпчика помінялися місцями. Для того щоб перетворити **B** у матрицю ***Дюрера*** **A**, переставимо їх місцями.

```
A=B(:,[1 3 2 4])
```

Це означає, що для кожного рядка матриці **B** елементи переписуються в порядку 1, 3, 2, 4

```
A =

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

Чому ***Дюрер*** перевпорядкував стовпці на відміну від того, що використовує **MATLAB**? Без сумніву, він хотів включити дату гравюри, 1514, в нижню частину магічного квадрату.

ВИРАЗИ

Як і більшість інших мов програмування, **MATLAB** надає можливість використання математичних виразів, але на відміну від багатьох з них, ці вирази в MATLAB включають матриці. Основні складові виразу:

- змінні
- числа
- оператори
- функції

ЗМІННІ

В **MATLAB** немає необхідності у визначенні типу змінних або розмірності. Коли **MATLAB** зустрічає нове ім'я змінної, він автоматично створює змінну і виділяє відповідний обсяг пам'яті. Якщо змінна вже існує, **MATLAB** змінює її склад і якщо це необхідно виділяє додаткову пам'ять. Наприклад,

```
num_students = 25
```

створює матрицю 1x1 з іменем **num_students** та зберігає значення 25 в її єдиному елементі.

Імена змінних складаються з літер, цифр або символів підкреслення. **MATLAB** використовує тільки перші 31 символ імені змінної. **MATLAB** чутливий до регістрів, він розрізняє великі і малі літери. Тому **A** і **a** - не одна і та ж змінна. Щоб побачити матрицю пов'язану зі змінною, просто введіть назву змінної.

ЧИСЛА

MATLAB використовує прийняту десяткову систему числення, з необов'язковою десятковою крапкою і знаками плюс-мінус для чисел. Наукова система числення використовує букву **e** для визначення множника ступеня десяти. Уявні числа використовують **i** або **j** як суфікс.

Всі числа для зберігання використовують формат long, визначений стандартом плаваючою точки IEE. Числа з плаваючою точкою мають обмежену точністю - приблизно 16 значущих цифр і обмеженим діапазоном -приблизно від 10^{-308} до 10^{308} (Комп'ютер VAX використовує інший формат чисел з плаваючою точкою, але їх точність і діапазон приблизно однакові).

ОПЕРАТОРИ

Вирази використовують звичайні арифметичні операції та правила старшинства.

\+ додавання

\- віднімання

* множення

/ ділення

\ ліве ділення (описано в розділі Матриці та Лінійна Алгебра у книзі "*Using MATLAB*")

^ степінь

' комплексно поєднане транспонування

() визначення порядку обчислення

ФУНКЦІЇ

MATLAB надає велику кількість елементарних математичних функцій, таких як *abs, sqrt, exp, sin*. Обчислення квадратного кореня або логарифму від'ємного числа не є помилкою: в цьому випадку результатом є відповідне комплексне

число. **MATLAB** також надає і більш складні функції, включаючи Гамма-функцію і функції Бесселя. Більшість з цих функцій мають комплексні аргументи. Щоб вивести список всіх елементарних математичних функцій, наберіть

```
help elfun
```

Для вводу складніших математичних та матричних функцій, наберіть

```
help specfun
```

```
help elmat
```

відповідно.

Деякі функції, такі як *sqrt* і *sin*, - вбудовані. Вони є частиною **MATLAB**, тому вони дуже ефективні, але їх обчислювальні деталі важко доступні. У той час як інші функції, такі як *gamma* і *sink*, реалізовані в M-файлах. Тому ви можете легко побачити їх код і, в разі необхідності, навіть модифікувати його.

Декілька спеціальних функцій надають значення часто використовуваних констант.

pi	3.14159265...
i	уявна одиниця
j	теж саме, що й i
eps	відносна точність числа з плаваючою крапкою
realmin	найменше число з плаваючою крапкою
realmax	найбільше число з плаваючою крапкою
Inf	нескінченність
NaN	не число

Нескінченність з'являється при діленні на нуль або при виконанні математичного виразу, який призводить до переповнення, тобто до перевищення *realmax*. Не число (*NaN*) генерується при обчисленні виразів типу 0/0 або *Inf- Inf*, які не мають певного математичного значення.

Імена функцій не є зарезервованими, тому можливо змінювати їх значення на нові, наприклад

```
eps = 1.e-6
```

і далі використовувати це значення в наступних обчисленнях. Початкове значення може бути встановлено наступним чином

```
clear eps
```

ВИРАЗИ

Ви вже познайомились з деякими прикладами використання виразів в MATLAB. Нижче наведено ще декілька прикладів з результатами.

```
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

MATLAB має чотири функції, які створюють матриці:

zeros	всі нулі
ones	всі одиниці
rand	рівномірний розподіл випадкових елементів
randn	нормальний розподіл випадкових елементів

Деякі приклади:

```
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 =

     9     2     6     4     8     7     4     0     8     4

R = randn(4,4)

R =

   -0.4326   -1.1465    0.3273   -0.5883
   -1.6656    1.1909    0.1746    2.1832
    0.1253    1.1892   -0.1867   -0.1364
    0.2877   -0.0376    0.7258    0.1139
```

ЗАВАНТАЖЕННЯ МАТРИЦЬ

Команда *load* зчитує двійкові файли, що містять матриці, створені в **MATLAB** раніше, або текстові файли, що містять чисельні дані. Текстові файли повинні бути сформовані у вигляді прямокутної таблиці чисел, відокремлених пробілами, з рівною кількістю елементів в кожному рядку. Наприклад, створимо поза **MATLAB** текстовий файл, який містить 4 рядки:

```
16.0    3.0    2.0    13.0
5.0    10.0    11.0    8.0
9.0    6.0    7.0    12.0
4.0    15.0    14.0    1.0
```

Збережемо цей файл під назвою magik.dat. Тоді команда `load magik.dat` прочитає цей файл і створить змінну magik, яка міститиме нашу матрицю.

ОБ'ЄДНАННЯ

Об'єднання - це процес з'єднання маленьких матриць для створення більших. Фактично, ви створили вашу першу матрицю об'єднанням її окремих елементів. Пара квадратних дужок - це оператор об'єднання. Наприклад, почнемо з матриці **A** (магічного квадрата 4x4) і сформуємо

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

Результатом буде матриця 8x8, отримана сполученням чотирьох підматриць

```
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
```

Це матриця лише наполовину є магічною. ЇЇ елементи є комбінацією цілих чисел від 1 до 64, а суми в стовпцях точно дорівнюють значенню для магічного квадрату 8x8.

```
sum (B)

ans =

   260   260   260   260   260   260   260   260
```

Однак, суми в рядках цієї матриці (*sum(B')'*) не всі одинакові. Необхідно провести додаткові операції, щоб зробити цю матрицю дійсно магічним квадратом 8x8.

ВИДАЛЕННЯ РЯДКІВ ТА СТОПЧИКІВ

Ви можете видаляти ряди та стовпчики матриці, використовуючи просто пару квадратних дужок. Розглянемо

```
X  =  A;
```

Тепер видалимо другий стовпчик матриці X.

```
X(:,2) = []
```

Ця операція змінить X наступним чином

```
X =

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

Якщо ви видаляєте один елемент матриці, то результат вже не буде матрицею. Так вираз

```
X(1,2) = []
```

результатом обчислення видасть помилку. Однак використання одного індексу видаляє окремий елемент або послідовність елементів і формує елементи, які залишились у вектор-рядок. Так

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

видасть результат

```
X =

    16     9     2     7    13    12     1
```

ПЕРЕМНОЖЕННЯ МАТРИЦЬ

При перемноженні двох матриць використовується оператор '*'. Наприклад, якщо

```
A =

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

    16     4     7     3
     5    -7     2     9
     0     8    23    65
    -7     4    17     9
```

Тоді **C = A*B** дасть результат

```
C =

       180         111         385         322
        74          70         444         892
        90          98         440         644
       132          27         397        1066
```

Також в системі MATLAB передбачена можливість поелеметного перемноження. Для цієї цілі використовується точка перед знаком множення. Наприклад:

```
C = A.*B
```

в результаті

```
C =
   256    12    14    39
    25   -70    22    72
     0    48   161   780
   -28    60   238     9
```

СТВОРЕННЯ М-ФАЙЛІВ

М-файли є звичайними текстовими файлами, які створюються за допомогою текстового редактора. Для операційного середовища персонального комп'ютера система MATLAB підтримує спеціальний вбудований редактор / відлагоджувач, хоча можна використовувати і будь-який інший текстовий редактор з ASCII-кодами.

Відкрити редактор можна двома способами:
• з меню File вибрати опцію New, а потім M-File.
• використовувати команду редагування **edit**.

М-функції є М-файлами, які допускають наявність вхідних і вихідних аргументів. Вони працюють зі змінними в межах своєї робочої області, відмінною від робочої області системи MATLAB.

Приклад

Функція `average` – це доволі простий М-файл, який обчислює середнє значення елементів вектору:

```
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
```

Спробуйте ввести ці команди в М-файл, який називається **average.m**. Функція `average` допускає єдиний вхідний і єдиний вихідний аргументи. Для того, щоб викликати функцію `average`, треба ввести наступні оператори:

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

Отримуємо результат

```
ans = 50
```

СТАТИСТИЧНІ ХАРАКТЕРИСТИКИ СИГНАЛІВ

Середнє значення сигналу (його постійна складова) визначається наступною формулою:

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

Середньоквадратичне відхилення (СКВ, девіація, змінна складова) сигналу визначається за такою формулою:

$$\sigma^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}(x_i - \mu)^2 \quad (1.2)$$

Значення статистичної помилки сигналу визначається за такою формулою:

$$TE = \frac{\sigma}{\sqrt{N}} \quad (1.3)$$

Функція нормального розподілу описується наступною формулою:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} \quad (1.4)$$

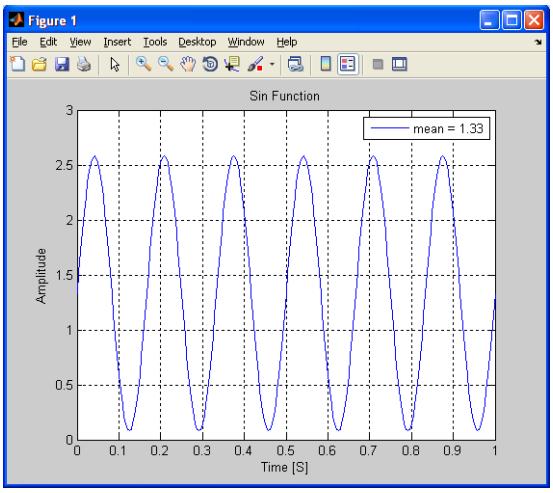
ПОРЯДОК ВИКОНАННЯ РОБОТИ

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** (постійна складова) 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:



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

ЗАВДАННЯ

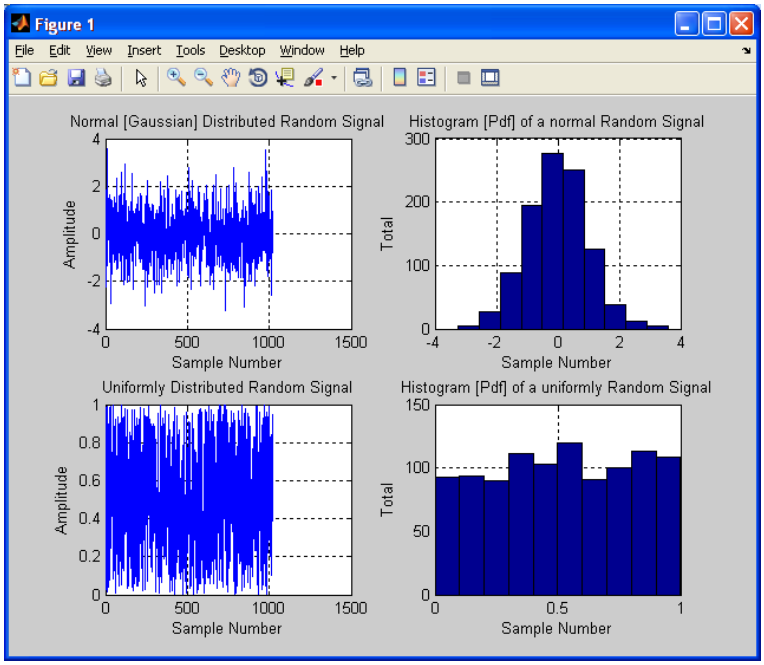
1). Змініть значення амплітуди, частоти сигналу і постійного зміщення. Перевірте отримані нові результати середнього значення і СКВ.
2). Згенеруйте послідовність прямокутних імпульсів (використовуючи функцію **square**). Задайте для неї амплітуду, частоту і зміщення. Розрахуйте середнє значення і СКВ.
3) Згенеруйте послідовність трикутних імпульсів (використовуючи функцію **sawtooth**). Задайте для неї амплітуду, частоту і зміщення. Розрахуйте середнє значення і СКВ.

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

```
% 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:



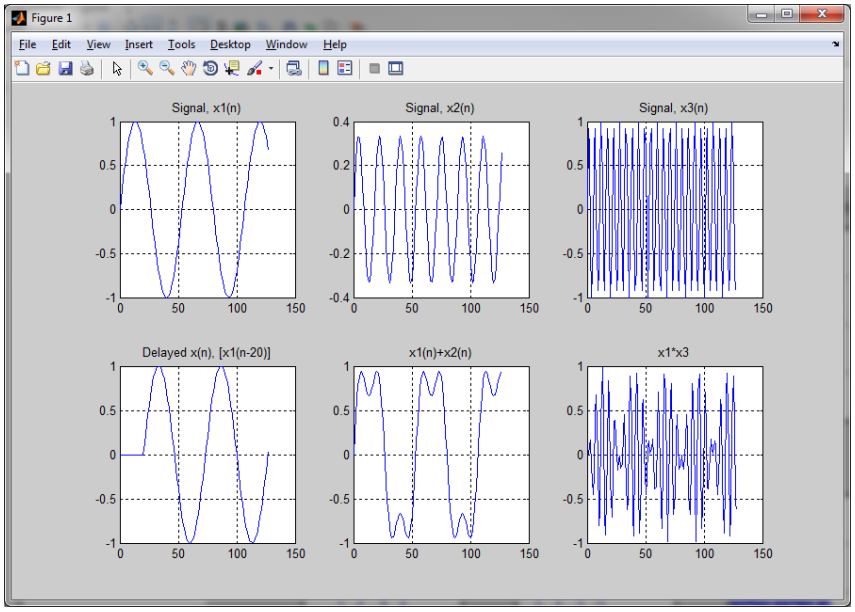
ЗАВДАННЯ

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

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:



ЗАВДАННЯ

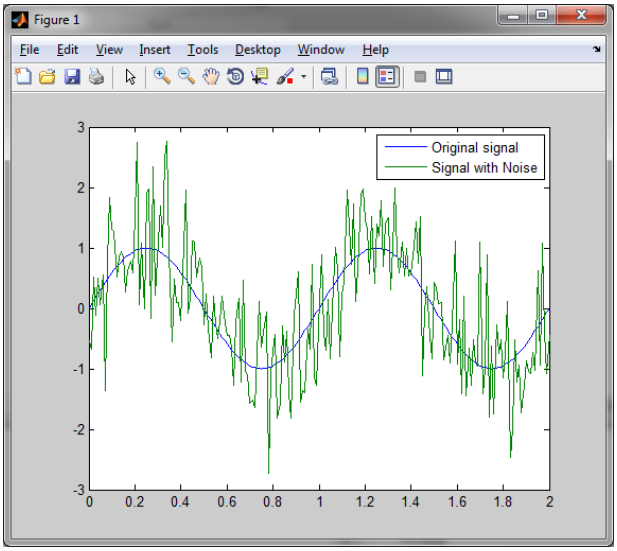
Згенеруйте нові сигнали, що представляють собою:
1) Затриманий на 35 відліків сигнал з частотою 450 Гц;
2) Суму сигналів частотою 450 і 1500 Гц;
3) Добуток сигналів 150 і 450 Гц.

У пакеті MatLab є ще одна корисна функція - **awgn**. Вона дозволяє додавати до сигналу білий шум (з нормальним розподілом) із заданим співвідношенням сигнал / шум (SNR). Формат виклику даної функції наступний:
y = **awgn (x, snr, sigpower, state, 'powertype')**, де
x - вектор відліків вихідного 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 параметри не є обов'язковими).

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:



ЗАВДАННЯ

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

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.**
3. **Створіть М-функцію, яка на вході отримує вектор довільної розмірності з даними і повертає значення статистичної похибки TE відповідно до формули (1.3).**
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.**
6. **Створіть М-функцію на основі команди randn, яка генерує випадковий шум з нормальним законом розподілу з заданим середнім значенням і середньоквадратичним відхиленням.**