UECM1703 INTRODUCTION TO SCIENTIFIC COMPUTING Dec 2021 Repl Marking Guide

Q1. (a) Given that **A** stores the following matrix

$$\begin{bmatrix} 0.63 & 0.04 & 0.53 & 0.70 & 0.16 & 0.37 & 0.18 \\ 0.04 & 0.42 & 0.53 & 0.26 & 0.75 & 0.44 & 0.92 \\ 0.46 & 0.43 & 0.45 & 0.03 & 0.90 & 0.31 & 0.32 \\ 0.52 & 0.31 & 0.33 & 0.13 & 0.65 & 0.92 & 0.69 \\ 0.48 & 0.42 & 0.37 & 0.75 & 0.79 & 0.99 & 0.60 \\ 0.30 & 0.29 & 0.12 & 0.58 & 0.33 & 0.89 & 0.58 \\ 0.39 & 1.00 & 0.95 & 0.73 & 0.89 & 0.94 & 0.04 \\ 0.42 & 0.23 & 0.55 & 0.54 & 0.59 & 0.56 & 0.31 \end{bmatrix}.$$

(i) Write the Python command to pick all the row 7, row 5, row 3 and row 1 of **A** and write down the output of your command. (1 mark)

Ans. `print(A[6::-2])` .... [0.8 mark]

```
[[0.39 1.   0.95 0.73 0.89 0.94 0.04]
 [0.48 0.42 0.37 0.75 0.79 0.99 0.6 ]
 [0.46 0.43 0.45 0.03 0.9  0.31 0.32]
 [0.63 0.04 0.53 0.7  0.16 0.37 0.18]]
```

.... [0.2 mark]

(ii) Write down the single line Python command which picks the row 5, row 3, row 1, row 3, row 5 and column 6, column 4, column 2 and column 3 (should be 4, not 3) and column 5 (should be column 6, not 5) from **A** so that the output is:

```
[[0.99 0.75 0.42 0.75 0.99]
 [0.31 0.03 0.43 0.03 0.31]
 [0.37 0.7  0.04 0.7  0.37]
 [0.31 0.03 0.43 0.03 0.31]
 [0.99 0.75 0.42 0.75 0.99]]
```

(1 mark)

Ans. Following the instruction:

`print(A[[4,2,0,2,4],:][:, [5,3,1,2,4]])`

Following the output:

`print(A[[4,2,0,2,4],:][:, [5,3,1,3,5]])` .... [1 mark]

(iii) Write down the Python command that would find all the row standard deviation of the matrix **A**. (0.5 mark)

Ans. `print(A.std(1))` .... [0.5 mark]

(iv) Write down the Python command that would subtract each **row** of the matrix **A** by the **row mean** and divide by the standard deviation of each **row**. (0.5 mark)

Ans. `((A.T - A.mean(1))/A.std(1)).T` .... [0.5 mark]

(b) Write down the command which generates the following Toeplitz matrix by importing the appropriate special function:

$$A = \begin{bmatrix} -1 & -2 & 1 & 1 & -1 \\ -1 & -1 & -2 & 1 & 1 \\ -1 & -1 & -1 & -2 & 1 \\ 5 & -1 & -1 & -1 & -2 \\ 0 & 5 & -1 & -1 & -1 \end{bmatrix}$$ [1 mark]

Ans. Appropriate input and the command .... [1 mark]

```
from scipy import linalg
print(linalg.toeplitz([-1,-1,-1,5,0],[-1,-2,1,1,-1]))
```

(c) Given the linear algebra problem below

$$
\begin{aligned}
12x_1 - 3x_2 &= 96\\
-3x_1 + 51x_2 - 4x_3 &= 603\\
-8x_2 + 40x_3 - 3x_4 &= -366\\
-7x_3 + 19x_4 - x_5 &= 216\\
-4x_4 + 63x_5 - 2x_6 &= 988\\
-x_5 + 98x_6 - 4x_7 &= -1048\\
-2x_6 + 23x_7 - 8x_8 &= 223\\
-x_7 + 67x_8 &= 791
\end{aligned}
$$

The problem can be expressed in the matrix form

$$A\boldsymbol{x} = \boldsymbol{b}$$

where A is 8×8, $\boldsymbol{x}$ is $[x_1, \ldots, x_8]^T$ and $\boldsymbol{b}$ is 8×1.

(i) Write down the commands to construct A and $\boldsymbol{b}$ in no direct entry of all values, no more than 4 commands and no loops. (2 marks)

Ans. One possibility is to use the indexing method.

```
A = np.diag([12, 51, 40, 19, 63, 98, 23, 67])
A[np.r_[0:7],np.r_[1:8]] = [-3,-4,-3,-1,-2,-4,-8]
A[np.r_[1:8],np.r_[0:7]] = [-3,-8,-7,-4,-1,-2,-1]
b = np.array([96, 603, -366, 216, 988, -1048, 223, 791])
```

The other is to use np.diag and elementwise addition. . . . . [2 marks]

(ii) Write down the command to take the diagonal of the matrix A without using loop. (0.5 mark)

Ans. A.diagonal() . . . . [0.5 mark]

(iii) Write down the command to obtain the matrix B in which $A - B$ is the diagonal matrix with the diagonal of matrix A. (0.5 mark)

Ans. `B = A - np.diag(A.diagonal())` . . . . [0.5 mark]

(iv) Write down the commands which solves the linear algebra problem by importing the appropriate solver as well as writing down the solution. (1 mark)

Ans. The problem can be solved using the solver from scipy.linalg as follows.

```
from scipy import linalg
x = linalg.solve(A, b)
print("x=", x)
```

. . . . [0.8 mark]

[11. 12. -6. 10. 16. -10. 13. 12.] . . . . [0.2 mark]

(d) Given a 2×2 matrix

$$A = \begin{bmatrix} 1 & -0.1 \\ 0.1 & 1 \end{bmatrix}.$$

Let X be a 2×2 matrix with entries x_{ij}, $i, j = 1, 2$. You are investigating the difference between the matrix exponential function

$$\exp^{[m]}(X) = I_2 + X + \frac{1}{2!}X^2 + \frac{1}{3!}X^3 + \frac{1}{4!}X^4 + \cdots + \frac{1}{k!}X^k + \cdots$$

and the elementwise exponential function

$$\exp(X) = \begin{bmatrix} e^{x_{11}} & e^{x_{12}} \\ e^{x_{21}} & e^{x_{22}} \end{bmatrix}.$$

(i) Write down the Python commands for calculating $\exp^{[m]}(A)$ and $\exp(A)$. Run the Python commands and write down the output of the commands. Then, write down the difference $\exp^{[m]}(A) - \exp(A)$. (1 mark)

Ans. The Python commands are respectively

- $\exp^{[m]}(A)$: linalg.expm(A) . . . . [0.2 mark]
- $\exp(A)$: np.exp(A) . . . . [0.2 mark]

The outputs are respectively . . . . [0.2+0.2=0.4 mark]

```
[[ 2.70470174 -0.27137536]
 [ 0.27137536  2.70470174]]
[[2.71828183 0.90483742]
 [1.10517092 2.71828183]]
```

and the difference is . . . . [0.2 mark]

```
[[-0.01358009 -1.17621278]
 [-0.83379556 -0.01358009]]
```

(ii) Write down the Python command to find the difference

$$\exp^{[m]}(A) - I_2 - A - \frac{1}{2!}A^2 - \frac{1}{3!}A^3$$

and write down the difference. (1 mark)

Ans. The Python command to find the difference is . . . . [0.8 mark]

```
linalg.expm(A) - np.eye(2) - A - 0.5*A@A - 1/6*A@A@A
```

and the output is . . . . [0.2 mark]

```
[[ 0.04803508 -0.02154203]
 [ 0.02154203  0.04803508]]
```

[Total : 10 marks]

Q2. (a) Write a Python script that finds the intersection points of the quadratic curve $y = ax^2 + bx + c$ with the line $y = mx + d$ and meets the following requirements.

- Show that the purpose of the program;
- Prompt the user for the coefficients a, b and c of the quadratic curve $y = ax^2 + bx + c$;
- Prompt the user for the coefficients m and d of the line $y = mx + d$;
- Detect the special case when $a = 0$ and tries to find and print out the intersection point;
- When $a \neq 0$, print out the intersections points by using the discriminant $B^2 - 4AC$ to detect if there is at least an intersection ($B^2 - 4AC \geq 0$) or print “is empty” if there is no intersection ($B^2 - 4AC < 0$)

Your Python script should generate the following output which finds the intersection between $y = x^2 + 4x + 5$ and $y = 3x + 7$.

```
The purpose of this program is to find intersection
points of the quadratic curve y = ax^2 + bx + c with
the line y = mx + d.

Enter the curve's coefficient a: 1
Enter the curve's coefficient b: 4
Enter the curve's coefficient c: 5
Enter the line's  coefficient m: 3
Enter the line's  coefficient d: 7

The intersection(s) of the quadratic curve

    y = 1.0x^2 + 4.0x + 5.0

and the line

    y = 3.0x + 7.0

are (1.0, 10.0) and (-2.0, 1.0).
```

In addition, your Python script should generate the following output which tries to find the intersection between $y = x^2 + 1$ and $y = x$.

```
The purpose of this program is to find intersection
points of the quadratic curve y = ax^2 + bx + c with
the line y = mx + d.

Enter the curve's coefficient a: 1
Enter the curve's coefficient b: 0
Enter the curve's coefficient c: 1
Enter the line's  coefficient m: 1
Enter the line's  coefficient d: 0

The intersection(s) of the quadratic curve

    y = 1.0x^2 + 0.0x + 1.0

and the line

    y = 1.0x + 0.0

is empty.
```

(5 marks)

Ans.

```
print("""
The purpose of this program is to find intersection
points of the quadratic curve y = ax^2 + bx + c with
the line y = mx + d.
""")
# ........................................ [1 mark]

a = float(input("Enter the curve's coefficient a: "))
b = float(input("Enter the curve's coefficient b: "))
c = float(input("Enter the curve's coefficient c: "))
m = float(input("Enter the line's  coefficient m: "))
d = float(input("Enter the line's  coefficient d: "))
# ........................................ [1 mark]

print("""
The intersection(s) of the quadratic curve

    y = {a}x^2 + {b}x + {c}

and the line

    y = {m}x + {d}
""".format(a=a,b=b,c=c,m=m,d=d))
# ........................................ [1 mark]

A = a
B = b - m
C = c - d
# Using np.root([A,B,C]) won't work because one needs to
# check for real roots!
from math import sqrt
if A == 0:
    if B != 0:
        x1 = -C/B
        y1 = m*x1 + d
        print("is ({x1}, {y1})")
    else:
        print("is empty.")
else:
    if B**2-4*A*C >= 0:
        x1 = (-B+sqrt(B**2-4*A*C))/2/A
        y1 = m*x1 + d
        x2 = (-B-sqrt(B**2-4*A*C))/2/A
        y2 = m*x2 + d
        print("are ({x1}, {y1}) and ({x2}, {y2})."
          .format(x1=x1,y1=y1,x2=x2,y2=y2))
    else:
        print("is empty.")
# ........................................ [2 marks]
```

[Total : 5+5 = 10 marks]