Лабораторна робота № 4.

Тема. Елементи лінійної алгебри та векторного аналізу.

Мета роботи: навчитись проводити операції з векторами та матрицями в системі **MATLAB**.

Теоретичний мінімум

Вектором називається величина, яка характеризується як числовим значенням, так і напрямгом у просторі: $\bar{a} = (a_1, a_2, ..., a_n)$.

Числове значення вектора називається довжиною або модулем вектора [16]: $|\bar{a}| = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}$

Матрицею розміру $m \times n$ називається упорядкована множина з $m \times n$ елементів $a_{ij}, i = 1, 2, ... m, j = 1, 2, ... n$, які розміщени у вигляді прямокутної таблиці з m рядків і n стовпців:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ a_{31} & a_{32} & a_{33} & ... & a_{3n} \\ ..... & & & & \\ a_{m1} & a_{m2} & a_{m3} & ... & a_{mn} \end{pmatrix}.$$

Одним з найзручніших способів зберігання даних, що використовуються у всіх мовах програмування і обчислювальних пакетах, є масиви. До особливостей роботи з масивами в системі **MATLAB** відноситься те, що одновимірний масив може бути вектор-рядком або вектор-стовпцем (одновимірний масив в **MATLAB** задається у вигляді матриці розміром $1 \times n$).

Вправа 1. Форми завдання вектор-стопців і вектор-рядків та здійснення елементраних операцій над ними.

Як вже раніше визначалось, для завдання вектора в системі **MATLAB** використовуються квадратні дужки, елементи вектора відділяються один від одного за допомогою:

- крапки з комою у разі необхідності отримати вектор-стовпець;
- пропуску або коми, якщо необхідно розмістити елементи у вигляді вектор-рядка.

Приклад 1. Представити вихідні вектор-стовпці і вектор-рядки у вигляді масивів, де: $\bar{a} = \begin{pmatrix} 1 \\ 2 \\ 5 \\ 3 \\ 4 \\ 7 \end{pmatrix}$, $\bar{b} = \begin{pmatrix} 6 \\ 5 \\ 4 \\ 8 \\ 9 \\ 2 \end{pmatrix}$, $\bar{u} = (1, 3, 6, 7, 9, 9)$, $\bar{v} = (5, 6, 7, 3, 6, 0)$.

Використовуємо наступну форму завдання інформації про вихідні вектори:

```
>> a=[1; 2; 5; 3; 4; 7]
>> b=[6; 5; 4; 8; 9; 2]
>> u=[1 3 6 7 9 9]
>> v=[5 6 7 3 6 0]
```

Крапка з комою в кінці останнього командного рядка використовується щоб позбутися отримання на екрані інформації, що підготовлена в відповідному командному рядку (зауважимо, що вона ніяк не пов'язана з крапкою з комою, яка є роздільником компонентів у вектор–стовпцях).

Приклад 2. Вивести в командне вікно значення векторів **a**, **b**, **u**, **v** і переконатись, як в системі **MATLAB** відображається зміст вектор–рядків і вектор–столбцов. Інформацію про вектори можна отримати за допомогою команди **whos**, яка допомогає переконатись, що компоненти вектора – числа – зберігаються в двовимірних масивах, розмір яких має вигляд 1×1.

Вектори також представляються двовимірними масивами, розміри яких записуються у формі $1\times n$ або $m\times1$.

Довжина (модуль) вектора може бути встановлена за допомогою команди **length(a)**, де **a** – вектор, для якого відшукується відповідна інформація.

Приклад 3. Знайти довжину вектора a=[1; 2; 5; 3; 4; 7].

Використовуємо команду **length (**з метою виключення в вихідній інформації змінної **ans** застосовуємо команду **disp**):

```
>> disp(length(a))
    6
```

До вектор-стовпців (вектор-рядків) з однаковим числом елементів можна застосовувати операції додавання та вирахування («+» і «–»).

Приклад 4. Виконати операції покомпонентного додавання вектор-стовпців $\bar{a} = \begin{pmatrix} 1 \\ 2 \\ 5 \\ 3 \\ 4 \\ 7 \end{pmatrix}$ і $\bar{b} = \begin{pmatrix} 6 \\ 5 \\ 4 \\ 8 \\ 9 \\ 2 \end{pmatrix}$ та покомпонентного вирахування вектор-рядків $\bar{u} = (1, 3, 6, 7, 9, 9)$ і $\bar{v} = (5, 6, 7, 3, 6, 0)$. Присвоїти результати новим векторам відповідно $\bar{c}$ та $\bar{w}$.

Для виконання завдання достатньо записати у командному рядку системи **MATLAB** наступне:

```
>> c=a+b
>> w=u–v
```

Операції додавання та вирахування вектор-рядка і вектор-стовпця або векторів різних розмірів приводить до помилки.

Оператор «*****» призначений для отримання добутку вихідніх векторів (наприклад, **a** і **b**) за правилом множення матриць. Оскільки система **MATLAB** відокремлює вектор-рядки від вектор-стовпців, то допускаються операції множення вектор-рядка **a** на такій же по довжині вектор-стовпець **b** (операція відповідає отриманню скалярного добутку для двох вектор-рядків **a** і $\mathbf{b}^{\mathbf{T}}$, де $\mathbf{b}^{\mathbf{T}}$ – вектор-рядок, що отримується з вектор-стовпця **b** за допомогою операції транспонування) або множення вектор-стовпця на вектор-рядок (в результаті якого виходить прямокутна матриця) [20].

Скалярний добуток двох векторів може бути знайдений за допомогою команди **dot**, а векторний добуток – команди **cross**.

Приклад 5. Знайти скалярний та векторний добутки вектор-стовпця $\bar{a} = \begin{pmatrix} 1 \\ 2 \\ 5 \end{pmatrix}$ та вектор-рядка $\bar{u} = (1, 3, 6)$.

Результат добутків можна отримати, якщо виконати наступні команди:

```
>> a=[1; 2; 5];
>> u=[1 3 6 ];
>> k=cross(a,u)
k =
   –3   –1    1
>> s=dot(a,u)
```

```
s =
    37
```

Зауваження. Векторний добуток визначен тільки для векторів, які складаються з трьох компонентів.

Для операції транспонування зарезервований апостроф «'». Якщо вектор містить комплексні числа, то операція «'» приводить до комплексно-сполученого вектора. При обчисленні скалярного і векторного добутків за допомогою команд **cross** і **dot** не обов'язково додержуватись того, щоб обидва вектори були або стовпцями, або рядками.

MATLAB підтримує поелементні операції з векторами [15]. Разом з операцією множення за правилом знаходження добутку матриць, існує операція покомпонентного множення «**.***» (точка із зірочкою). Дана операція застосовується до векторів однакової довжини і приводить до вектора тієї ж довжини, що і вихідні вектори, компоненти якого будуть відповідати добутку відповідних компонети вихідних векторів.

Приклад 6. Виконати покомпонентне множення векторів $\bar{a} = \begin{pmatrix} 1 \\ 2 \\ 5 \\ 3 \\ 4 \\ 7 \end{pmatrix}$ і $\bar{b} = \begin{pmatrix} 6 \\ 5 \\ 4 \\ 8 \\ 9 \\ 2 \end{pmatrix}$.

Ця операція може бути виконана, зокрема, наступним чином: в відповідних командних рядках задаються вектор-стовпець **a** та вектор-стовпець **b** і застосовується операція покомпонентного множення. Результат має форму вектор-стовпця:

```
>> a=[1; 2; 5; 3; 4; 7];
b=[6; 5; 4; 8; 9; 2];
>> f=a.*b
f =
    6
   10
   20
   24
   36
   14
```

Рекомендуємо виконати операцію покомпонентного множення векторів $\bar{a}$ і $\bar{b}$, що задані у формі вектор-рядків.

Аналогічним чином працює покомпонентний розподіл, що визначаэться символом «**./**» (крапка з косою рисою – прямий слеш). Крім того, операція «**.**» (крапка із зворотною косою рисою) здійснює зворотне покомпонентний розподіл, тобто вирази **a./b** і **b.\a** еквівалентні. Зведення компонентыв вектора **a** в скінчену степень виконується з використанням символу «**.^**».

Для виконання операції транспонування вектор-рядків або вектор-стовпців призначено поєднання двох символів точки з апострофом – «**.'**». Операції «**'**» і «**.'**» для дійсних векторів приводять до однакових результатів.

Зауваження. Покомпонентні операції не обов'язково застосовувати при здійсненні операції множення вектора на число або числа на вектор, операції розподілу вектора на число, операцій додавання і вирахування вектора і числа. Наприклад, при виконанні операції **a*2** результат є вектор такого виміру, що і вихідний вектор **a** з подвоєнимикомпонентами.

Вектори можуть бути аргументами вбудованих математичних функцій, таких, як **sin**, **cos** і т.д. В результаті отримуємо вектор зкомпонентами, які дорівнюють значенню вихідної функції від відповідних компонентів вихідного вектора.

Приклад 6. Знайти значення функції $\sin x$, якщо $\vec{x} = (0 \;\; \pi/2 \;\; \pi)$.

```
>> q = sin([0 pi/2 pi])
q =
0 1.0000 0.0000
```

Часто потрібно обчислити функцію від вектора значень аргументів, які відрізняються один від одного на постійну величину. Для створення таких вектор-рядків передбачений оператор «:».

Приклад. 7. Обчислити значення функції $f(x) = \dfrac{x\cos x + x^3}{x+2}$ в інтервалі $x \in [-1 \;\; 1]$ та величиною (кроком) зміни аргументу $h = 0.5$.

Результат можна отримати, якщо виконати наступні дії:

```
>> x= –1:.5:1
x =
  –1.0000   –0.5000        0    0.5000    1.0000
>> f=(x.*cos(x)+x.^3)./(x+2)
f =
  –1.5403   –0.3759        0    0.2255    0.5134
```

Зауваження. Слід звернути увагу на те, що:

- якщо величина зміни аргументу дорівнює одиниці, то його можна не вказувати;
- для заповнення вектор-стовпця компонентами з постійною величиною зміни аргументів слід транспонувати вектор-рядок;

- для візуалізації масивів даних необхідно вміти формувати вектори за допомогою оператора двокрапки і проводити поелементні операції.

Вправа 2. Індексація векторних даних.

Система **MATLAB** містить значний набір вбудованих функцій для обробки векторних даних. Повний лист таких функцій виводиться в командне вікно за допомогою команди **help datafun**, а для отримання детальньої інформації про кожну з функцій потрібно вказати її ім'я як аргумент команди **help** [44]. Зверніть увагу на те, що ряд команд допускають звернення до них як з одним, так і з двома вхідними параметрами. Вихідні параметри результату записуються в квадратні дужки і відділяються один від одного комами.

Дуже часто потрібно виконати дії з тільки частиною вектора, або звернутись до деяких компонентів вектора. Для цього необхідно провести індексування компонентів вектора. Розглянемо правила системи **MATLAB**, згідно яких проводиться така індексація компонентів вектора. Для виділення елементу вектора необхідно вказати його номер в круглих дужках відразу після імені змінної, в якій міститься вектор.

Приклад 1. Знайти суму першого і третього компонентів вектора $\bar{v} = (5, 6, 7, 3, 6, 0)$ та вивести у командний рядок останній компонент вектора $\bar{v}$.

Суму елементів вектора можна знайти таким чином:

```
>> v=[5 6 7 3 6 0];s=v(1)+v(3)
s =
   12
```

Виділення останнього компоненту вектора можна провести за допомогою звернення до вектору **v** з використанням команди **end** як вхідного параметру:

```
>> v(end)
ans =
    0
```

Вказівку номерів компонентів вектора можна використовувати і при формуванні векторів, якщо послідовно додавати нові елементи (не обов'язково в порядку зростання їх номерів).

Приклад 2. Сформувати вектор-рядок $\bar{d}$, компоненти якого є числа: 1, 7, 0, 4.

Операцію утворення вектор-рядка можна здійснити за допомогою наступних операцій:

```
>> d=1;d(2)=7;d(4)=4;d
d =
    1    7    0    4
```

Зверніть увагу, що для введення першого компонента не обов'язково вказувати його індекс, оскільки при виконанні оператора присвоювання створюється вектор (масив розміру 1×1). Наступні відповідні оператори присвоювання приводять до автоматичного збільшення довжини вектора, а пропущені компоненти (у нашому випадку **d(3)**) набувають значення нуль.

Оператор індексації **«:»** дозволяє виділити компоненти (що йдуть підряд) в новий вектор. Початковий і кінцевий номери вказуються в круглих дужках через двокрапку.

Приклад 3. Створити вектор, компонентами якого являються 2-й і 3-й компоненти вектора $\overline{d}$.

Формування шуканого вектора відбувається за допомогою виконання однієї операції:

```
>> dnew=d(2:3)
dnew =
    7     0
```

Застосування вбудованих функцій обробки даних до деяких послідовно розташованих компонентыв вектора виконується за допомогою відомих стандартних команд (дивись лабораторну роботу № 5, завдання 1).

Приклад 4. Обчислити добуток компонентів вектора $\overline{w} = (-4, -3, -1, 4, 3, 9)$ з другого по п’ятий та створити новий вектор з 1-го, 3-го, 6-го компонентів вектора $\overline{w}$.

Операція покомпонентного добутку виконується за допомогою команди **prod**:

```
>> disp(prod(w(2:5)))
   36
```

Для виділення компонентів із заданими індексами в новий вектор застосовується спеціальна команда **ind,** яка повиненна утримувати номери необхідних елементів:

```
>> ind=[1 3 6];dn=w(ind)
dn =
   –4    –1     9
```

Приклад 5. Знайти суму компонентів вектора $\overline{v} = (5, 6, 7, 3, 6, 0)$ з парними індексами.

Ця процедура буде виконуватись за допомогою команди **sum**:

```
>> ind=2:2:length(v);
>> s=sum(v(ind))
s =
    9
```

Створення нових векторів з компонентів заданих векторів проводиться за допомогою квадратних дужок.

Приклад 6. Сформувати вектор **vu**, в якому перші три елемента утворені з перших трьох елементів вектора $\bar{v} = (5, 6, 7, 3, 6, 0)$, а два останніх елемента – це п'ятий та шостий елементи вектора $\bar{u} = (1, 3, 6, 7, 9, 9)$. Операція формування нового вектора виконується наступним чином:

```
>> vu=[v(1:3) u(5:end)]
vu =
    5     6     7     9     9
```

Вправа 3. Формування матриць, основні операції над матрицями.

Матриці невеликих розмірів зручно вводити з командного рядка. Існують три способи формування матриць [20].

Наприклад, матрицю можна сформувати таким чином: набрати в командному рядку (розділяючи елементи рядка матриці пропусками): A=[0.7 –2.5 9.1 і натиснути клавішу **Enter**. Курсор переміщається в наступний рядок (символ запрошення командного рядка >> не з'являється). Елементи кожного наступного рядка матриці записуються через пропуск, а операція введення рядка завершується натисненням на клавішу **Enter**. Після введення останнього рядка в кінці ставиться квадратна дужка.

Приклад 1. Сформувати у командному рядку матрицю:

$$A = \begin{bmatrix} 1.2 & 3.4 & 5.6 \\ 2 & 4 & 6 \\ -1.2 & -5 & 0.1 \end{bmatrix}.$$

Застосуємо вищенаведений спосіб:

```
>> A=[1.2 3.4 5.6
2 4 6
–1.2 –5 0.1]
```

Якщо після квадратної дужки не ставити крапку з комою (для вилучення висновку з командного вікна), то матриця буде виведена у вигляді таблиці.

Інший спосіб формування матриці заснований на передбаченні, що матрицю можна розглядати як вектор-стовпець, кожен елемент якого є рядком матриці. Оскільки крапка з комою використовується для розділення компонентів вектор-стовпця, то формування матриці здійснюється оператором присвоєння.

Приклад 2. Сформувати матрицю

$$B=\begin{bmatrix} 4.2 & 3.2 \\ 3.4 & 6.7 \\ 0 & 5.7 \end{bmatrix}$$

вищенаведеним способом.

Для цього виписується наступна операція:

```
>> B=[4.2 3.2;3.4 6.7;0 5.7];
```

Подібним чином можна утворити матрицю, при якій вона розглядається як вектор-рядок, кожен компонент якого є стовпцем цієї матриці.

Приклад 3. Утворити в командому рядку матрицю $C=\begin{bmatrix} 8 & 2 & 2 \\ 1 & 8 & 5 \end{bmatrix}$ та визначити інформацію про вищенаведені матриці A, B і матрицю C.

Для представлення матриці **C** достатньо виконати операцію:

```
>> C=[[8;1] [2;8] [2;5]];
```

Зверніть увагу на необхідність дотриммуватись правил запису (правил синтаксису) оператора, а саме, внутрішні квадратні дужки дійсно потрібні. Оператор **C=[8;1 2;8 2;5]** є недопустимим і приводить до повідомлення про помилку, оскільки виявляється, що в першому рядку матриці міститься тільки один елемент, в другому і третьому – по два, а в четвертому – знову один.

Для отримання інформації про матриці A, B і C робочого середовища (**WorkSpace**) скористаємось командою **whos**. У командне вікно виводиться таблиця з інформацією про розміри масивів, про пам'ять, що необхідна для зберігання кожного з масивів, і типу введених даних– **double array**:

```
>> whos A B C
  Name      Size              Bytes  Class
  A         3x3                  72  double array
  B         3x2                  48  double array
  C         2x3                  48  double array
  Grand total is 21 elements using 168 bytes
```

Команда **size** дозволяє встановити розміри масивів і вона представляє результат у вигляді вектора, перший компонент якого дорівнює числу рядків матриці, а другий – числу стовпців цієї матриці.

Приклад 4. Становити розмір матриці C.

Застосовуємо команду **size**:

```
>> size(C)
ans =
    2     3
```

Вправа 4. Елементарні операції з матрицями.

Додавання і вирахування матриць однакових вимірів проводиться з використанням знаків «+» та «–».

Для обчислення матричного добутку застосовується оператор **«*».** При використанні цього оператора передбачається, що відповідні розміри матриць, при яких можлива операція матричного добутку, повинні співпадати.

Приклад 1. Знайти добуток матриць і

$$B = \begin{bmatrix} 4.2 & 3.2 \\ 3.4 & 6.7 \\ 0 & 5.7 \end{bmatrix}.$$

Виконуємо операцію добутку матриць за допомогою оператора **«*»:**

```
>> M=A*B
M =
  16.6000   58.5400
  22.0000   67.4000
 –22.0400  –36.7700
```

Крім операції матричного добутку мають місце таки операції, як множення матриці на число і числа на матрицю. При цьому відбувається множення кожного елементу матриці на число і результатом є матриця тих же вимірів, що і вихідна.

Оператор «**'» (**апостроф**)** призначений для транспонування дійсної матриці або знаходження комплексно-спряженої матриці.

Для зведення квадратної матриці в степень застосовується оператор «**^».**

Приклад 2. Обчислити матричний вираз $R = (A - BC) - ABC$, в якому:

$$A = \begin{bmatrix} 1.2 & 3.4 & 5.6 \\ 2 & 4 & 6 \\ -1.2 & -5 & 0.1 \end{bmatrix},\ B = \begin{bmatrix} 4.2 & 3.2 \\ 3.4 & 6.7 \\ 0 & 5.7 \end{bmatrix},\ C = \begin{bmatrix} 8 & 2 & 2 \\ 1 & 8 & 5 \end{bmatrix}.$$

Нижче наведений запис цього виразу в системі **MATLAB**:

```
>> R=(A–B*C)–A*B*C
R =
–226.9400 –532.1200 –344.7000
 –275.3000 –639.6000 –415.3000
  206.1900  287.6400  199.5300
```

Суму елементів на головній діагоналі можна легко отримати за допомогою команди **diag**, яка визначає цю діагональ. Інша діагональ, що називається *антідіагоналлю*, не так поширено використовується, тому у системі **MATLAB** не має спеціальної команди для неї. Але команда **fliplr**, яка спочатку

передбачалася для використання в графіці дзеркально відображає матрицю зліва направо [24].

Приклад 3. Визначити головну діагональ та антідіагональ матриці А, знайти суму елементів головної діагоналі та антідіагоналі матриці А.

Спочатку визначимо головну діагональ матриці **А**:

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

Команда **sum** дозволяє виконувати операцію підсумовування діагональних елементів матриці **А**:

```
>> sum (diag (A) )
ans = 34
```

Команда **fliplr** визначає антідіагональ, а за допомогою команди **sum** підсумовуємо елементи антідіагоналі матриці **А**:

```
>> sum(diag(fliplr(A)))
ans = 34
```

Вправа 5. Операції індексування матриць.

У системі **MATLAB** існує кількість команд і способів для роботи з матричними даними. При звертанні до елементу двовимірного масиву слід вказати номер (індекс) рядка та номер стовпця в круглих дужках після імені масиву [46].

Приклад 1. Визначити елемент a_{23} матриці

$$A = [a_{ij}] = \begin{bmatrix} 1.2 & 3.4 & 5.6 \\ 2 & 4 & 6 \\ -1.2 & -5 & 0.1 \end{bmatrix},$$

утворити вектор-стовпець $A1$ з елементів матриці А: a_{23}, a_{33}, утворити вектор-рядок $A2$ з елементів матриці А: a_{21}, a_{22}, a_{23}.

Спочатку визначимо елемент a_{23}:

```
>> A(2,3)
ans =
    6
```

Індексація відновленого елементу за допомогою оператора **«:»** (двокрапка) дозволяє отримати визначену частину матриці – рядок, стовпець або блок:

```
>> A1=A(2:3,3)
A1 =
   6.0000
   0.1000
>> A2=A(2,1:3)
A2 =
    2    4    6
```

Для звернення до всього рядка або всього стовпця вихідної матриці не обов'язково вказувати за допомогою оператора «:» початковий (перший) і останній (заключний) індекси.

Таким чином, оператори **r2=R(2,1:3)** і **r2=R(2,:)** еквівалентні.

Для визначення елементів рядка або стовпця матриці від заданого до останнього можна використовувати параметр **end**, так само як і для векторів: **r2=R(2,1:end).** Виділення блоку, що складається з декількох рядків і стовпців, вимагає індексації двокрапкою як по першому вимірюванню, так і по другому.

Приклад 2. Нехай задана матриця

$$T = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 8 & 9 & 7 & 6 \\ 3 & 4 & 5 & 6 \\ 6 & 7 & 9 & 0 \end{bmatrix}.$$

Утворити нову матрицю за рахунок вибіркі елементів матриці T з другого рядка по третій і з елементів другого стовпця по четвертий.

Для виділення вказаних елементів необхідно виконати наступні операції:

```
>> T=[1 2 3 4;8 9 7 6;3 4 5 6;6 7 9 0];T1=T(2:3,2:4)
T1 =
    9    7    6
    4    5    6
```

Індексація за допомогою оператора **«:»** може застосовуватись також при здійсненні різних перестановок в масивах.

Приклад 3. Поміняти місцями перший та останній рядки матриці A.

Перестановка першого і останнього рядків в матриці реалізується послідовністю наступних команд:

```
>> z=A(1,:)
z =
  –1.2000   –5.0000    0.1000
>> A(1,:)=A(end,:)
A =
   1.2000    3.4000    5.6000
```

```
    2.0000    4.0000    6.0000
    1.2000    3.4000    5.6000
>> A(end,:)=z
A =
    1.2000    3.4000    5.6000
    2.0000    4.0000    6.0000
   –1.2000   –5.0000    0.1000
```

В системі **MATLAB** підтримується операція викреслення рядків або стовпців з вихідної матриці. Блоку елементів, що видаляються, треба присвоїти порожній масив, який задається за допомогою квадратних дужок.

Приклад 4. Викреслити другий і третій рядки з матриці T.

Ця операція проводиться наступною командою:

```
>> T(2:3,:)=[]
T =
    1    2    3    4
    6    7    9    0
```

Індексація оператором «:» спрощує формування матриць, що мають певну структуру.

Приклад 5. Сформувати матрицю

$$V = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Перший крок полягає у визначенні одиничної матриці розміру п'ять на п'ять; потім заповнюються перший і останній рядки цієї матриці та перший і останній стовпці. Перелік операцій наведений нижче:

```
>> V(1:5,1:5)=1;
>> V(1,:)=0;
>> V(end,:)=0;
>> V(:,1)=0;
>> V(:,end)=0
```

Переконайтесь, що в результаті отримується шукана матриця.

Вправа 6. Формування матриць за допомогою деяких команд у системі **MATLAB**.

В табл. 4.1 наведені ряд вбудованих команд, які дозволяють сформувати стандартні матриці заданих розмірів [20]. Зверніть увагу, що у всіх цих командах, окрім команди **diag**, дозволяється вказувати розміри матриці наступними способами:

- числами через кому (у двох вхідних параметрах);
- одним числом, результат операції – квадратна матриця;
- вектором з двохкомпонентів, які визначають число рядків і стовпців матриці.

Останній варіант дуже зручний, коли потрібно створити стандартну матрицю тих же розмірів, що і вихідна матриця.

Наприклад, команда **eye(size(A))** приводить до появи одиничної матриці, розміри якої співпадають з розмірами вихідної матриці A.

Приклад 1. За допомогою засобів **MATLAB** сформувати трьохдіагональну матрицю

$$F=\begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 5 & 3 & -3 & 0 & 0 & 0 \\ 0 & 0 & 5 & 4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 5 & -5 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 & -6 \\ 0 & 0 & 0 & 0 & 0 & 5 & 7 \end{bmatrix}$$

розміру 7×7.

Спочатку треба задати вектор f=$(a_1, a_2, ..., a_7)$, де a_i – цілі числа від одного до семи, $i=1, 2, ..., 7$. Потім за допомогою команди **diag** створюється діагональна матриця (матриця M1) і матриця з побічною правою діагоналлю (матриця M2). Матрицю розміром 6×6 з побічною лівою діагоналлю, яка містить п'ятірки, отримується за допомогою команди **ones**: **5*ones(1,6)**. Цю матрицю слід вказати першим параметром команди **diag**, а на місце другого параметру поставити мінус одиницю. Отримана матриця буде третьою допоміжною матрицею (матриця M3). Тепер досить виконати наступні матричні операції: M1-M2+M3.

Команди, що реалізують ці операції, мають вигляд:

```
>> F=diag(f)–diag(f(1:6),1)+diag(5*ones(1,6),–1)
```

Таблиця. 4.1 Команди для створення стандартних матриць

Функція	Результат операції;приклади формування команди
zeros	нульова матриця; F=zeros(4,5), F=zeros(3), F=zeros([3 4])
eye	одинична прямокутна матриця (одиниці розташовані на головній діагоналі); I=eye(5,8), I=eye(5), I=eye([5 8])
ones	матриця, що цілком складається з одиниць; E=ones(3,5), E=ones(6), E=ones([2 5])
rand	матриця, елементи якої – випадкові числа, рівномірно розподілені на інтервалі (0,1); R=rand(5,7), R=rand(6), R=rand([3 5])
randn	матриця, елементи якої – випадкові числа, розподілені по нормальному закону з нульовим середнім і дисперсією, яка дорівнює одиниці; N=randn(5,3), N=randn(9), N=randn([2 4])
diag	• діагональна матриця, елементи діагоналі якої задаються вектором у вхідному параметрі; D=diag(v); • виділення головної діагоналі з матриці A і утворення з елементів цієї діагоналі вектора d; d=diag(A); • виділення к-ї (побічної) діагоналі з матриці A і утворення з елементів цієї діагоналі вектора d; d=diag(A, k);

Вправа 7. Поелементні операції з матрицями та спеціальні матричні функції.

Поелементні операції з матрицями проводяться за схемами, що аналогічні операціям з вектор-стовпцями та вектор-рядками [20]. Але необхідно стежити за відповідністю вимірів вихідних матриць:

- **A.*B, A./B** (оператори **«.*», «./»)** – операція поелементного множення та поелементного ділення;
- **A.^p** (оператор «.^») – операція поелементного возведення до степені **p**;
- **A.^B** (оператор «.^») – операція піднесення елементів матриці **A** до степені, які визначаються відповідними елементами матриці B (матриці повинні мати однакові виміри);
- **A.'** – транспонування матриці (для дійсних матриць операції A' і A.' приводять до однакових результатів; операція «.'» є операцією поелментного виконання).

Коли потрібно "розвернути" вихідну матрицю на 90° проти годинникової стрілки слід використати команду **rot90.**

Приклад 1. Виконати розворот на 90° матриці $D = \begin{bmatrix} 5 & 6 & 7 \\ 3 & 4 & 5 \end{bmatrix}$.

Операція розвороту виконуєтьсяза допомогою наступних команд:

```
>> D=[5 6 7;3 4 5];
>> R=rot90(D)
```

```
R =
    7    5
    6    4
    5    3
```

Допускається записувати суму або різницю матриці і числа. При цьому операції додавання або відрахування застосовується, відповідно, до всіх елементів матриці. Формування математичної функції від матриці приводить до матриці того ж розміру, на відповідних позиціях (елементах) якої стоять значення функції від елементів вихідної матриці [44,с.88].

У системі **MATLAB** визначені команди **sqrtm (**знаходження поелементного квардатного кореня з матриці**)** і **expm (**матрична експонента**)** для утворення матричних функцій наступні матричні функції.

Приклад 2. Знайти квадратний корінь з матриці

$$H = \begin{bmatrix} 5 & 2 & 1 \\ 4 & 6 & 1 \\ 6 & 3 & 4 \end{bmatrix}$$

перевірити результат за допомогою правила матричного множення.

Шукана операція виконується за допомогою наступних команд:

```
>> H=[5 2 1;4 6 1;6 3 4];
>> S=sqrtm(H)
S =
   2.0790   0.4215   0.2319
   0.8430   2.3530   0.1896
   1.3911   0.5689   1.8893
>> S*S
ans =
   5.0000   2.0000   1.0000
   4.0000   6.0000   1.0000
   6.0000   3.0000   4.0000
```

Рекомендуємо виконати формування матричної експоненти від матриці H з використанням команди **expm**.

Спеціальна команда **funm** служить для обчислення довільної матричної функції.

Приклад 3. Обчислити елементи матриці, якщо кожен елемент її – це елемент магічної матриці, за допомогою якої формується матрична функція, а $\sin$ – функція для обчислення матриці.

Ця задача виконується за допомогою команди:

```
>> F = funm(magic(3),@sin)
```

```
F =
  –0.3850    1.0191    0.0162
   0.6179    0.2168   –0.1844
   0.4173   –0.5856    0.8185
```

Всі команди для обробки даних (одновимірні масиви), які використовуються у випадку вектор-стопців та вектор-рядків, можуть бути застосовані і до двовимірних масивів. Основна відмінність від обробки векторних даних полягає в тому, що ці команди працюють з двовимірними масивами по стовпцях, наприклад, команда **sum** підсумовує елементи кожного із стовпців і повертає вектор-рядок, довжина якого дорівнює числу стовпців вихідної матриці.

Приклад 4. Знайти суму рядків та стовпців матриці N та загальну суму елементів матриці.

Перелік команд наведений нижче:

```
>> N=[1 2 3; 4 5 6; 7 8 9]
N =
    1    2    3
    4    5    6
    7    8    9
>> s=sum(N)
s =
   12   15   18
```

Якщо вибрати другим вхідним параметром команди **sum** величину 2, то підсумовування відбудеться по рядках:

```
>> s=sum(N,2)
s =
    6
   15
   24
```

Для обчислення суми всіх елементів матриці потрібно двічі послідовно застосувати команду **sum**:

```
>> s=sum(sum(N))
s =
   45
```

У системі **MATLAB** є можливість конструювання матриць на основі використання матриць меньших розмірів.

Приклад 5. Нехай задані квадратні матриці другого порядку **M1, M2, M3, M4**. Потрібно скласти з матриць **M1**, **M2, M3, M4** блочну матрицю **M**:

$$M = \begin{bmatrix} M1 & M2 \\ M3 & M4 \end{bmatrix}.$$

Формуємо у командних рядках вихідні матриці:

```
>> M1=[1 2;3 4];
>> M2=[5 6;7 8];
>> M3=[4 3;2 1];
>> M4=[8 7;6 5];
```

В цьому прикладі вважаємо, що матриця **M** має розміри два на два, а кожен елемент цієї матриці представляється, відповідно, матрицями **M1, M2, M3, M4**. Отже, для отримання в робочому середовищі системи **MATLAB** матриці **M** достатньо використати оператор:

```
>> M=[M1 M2; M3 M4]
```