Name : Rahma Farag Yehia

Group 2

Question set 1:

Question	Matrix A	Matrix B	Matrix C	Matrix D	Matrix E	Matrix F	Matrix G
a	(2,3)	(4,3)	(1,5)	(2,2)	(1,1)	(4,1)	(3,3)
b	Not square	Not square	Not square	Square	Square	Not Square	Square
c	Not	Not	Not	Not	Symmetric	Not	Symmetric
d	-	Yes	-	-	-	-	-
e	Yes	Yes	-	-	-	-	-
f	-	-	-	-	-	Yes	-
g	-	-	Yes	-	-	-	-
h	$\begin{bmatrix} -1 & 0 \\ 23 & -2 \\ 10 & -11 \end{bmatrix}$	$\begin{bmatrix} -6 & 3 & -5 & 1 \\ 2 & -3 & -11 & 9 \\ 10 & 4 & -1 & 9 \end{bmatrix}$	$\begin{bmatrix} -3 \\ 2 \\ 9 \\ -5 \\ 7 \end{bmatrix}$	-	$[3]$	-	$\begin{bmatrix} -6 & -4 & 23 \\ -4 & -3 & 4 \\ 23 & 4 & 1 \end{bmatrix}$

Questions set 2:

Question	Answer
a	C, E
b	B, C, E
c	A, C, D, E

Questions set 3:

Question	Answer
a) AB	$\begin{bmatrix} (-1*-1)+(1*0)+(-2*-1) & (-1*2)+(1*-3)+(-2*-2) & (-1*0)+(1*4)+(-2*3) \\ (0*-1)+(-2*0)+(1*-2) & (0*2)+(-2*-3)+(1*-2) & (0*0)+(-2*4)+(1*3) \end{bmatrix}$ $= \begin{bmatrix} 3 & -1 & -2 \\ -1 & 4 & -5 \end{bmatrix}$
b) BC	Not possible
c) AD	Not possible
d) EF	Not possible
e) FE	$\begin{bmatrix} (-1*3)+(0*5)+(2*-11) \\ (-2*3)+(-3*5)+(4*-11 \\ (1*3)+(4*5)+(-3*-11 \end{bmatrix}$ $= \begin{bmatrix} -25 \\ -65 \\ 56 \end{bmatrix}$

Question set 4:

Question	Answer
A)	$2x + y = 2 \text{ and } y + 1 = 3$ $\therefore y = 2$ $2x + 2 = 2$ $\therefore x = 0$
B)	$-4x - 4y = -6 \text{ and } x - y = -2 * 13$ $x = \frac{-49}{4} = -12.25 \text{ and } y = \frac{55}{4} = 13.75$
C)	$\begin{bmatrix} 2(x+y) & -(x+y)+4 \\ 2(x-y) & -(x-y)-2 \end{bmatrix} = \begin{bmatrix} 8 & 0 \\ 12 & -8 \end{bmatrix}$ $2(x+y) = 8 \rightarrow x + y = 4$ $2(x-y) = 12 \rightarrow x - y = 6$ $x = 5 \; and \; y = -1$

Question set 5:

i-

Question	Answer
a	$2\mathbf{u} = \begin{bmatrix} 8 \\ 9 \\ 0 \end{bmatrix} \rightarrow \mathbf{u} = \begin{bmatrix} 4 \\ 4.5 \\ 0 \end{bmatrix}$
b	$\boldsymbol{u} = -3\begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix} + 2\begin{bmatrix} -1 \\ 3 \\ 4 \end{bmatrix} \rightarrow \mathbf{u} = \begin{bmatrix} -5 \\ 0 \\ 11 \end{bmatrix}$
c	$1 * r + 0 * s = 2 \rightarrow r = 2$ $0 * r + 1 * s = -3 \rightarrow s = -3$
d	$1 * r + 2 * s = -1$ $2 * r + 2 * s = 3$ $r = 4 \ and\ s = -2.5$

ii-

$$a * \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + b * \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + c * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

For the vectors u1, u2, u3 to span R3 we need to be able to represent any vector in the space using a linear combination of the three vectors

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$a = x\ , b = y\ , c = z$$

We can find a, b, c for any vector $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$

We can represent any vector $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ **using the three vectors u1, u2, u3**

iii-

$$\alpha * \begin{bmatrix}1\\1\\0\end{bmatrix} + \beta * \begin{bmatrix}0\\1\\2\end{bmatrix} + \begin{bmatrix}3\\1\\-4\end{bmatrix} = 0$$

$$\alpha + 3 = 0 \rightarrow \alpha = -3$$
$$\alpha + \beta + 1 = 0 \rightarrow \beta = 2$$
$$2 * \beta - 4 = 0$$

Z is a linear combination of x and y z = 3x -2y

Iv –

1) $w = v - 4\,\alpha\,u$
2) u^T**w = 0**
3) **||u|| = 5**
4) u^T**v= 3**

in 1) -> u^T**w =** u^T**v – 4** $\alpha * u^T u$

0 = 3 – 4 α***** $||u||^2$

0 = 3 – 4 α *** 25**

α **= 0.03**

v –

For u and v to be orthogonal u.v = $u^T v$ **= 0**

$$[\sin(\theta) \quad \cos(\theta)]\begin{bmatrix}-1\\1\end{bmatrix} = 0$$
$$-\sin(\theta) + \cos(\theta) = 0$$
$$\tan(\theta) = 1$$
$$\theta = \frac{\pi}{4} \pm 2\,n\,\pi$$

vi –

a-

The three vectors are orthogonal if v and w are both perpendicular on u and on each other

1) $u.v = \begin{bmatrix} \boldsymbol{k} & \boldsymbol{0} & \boldsymbol{2k} \end{bmatrix} \begin{bmatrix} \boldsymbol{4L} \\ \boldsymbol{1} \\ \boldsymbol{-2L} \end{bmatrix} = \boldsymbol{4lK - 4lK = 0} \;\therefore \boldsymbol{u\ and\ v\ are\ orthogonal}$

2) $u.w = \begin{bmatrix} \boldsymbol{k} & \boldsymbol{0} & \boldsymbol{2k} \end{bmatrix} \begin{bmatrix} \boldsymbol{-2M} \\ \boldsymbol{10M} \\ \boldsymbol{M} \end{bmatrix} = \boldsymbol{-2MK + 2MK = 0} \;\therefore \boldsymbol{u\ and\ w\ are\ orthogonal}$

3) $\mathrm{w}.v = \begin{bmatrix} \boldsymbol{-2M} & \boldsymbol{10M} & \boldsymbol{M} \end{bmatrix} \begin{bmatrix} \boldsymbol{4L} \\ \boldsymbol{1} \\ \boldsymbol{-2L} \end{bmatrix} = \boldsymbol{-8ML + 10M - 2ML = 0\ w\ and\ v\ are\ orthogonal}$

b-

They are orthonormal if they are orthogonal and has a length of 1

$$\therefore k = \frac{1}{\sqrt{1^2 + 2^2}} = \frac{1}{\sqrt{5}}$$

$$\therefore L = \frac{1}{\sqrt{4^2 + 1^2 + 2^2}} = \frac{1}{\sqrt{21}}$$

$$\therefore M = \frac{1}{\sqrt{1^2 + 10^2 + 2^2}} = \frac{1}{\sqrt{105}}$$

Practice with Code:

1. Write a NumPy code line(s) to get and print your numpy library version
   - Import numpy as np
   - np.__version__

```
In [1]: import numpy as np

In [2]: np.__version__

Out[2]: '1.19.5'
```

2. Write a NumPy code line(s) to get help on the “add” function.

   - help(np.add)
   - np.info(np.add)

```
In [4]: help(np.add)

Help on ufunc object:

add = class ufunc(builtins.object)
 |  Functions that operate element by element on whole arrays.
 |
 |  To see the documentation for a specific ufunc, use `info`.  For
 |  example, ``np.info(np.sin)``.  Because ufuncs are written in C
 |  (for speed) and linked into Python with NumPy's ufunc facility,
 |  Python's help() function finds this page whenever help() is called
 |  on a ufunc.
 |
 |  A detailed explanation of ufuncs can be found in the docs for :ref:`ufuncs`.
 |
 |  Calling ufuncs:
 |  ===============
 |
 |  op(*x[, out], where=True, **kwargs)
 |  Apply `op` to the arguments `*x` elementwise, broadcasting the arguments.
 |
```

```
In [7]: np.info(np.add)

add(x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True[, signature, extobj])

Add arguments element-wise.

Parameters
----------
x1, x2 : array_like
    The arrays to be added.
    If ``x1.shape != x2.shape``, they must be broadcastable to a common
    shape (which becomes the shape of the output).
out : ndarray, None, or tuple of ndarray and None, optional
    A location into which the result is stored. If provided, it must have
    a shape that the inputs broadcast to. If not provided or None,
    a freshly-allocated array is returned. A tuple (possible only as a
    keyword argument) must have length equal to the number of outputs.
where : array_like, optional
    This condition is broadcast over the input. At locations where the
    condition is True, the `out` array will be set to the ufunc result.
    Elsewhere, the `out` array will retain its original value.
```

3. Write a NumPy code line(s) to test whether any of the elements of an input array is non-zero

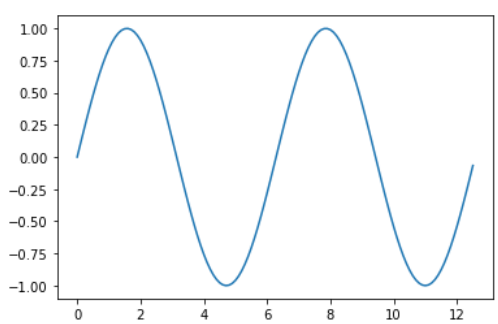
- np.array([1,2,3,0,0])
- np.nonzero(x)

```
In [5]: x = np.array([1, 2, 3, 0, 0])

        np.nonzero(x)

Out[5]: (array([0, 1, 2], dtype=int64),)
```

4. Write a NumPy code line(s) to compute the x and y coordinates for points on a sine curve and plot the points using matplotlib.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 4 * np.pi,0.1)
y = np.sin(x)

plt.plot(x, y)
plt.show()
```

```
In [9]: import numpy as np
        import matplotlib.pyplot as plt

        x = np.arange(0, 4 * np.pi,0.1)
        y = np.sin(x)

        plt.plot(x, y)
        plt.show()
```



5. Write a NumPy code line(s) to extract all numbers which are less and greater than a specified integer in an input array

a = np.array([1,2,3,4,5,6,7,8,9,10])

print("values less than or greater than 5 ",a[a!=5])

print("values less than 5 ",a[a<5])

print("values greater than 5 ",a[a>5])

```
In [15]: a = np.array([1,2,3,4,5,6,7,8,9,10])

         print("values less than or greater than 5 ",a[a!=5])
         print("values less than 5 ",a[a<5])
         print("values greater than 5 ",a[a>5])

         values less than or greater than 5  [ 1  2  3  4  6  7  8  9 10]
         values less than 5  [1 2 3 4]
         values greater than 5  [ 6  7  8  9 10]
```

6. Write a NumPy code line(s) to find the missing (hint: undefined) data in an input array

a = np.array([1,2,3, np.nan, 5, 6])

np.isnan(a) # check for not a number in the array

```
In [23]: a = np.array([1,2,3, np.nan, 5, 6])

         np.isnan(a) # check for not a number in the array

Out[23]: array([False, False, False,  True, False, False])
```