Звіт

до лабораторної роботи № 1

варіант № 2

з дисципліни

«Прикладна математика»

Студентки групи ІПЗ-26

Ващіліної Діани Олександрівни

на тему:

«ЗАСТОСУВАННЯ ЛІНІЙНОЇ АЛГЕБРИ з використанням математичних бібліотек мови програмування Python»

Варіант 2

Основні теоретичні відомості:

Для роботи буде використовуватися мова програмування Python та її бібліотеки numpy та sympy. Ці бібліотеки дозволяють значно скоротити час вирішення математичних задач, навіть якщо навички програмування значно слабкіші ніж математичні.

Завдання 1 (№ 2.1.2)

Умова:

Знайти f(A), якщо:

$$\mathbf{2.1.2}.\ A=\begin{pmatrix}3 & 4 & 0\\ 1 & 2 & 3\\ -3 & 1 & -1\end{pmatrix},\ f(x)=x^2+5x+2.$$

Хід рішення:

Використавши бібліотеку numpy створимо матрицю. Спочатку піднесемо матрицю А до квадрата функцією np.linalg.matrix_power(), потім кожен її елемент помножимо на 5. Останнім кроком до матриці А додаємо 2 помножене на одиничну матрицю, адже відняти від матриці саме по собі «2» не має сенсу.

Код:

```
import numpy as np
A = np.array([[3,4,0],
              [1,2,3],
              [-3,1,-1]])
result = np.linalg.matrix_power(A,2) + 5 * A + 2 * np.eye(3)
                                        #* np.identity(3)

print(result)
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\numpy_lab'; & 'C:\Python311\pyt
/..\debugpy\launcher' '57129' '--' 'e:\dz ipz\numpy_lab\lab_1\2.1.2.py'
[[ 30.  40.  12.]
 [  1.  23.  18.]
 [-20.  -6.   1.]]
```

Перевірка результату:

Для перевірки проведемо дії над матрицею вручну:

№2.1.2.

Знайти $f(A)$ якщо:

$$A = \begin{pmatrix} 3 & 4 & 0 \\ 1 & 2 & 3 \\ -3 & 1 & -1 \end{pmatrix}, \quad f(x) = x^2 + 5x + 2$$

$f(A)$ - ?

$$f(A) = A^2 + 5A + 2E$$

$$1.\ A^2 = A \cdot A = \begin{pmatrix} 3 & 4 & 0 \\ 1 & 2 & 3 \\ -3 & 1 & -1 \end{pmatrix} \begin{pmatrix} 3 & 4 & 0 \\ 1 & 2 & 3 \\ -3 & 1 & -1 \end{pmatrix} =$$

$$= \begin{pmatrix} 3\cdot3+4\cdot1+0\cdot(-3) & 3\cdot4+4\cdot2+0\cdot1 & 3\cdot0+4\cdot3+0\cdot(-1) \\ 1\cdot3+2\cdot1+3\cdot(-3) & 1\cdot4+2\cdot2+3\cdot1 & 1\cdot0+2\cdot3+3\cdot(-1) \\ -3\cdot3+1\cdot1+(-1)\cdot(-3) & -3\cdot4+1\cdot2+(-1)\cdot1 & -3\cdot0+1\cdot3+(-1)(-1) \end{pmatrix} =$$

$$= \begin{pmatrix} 13 & 20 & 12 \\ -4 & 11 & 3 \\ -5 & -11 & 4 \end{pmatrix}$$

$$2.\ A^2 + 5A = \begin{pmatrix} 13 & 20 & 12 \\ -4 & 11 & 3 \\ -5 & -11 & 4 \end{pmatrix} + \begin{pmatrix} 15 & 20 & 0 \\ 5 & 10 & 15 \\ -15 & 5 & -5 \end{pmatrix} =$$

$$= \begin{pmatrix} 28 & 40 & 12 \\ 1 & 21 & 18 \\ -20 & -6 & -1 \end{pmatrix}.$$

$$3.\ (A^2 + 5A) + 2E = \begin{pmatrix} 28 & 40 & 12 \\ 1 & 21 & 18 \\ -20 & -6 & -1 \end{pmatrix} + \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} =$$

$$= \begin{pmatrix} 30 & 40 & 12 \\ 1 & 23 & 18 \\ -20 & 6 & 1 \end{pmatrix}$$

Відповіді повністю збігаються, тому результат роботи можна вважати перевіреним.

Завдання 2 (№2.2.2)

Умова:

Розв’язати матричні рівняння

2.2.2. $X \cdot A = B$, $A = \begin{pmatrix} 4 & 3 \\ 1 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 8 \\ 1 & 1 \\ 0 & -1 \end{pmatrix}$.

Хід рішення:

Використавши бібліотеку numpy розв’язуємо систему лінійних рівнянь. Оскільки матриці A і B не є сумісними, ми спочатку обчислюємо обернену матрицю A (якщо вона існує) за допомогою np.linalg.inv(A), назвавши обернену матрицю A1. Після, множимо матрицю B на матрицю A1 (обернену A) для знаходження матриці X, яка є розв'язком.

Код:

```
import numpy as np
#A * X = B
A = np.array([[4,3],
            [1,1]])
B = np.array([[2,8],
            [1,1],
            [0,-1]])
#матриці є неузгодженими, тож X * A * A^-1 = B * A^-1,   X * E = B * A^-1,    X=B*A^-1
#знаходжу обернену матрицю A
A1 =  np.linalg.inv(A)

X = np.dot(B,A1)
print(X)
```

Скріншот виконання:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\dz ipz\numpy_lab>  & 'C:\Python311\python.exe' 'c:\Users\diana\.vscode\extensi
_lab\lab_1\2.2.2.py'
[[-6. 26.]
 [ 0.  1.]
 [ 1. -4.]]
```

Перевірка:

Перевіримо відповідь за допомогою Python, за допомогою знаходження матриці B шляхом множення X*A. Код перевірки підставимо до вже написаного рішення задачі на новому рядку.

Код:

```
print("Перевірка\n", np.dot(X,A))
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\numpy_lab'; & 'C:\Python311\py
/..\debugpy\launcher' '57259' '--' 'e:\dz ipz\numpy_lab\lab_1\2.2.2.py'
Перевірка
 [[ 2.  8.]
 [ 1.  1.]
 [ 0. -1.]]
```

Результати відповіді, що відповідають матриці В співпадають з даними в умові значеннями. Отже, результат роботи можна вважати перевіреним.

Завдання 3 (№2.4.2)

Умова:

Знайти ранг матриці

2.4.2. $\begin{pmatrix} 1 & 1 & -2 & 1 & 3 \\ -1 & 2 & 1 & -1 & -2 \\ 3 & 0 & 2 & 1 & -1 \\ 3 & 3 & 1 & 1 & -1 \end{pmatrix}$.

Хід рішення:

Використавши бібліотеку numpy створимо матрицю. Використавши функцію np.linalg.matrix_rank() обчислимо ранг матриці.

Код:

```
import numpy as np
A = np.array([[ 1, 1,-2, 1, 3],
              [-1, 2, 1,-1,-2],
              [ 3, 0,  2, 1,-1],
              [ 3, 3, 1, 1,-1]])
print(np.linalg.matrix_rank(A))
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\numpy_lab'; & 'C:\Python311\py
/..\debugpy\launcher' '57159' '--' 'e:\dz ipz\numpy_lab\lab_1\2.4.1.py'
4
```

Перевірка результату:

Перевіримо ранг матриці вручну. Виконаємо над матрицею елементарні перетворення, впишемо мінор 4-го порядку, взявши всі рядки та 1,2,3,5 ствопці:

№2.4.2.

$$\begin{pmatrix} 1 & 1 & -2 & 1 & 3 \\ -1 & 2 & 1 & -1 & -2 \\ 3 & 0 & 2 & 1 & -1 \\ 3 & 3 & 1 & 1 & -1 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & -2 & 1 & 3 \\ -3 & 0 & 5 & -3 & -8 \\ 3 & 0 & 2 & 1 & -1 \\ 0 & 0 & 7 & -2 & -10 \end{pmatrix} \sim$$

$$\sim \begin{pmatrix} 1 & 1 & -2 & 1 & 3 \\ 0 & -3 & 5 & -3 & -8 \\ 0 & 3 & 2 & 1 & -1 \\ 0 & 0 & 7 & -2 & -10 \end{pmatrix} \sim \begin{pmatrix} 1 & 1 & -2 & 1 & 3 \\ 0 & -3 & 5 & -3 & -8 \\ 0 & 0 & 7 & -2 & -9 \\ 0 & 0 & 7 & -2 & -10 \end{pmatrix} \sim$$

$$\sim \begin{pmatrix} 1 & 1 & -2 & 1 & 3 \\ 0 & -3 & 5 & -3 & -8 \\ 0 & 0 & 7 & -2 & -9 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Випишемо мінор 4-го порядку, взявши всі рядки та 1, 2, 3, 5 стовпці

$$M^{(4)} = \begin{vmatrix} 1 & 1 & -2 & 3 \\ 0 & -3 & 5 & -8 \\ 0 & 0 & 7 & -9 \\ 0 & 0 & 0 & -1 \end{vmatrix} = 1 \cdot (-3) \cdot 7 \cdot (-1) = 21 \neq 0$$

Найвищий порядок мінора відмінного від 0 $\neq 4$

Отже, ранг $A = 4$.

Відповідь збігається, тому результат роботи можна вважати перевіреним.

Завдання 4 (№3.1.2)

Умова:

Розв’яжіть систему: а) матричним методом; б) за формулами Крамера

3.1.2. $\begin{cases} 2x_1 + x_2 + x_3 = 2, \\ x_1 + 2x_2 + x_3 = 3, \\ x_1 + x_2 + 2x_3 = -1. \end{cases}$

Хід рішення:

Використавши бібліотеку numpy, запишемо систему у матричній формі: основну матрицю A, матрицю правої частини B. Матриця X- є матрицею невідомих. Спершу рахуємо визначник матриці за допомогою функції np.linalg.det(), щоб переконатися що він не дорівнює нулю і матрицю можливо розв'язати, визначник використаємо для розв'язку за формулою Крамера. Далі, розв'язуємо матричним методом. Для цього знаходиться обернена матриця A1 за допомогою np.linalg.inv(). Знаходяться невідомі X за допомогою множення оберненої матриці на матрицю правої частини B. Після чого, розв'язуємо методом Крамера. Формуються три матриці C, D, і F, замінюючи відповідно, по черзі, стовпці матриці A стовпцями матриці B. Знаходяться визначники цих матриць (deltaX1, deltaX2, deltaX3). Знаходяться невідомі X1, X2, і X3 за допомогою відношення deltaX до визначника deltaA. Виводяться результати обчислень для невідомих X1, X2, і X3. Невідомі виводяться у матрицю невідомих X_kramer.
Код:

```
import numpy as np
# запишемо систему у матричній формі AX=B —> X = A^-1 * B
#основна матриця системи А
A = np.array([[ 2, 1, 1],
            [ 1, 2, 1],
            [ 1, 1, 2]])
#матриця правої частини В
B = np.array([[ 2],
            [ 3],
            [-1]])

#визначник А
deltaA = np.linalg.det(A)
print("Переконаємось що визначник не = 0, визначник матриці А:  ",
deltaA)
```

```
# а) - матричним методом
#обернена матриця -
A1 =  np.linalg.inv(A)

X = np.dot(A1, B)
print("Матричний метод, матриця невідомих X:")
print(X)

# б) - за формулою Крамера
#по черзі b1,b2,b3 замість стовпців
C = np.array([[ 2, 1, 1],
             [ 3, 2, 1],
             [-1, 1, 2]])
deltaX1 = np.linalg.det(C)

D = np.array([[ 2, 2, 1],
             [ 1, 3, 1],
             [ 1,-1, 2]])
deltaX2 = np.linalg.det(D)

F = np.array([[ 2, 1, 2],
             [ 1, 2, 3],
             [ 1, 1,-1]])
deltaX3 = np.linalg.det(F)

X1 = round(deltaX1/deltaA)
X2 = round(deltaX2/deltaA)
X3 = round(deltaX3/deltaA)
print("За формулою Крамера X1: ", X1, ", X2: ", X2, ", X3: ", X3)

X_kramer = np.array([[X1],[X2],[X3]])
print("Крамер, матриця невідомих X:\n", X_kramer)
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\numpy_lab'; & 'C:\Pytl
 '54742' '--' 'e:\dz ipz\numpy_lab\lab_1\3.1.2.py'
Переконаємось що визначник не = 0, визначник матриці A:   4.0
Матричний метод, матриця невідомих X:
[[ 1.]
 [ 2.]
 [-2.]]
За формулою Крамера X1:  1 , X2:  2 , X3:  -2
Крамер, матриця невідомих X:
 [[ 1]
 [ 2]
 [-2]]
```

Перевірка результату:

Перевіримо відповідь за допомогою Python, знайшовши матрицю B шляхом множення X*A. Код перевірки підставимо до вже написаного рішення задачі на новому рядку.

Код:

```
print("Перевірка, де B:")
print(np.dot(A,X))
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\numpy_lab'; &
/..\debugpy\launcher' '57293' '--' 'e:\dz ipz\numpy_lab\l
Перевірка, де B:
[[ 2.]
 [ 3.]
 [-1.]]
```

Результати відповіді, що відповідають матриці B співпадають з даними в умові значеннями. Отже, результат роботи можна вважати перевіреним.

Завдання 5 (№5.1.2)

Умова:

Обчисліть:

5.1.2. а) $(2\vec{a}+5\vec{b})(3\vec{a}-2\vec{b})$; б) $|\vec{a}-3\vec{b}|$, якщо $|\vec{a}|=3$, $|\vec{b}|=4$, $\varphi=\dfrac{2\pi}{3}$.

Хід рішення:

Використавши бібліотеку sympy і стандартного модуля math, обчислимо. Визначимо символьні змінні a і b, які використовуватимуться для створення виразів. Обчислимо значення косинуса кута fi, яке дорівнює cos(2π/3) та округлимо його до десятих. Для вирішення підпункту а- XA, розкриваємо вираз за допомогою sp.expand(). Потім за допомогою evalf() звертаємось до минулого виразу, і за допомогою subs замінюємо a, b, a*b на числові значення, результат округлюємо до цілого числа. Для підпункту б) – XB, записуємо данний вираз з розкритим модулем у форматі кореню з квадратів виразу. Аналогічно підпункту а, розв'язуємо підпункт б скориставшись evalf() та subs() і округливши результат до сотих.

Код:

```
import sympy as sp
import math
# модуль a=3, модуль b=4
a = sp.Symbol('a')
b = sp.Symbol('b')
fi= 2 * math.pi/3
fiCos = round(math.cos(fi), 1)
print("косінус fi: ", fiCos)

# XA - a)
XA = sp.expand((2*a + 5*b)*(3*a-2*b))
print("a): ",XA)
XA=  XA.evalf(subs={a**2:9,b**2:16,a*b:3*4*fiCos})
#subs підставляє і обчислює
#X2.evalf до A звертається до мин. приклада
print("відповідь а): ",round(XA))

# XB - б)
XB= sp.expand(sp.sqrt((a-3*b)**2))
print("б): ",XB)
XB=  XB.evalf(subs={a**2:9,b**2:16,a*b:3*4*fiCos})
print("відповідь б): ",round((XB), 2))
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\numpy_lab'; & 'C:\Python
 '54765' '--' 'e:\dz ipz\numpy_lab\lab_1\5.1.2.py'
косінус fi:  -0.5
a):  6*a**2 + 11*a*b - 10*b**2
відповідь а):  -172
б):  sqrt(a**2 - 6*a*b + 9*b**2)
відповідь б):  13.75
PS E:\dz ipz\numpy_lab> 
```

Перевірка результату:

Перевіримо розв’язавши вручну:

№5.1.2.

а) $(2\bar{a}+5\bar{b})(3\bar{a}-2\bar{b})$; б) $|\bar{a}|=3,\ |\bar{b}|=4,\ \varphi=\frac{2\pi}{3}$ 120

$= 2\bar{a}\cdot 3\bar{a} - 2\bar{a}\cdot 2\bar{b} + 5\bar{b}\cdot 3\bar{a} - 5\bar{b}\cdot 2\bar{b} =$

$= 6\bar{a}^2 - 4\bar{a}\bar{b} + 15\bar{b}\bar{a} - 10\bar{b}^2 = 6|a|^2 - 4\bar{a}\bar{b} + 15\bar{a}\bar{b}$

$-10|b|^2 = 6|a|^2 + 11\bar{a}\bar{b} - 10|b|^2 =$

$= 6|a|^2 + 11|a||b|\cdot\cos\varphi - 10|b|^2 =$

$6\cdot 9 + 11\cdot 3\cdot \cancel{4}^{2}\cdot\left(-\frac{1}{\cancel{2}}\right) - 10\cdot 16 = 54 - 66 - 160 = -172$

б) $|\bar{a}-3\bar{b}| = \sqrt{(\bar{a}-3\bar{b})(\bar{a}-3\bar{b})} = \sqrt{\bar{a}^2 - 6\bar{a}\bar{b} + 9\bar{b}^2} =$

$= \sqrt{|a|^2 - 6|a||b| + 9|b|^2} =$

$= \sqrt{9 - 6\cdot 12\cdot \cos 120 + 9\cdot 16} =$

$= \sqrt{9 - 6\cdot \cancel{12}^{6}\cdot\left(-\frac{1}{\cancel{2}}\right) + 144} = \sqrt{9 + 36 + 144} = \sqrt{189} \approx$

$\approx 13{,}75$

Відповідь збігається, тому результат роботи можна вважати перевіреним.

Завдання 6 (№5.3.2)

Умова:

Знайдіть вектор x , якщо:

5.3.2. $\vec{x}\cdot(\vec{i}-\vec{j}+2\vec{k})=2$, $\vec{x}\cdot(\vec{i}-\vec{j}+5\vec{k})=8$, $\vec{x}\cdot(\vec{i}-4\vec{j}+8\vec{k})=2$.

Хід рішення:

Використавши бібліотеку numpy створимо матрицю A з координатами a, i, j. Матриця B- це права сторона системи. X- вектор, що ми шукаємо. X*A=B, X=B* A1 - таким чином за допомогою np.dot() множимо B на обернену матрицюдо матриці A- A1.

Код:

```
import numpy as np
A= np.array([[ 1,-1, 2],
            [ 1,-1, 5],
            [ 1,-4, 8]])
B=np.array([2,8,2])
A1 = np.linalg.inv(A)
X=np.dot(B, A1)
print(np.round(X))
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>
/..\debugpy\launcher' '57
[-1.  6. -3.]
```

Перевірка результату:

Перевіримо відповідь за допомогою Python, знайшовши матрицю B шляхом множення X*A. Код перевірки підставимо до вже написаного рішення завдання на новому рядку.

Код:

```
print("Перевірка\n", np.dot(X,A))
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\num
/..\debugpy\launcher' '57228' '--' 'e:\dz ipz\
Перевірка
 [2. 8. 2.]
```

Результати відповіді, що відповідають матриці B співпадають з даними в умові значеннями. Отже, результат роботи можна вважати перевіреним.

Завдання 7 (№6.2.2)

Умова:

Обчисліть площу грані ABC і об’єм піраміди ABCD, вершини якої містяться в точках:

6.2.2. $A(-3; 5; 4)$, $B(0; 0; 8)$, $C(-1; 3; -2)$, $D(2; 6; 1)$.

Хід рішення:

За допомогою бібліотеки numpy, створюємо масиви, точки A, B, C, D. Після, порахуємо вектори як різницю координат між точками.

Після цього, обчислимо векторний добуток векторів vAB і vAC за допомогою np.cross(), що дає вектор ABxAC. Площа грані обчислюється як половина модулю np.linalg.norm() цього векторного добутку. Далі, формується матриця ABCD, в якій рядки представляють вектори vAB, vAC, і vAD. Знаходиться визначник цієї матриці за допомогою np.linalg.det(), який дорівнює мішаним добутку векторів vAB, vAC, і vAD. Об'єм піраміди ABCD обчислюється як модуль визначника, поділений на 6.

Код:

```
import numpy as np
A = np.array([-3,5,4])
B = np.array([0,0,8])
C = np.array([-1,3,2])
D = np.array([2,6,1])

#vAB - вектор AB і т.д.
vAB= B - A
vAC= C - A
vAD= D – A

# векторний добуток np.cross()
ABxAC = np.cross(vAB, vAC)
# np.linalg.norm() - знаходимо довжину(модуль) вектора
S_ABC = np.linalg.norm(ABxAC) / 2

print("Площа грані ABC")
print(round(S_ABC, 2))


ABCD = np.array([vAB, vAC, vAD])
#мішайни добуток = визначнику, тож
detABCD = round(np.linalg.det(ABCD))
#np.abs - це модуль
V_ABCD = np.abs(detABCD / 6 )
print("Об'єм піраміди ABCD")
print(round(V_ABCD, 2))
```

Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\numpy_lab'; & 'C:\Python311\python.exe' 'c:\Users\diana\
 '50511' '--' 'e:\dz ipz\numpy_lab\lab_1\6.2.2 copy.py'
Площа грані ABC
11.58
Об'єм піраміди ABCD
15.33
```

Хід рішення - варіант 2:

Для розв'язку, скористаємось аналогічними методами та бібліотекою, але порахуємо вектори як різницю координат між точками, де віднімемо кожну координату по черзі вручну. Це є більш часозатратний хід розв'язку.

Код:

```
import numpy as np
A = np.array([-3,5,4])
B = np.array([0,0,8])
C = np.array([-1,3,2])
D = np.array([2,6,1])

#vAB - вектор AB і т.д.
vAB= np.array( [B[0] - A[0], B[1] - A[1], B[2] - A[2]])
vAC= np.array( [C[0] - A[0], C[1] - A[1], C[2] - A[2]])
vAD= np.array( [D[0] - A[0], D[1] - A[1], D[2] - A[2]])

# векторний добуток np.cross()
ABxAC = np.cross(vAB, vAC)
# np.linalg.norm() - знаходимо довжину(модуль) вектора
S_ABC = np.linalg.norm(ABxAC) / 2

print("Площа грані ABC")
print(round(S_ABC, 2))


ABCD = np.array([vAB, vAC, vAD])
#мішайни добуток = визначнику, тож
detABCD = round(np.linalg.det(ABCD))
#np.abs - це модуль
V_ABCD = np.abs(detABCD / 6 )
print("Об'єм піраміди ABCD")
print(round(V_ABCD, 2))
```

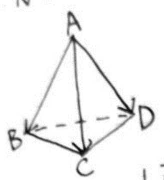
Скріншот виконання:

```
PS E:\dz ipz\numpy_lab>  e:; cd 'e:\dz ipz\numpy_lab'; & 'C:
 '54801' '--' 'e:\dz ipz\numpy_lab\lab_1\6.2.2.py'
Площа грані ABC
11.58
Об'єм піраміди ABCD
15.33
```

Перевірка результату:

Перевіримо виконання, розв’язавши вручну:

№ 6.2.2 $A(-3,5,4),\ B(0,0,8),\ C(-1,3,2),\ D(2,6,1)$



$$S_{ABC} = \frac{1}{2}\left|\overline{AB} \times \overline{AC}\right|$$

$$V_{ABC} = \frac{1}{6}\left|\overline{AB} \cdot \overline{AC} \cdot \overline{AD}\right|$$

1;2 => 3 => з мінусом $(-1)^{k+i}$

$a_y \cdot b_z - a_z \cdot b_y;$

векторний добуток

$$\bar{a} \times \bar{b} = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = \begin{vmatrix} a_y & a_z \\ b_y & b_z \end{vmatrix} \cdot \bar{i} - \begin{vmatrix} a_x & a_z \\ b_x & b_z \end{vmatrix} \cdot \bar{j} + \begin{vmatrix} a_x & a_y \\ b_x & b_y \end{vmatrix} \cdot \bar{k};$$

$$\overline{AB}(0-(-3);\ 0-5;\ 8-4) = (3;\ -5;\ 4)$$

$$\overline{AC}(-1+3;\ 3-5;\ 2-4) = (2;\ -2;\ -2)$$

$$\overline{AD}(2+3;\ 6-5;\ 1-4) = (5;\ 1;\ -3)$$

$$\overline{AB} \times \overline{AC} = \begin{vmatrix} \bar{i} & \bar{j} & \bar{k} \\ 3 & -5 & 4 \\ 2 & -2 & -2 \end{vmatrix} = \begin{vmatrix} -5 & 4 \\ -2 & -2 \end{vmatrix} \cdot \bar{i} - \begin{vmatrix} 3 & 4 \\ 2 & -2 \end{vmatrix} \cdot \bar{j} + \begin{vmatrix} 3 & -5 \\ 2 & -2 \end{vmatrix} \bar{k} = \overset{18}{(+10+8)}\,i - \overset{-14}{(-6-8)}\,j +$$

$$+(-6+10)k = -2i+2j+4k \qquad (18;\ 14,\ 4)$$

$$\left|\overline{AB} \times \overline{AC}\right| = \sqrt{(18)^2+14^2+4^2} = \sqrt{324+196+16} = \sqrt{536} =$$

$$S_{ABC} = \frac{1}{2}\sqrt{536} \approx 11{,}5$$

$$\bar{a}\,\bar{b}\,\bar{c} = \begin{vmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \end{vmatrix}$$

$$\overline{AB} \cdot \overline{AC} \cdot \overline{AD} = \begin{vmatrix} 3 & -5 & 4 \\ 2 & -2 & -2 \\ 5 & 1 & -3 \end{vmatrix} = 3 \cdot \begin{vmatrix} -2 & -2 \\ 1 & -3 \end{vmatrix} + 5 \cdot \begin{vmatrix} 2 & -2 \\ 5 & -3 \end{vmatrix} +$$

$$+4\begin{vmatrix} 2 & -2 \\ 5 & 1 \end{vmatrix} = 3 \cdot (6+2) + 5 \cdot (-6+10) + 4(2+10)$$

$$= 24+20+48 = 92$$

$$V_{ABC} = \frac{1}{6} \cdot 92 = 15{,}33$$

Відповіді збігаються, тому результат роботи можна вважати перевіреним.

Висновок:

Для виконання лабораторної роботи були використані бібліотеки SymPy та NumPy для розв'язання різних математичних завдань. SymPy використовувався для символьних обчислень та маніпуляцій з алгебраїчними виразами, в той час як NumPy дозволив розв'язати системи лінійних рівнянь та виконати чисельні операції з матрицями.

Завдяки цим бібліотекам, можна значно економити час та зручно вирішувати складні математичні задачі, зокрема, обчислення визначників, обернених матриць, векторних операцій. Під час роботи було показано, що коректно використовуючи функції бібліотек, можна отримати точні та надійні результати, які можна перевірити.

Важливою властивістю бібліотек є можливість автоматизувати обчислення та спростити складні завдання, а також зменшити ризик помилки при виконанні математичних операцій. Таким чином, бібліотеки SymPy та NumPy є ефективними інструментами для виконання математичних завдань у мові програмування Python.