UECM1703 Introduction to Scientific Computing Oct 2020 Marking Guide

Q1. (a) Write down the four basic data types in Python and explain how one of them is different from the basic type using in Numpy module. (2 marks)

Ans. Boolean, Integer, Floating point numbers, Strings . . . . [0.3×4=1.2 mark]

Python integers are supports arbitrary large integers while Numpy only allow integers of maximum 64 bits. . . . . [0.8 mark]

(b) Write down the single line Python command to generate the following vectors (1-D Numpy arrays). In each of the answer below, if your Python command is regarded as a Python string, its length cannot be more than 30. Otherwise, marks will be deducted.

(i) Arithmetic sequence: [2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50] (0.5 mark)

Ans. `np.arange(2,51,3)` . . . . [0.5 mark]

(ii) Geometric sequence:

```
array([1.5       , 0.75      , 0.375     , 0.1875    ,
       0.09375   , 0.046875, 0.0234375, 0.01171875,
       0.00585938, 0.00292969])
```

(1 mark)

Ans. `3*2.0**np.arange(-1,-11,-1)` . . . . [1 mark]

(c) By using Python, **write a function** `catalan(n)` that allows you to generate a the nth Catalan number (`https://en.wikipedia.org/wiki/Catalan_number`):

$$C_n = \prod_{k=2}^{n} \frac{n+k}{k}, \quad n \geq 2; \quad C_0 = C_1 = 1.$$

Write down the Python command which allows you to generate the array of Catalan numbers $\{C_0, C_1, C_2, \cdots, C_{10}\}$. (2.5 marks)

Ans. A sample implemention of `catalan(n)` and the array of Catalan numbers can be generated as follows.

```
def catalan(n):
    num = 1
    den = 1
    for k in range(2, n+1):
        num *= (n+k)
        den *= k
    return num//den                               #[2  marks]

import numpy as np
print(np.array([catalan(n) for n in range(11)]))#[0.5 mark ]
```

(d) A function f is defined by

$$f(x) = \cos(\pi x) + 0.9\cos(7\pi x) + 0.9^2\cos(7^2\pi x) + \cdots + 0.9^{10}\cos(7^{10}\pi x)$$
$$= \sum_{n=0}^{10} 0.9^n \cos(7^n \pi x).$$

Plot the function f for the range $[-\pi, \pi]$ with 1001 equally distributed points on the x-axis. Show your Python code to plot the graph of f as well as inserting the graph of f into your answer script. (2.5 marks)

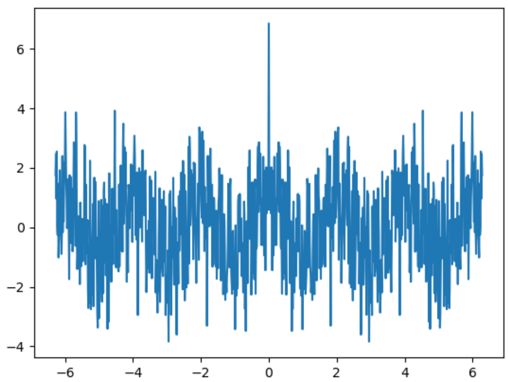
Ans. A possible sequence of Python commands to generate the plot is shown below.

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

a, b = 0.9, 7
x = np.linspace(-2*np.pi,2*np.pi,1000+1)
y = sum(a**n * np.cos(b**n * np.pi * x) for n in range(11))
plt.plot(x, y)
plt.savefig("weierstrass2.png")
plt.show()
```

. . . . [2 marks]

The plot is shown below. . . . . [0.5 mark]



(e) Given the function $g(x) = x^2(4-x)^3$.

(i) Define the function g in Python. (0.5 mark)
Ans. def g(x): return x**2*(4.0-x)**3 . . . . [0.5 mark]

(ii) Use an appropriate function from Scipy to find $\int_0^4 g(x)dx$. Write down the Python command and the output. (1 mark)
Ans. scipy.integrate.quad(g, 0, 4) . . . . [0.5 mark]
(68.26666666666667, 7.579122514774402e-13) . . . . [0.5 mark]

(iii) Use either a brute force method or Scipy to **find the** x in which the function $g(x)$ is **maximum value** for the range $[0,4]$. (1 mark)
Ans. x=np.linspace(0,4,100+1); x[np.argmax(g(x))] # x=1.6 . . . . [1 mark]
Alternative: scipy.optimize.fmin(lambda x: -g(x), 0)

(f) Use Numpy array to generate the following $n \times 2n$ matrix:

$$\begin{bmatrix} 0&0&0&0&0&0&1&1&0&0&0&0&0&0\\ 0&0&0&0&0&1&1&1&1&0&0&0&0&0\\ 0&0&0&0&1&1&1&1&1&1&0&0&0&0\\ 0&0&0&1&1&1&1&1&1&1&1&0&0&0\\ 0&0&1&1&1&1&1&1&1&1&1&1&0&0\\ 0&1&1&1&1&1&1&1&1&1&1&1&1&0\\ 1&1&1&1&1&1&1&1&1&1&1&1&1&1 \end{bmatrix}$$

Note that this matrix is 7×14 but you must write a Python program which allow you to generate similar matrix of any size $n \times 2n$. (2 marks)

Ans. A sample implemention is given below. Other equivalent methods will also receive marks. [2 marks]

```
import numpy as np
n = 7
A = np.zeros((n,2*n),dtype=np.int64)
for i in range(n):
    A[i,(n-i-1):(n+i+1)] = 1
print(A)
```

(g) You are trying to use a polynomial $y = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$ to fit the following 2D data points:

$$(145,7),\ (155,17),\ (165,32),\ (175,51),\ (180,60).$$

This will lead to the following system of linear equations:

$$\begin{aligned} a_0 + 145a_1 + 145^2a_2 + 145^3a_3 + 145^4a_4 &= 7 \\ a_0 + 155a_1 + 155^2a_2 + 155^3a_3 + 155^4a_4 &= 17 \\ a_0 + 165a_1 + 165^2a_2 + 165^3a_3 + 165^4a_4 &= 32 \\ a_0 + 175a_1 + 175^2a_2 + 175^3a_3 + 175^4a_4 &= 51 \\ a_0 + 185a_1 + 185^2a_2 + 185^3a_3 + 185^4a_4 &= 60 \end{aligned}$$

which can be transformed into a matrix form:

$$A\boldsymbol{a} = \begin{bmatrix} 1 & 145 & 145^2 & 145^3 & 145^4 \\ 1 & 155 & 155^2 & 155^3 & 155^4 \\ 1 & 165 & 165^2 & 165^3 & 165^4 \\ 1 & 175 & 175^2 & 175^3 & 175^4 \\ 1 & 185 & 185^2 & 185^3 & 185^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 7 \\ 17 \\ 32 \\ 51 \\ 60 \end{bmatrix} = \boldsymbol{b}.$$

where A is the 5×5 matrix and $\boldsymbol{b}$ is the y-values.

(i) Construct the matrix A using **for loop(s)**. (2.5 marks)

Ans. A sample implementation is shown below. An implementation without using for loops will receive mark deduction.

```
# Vandermonde matrix
import numpy as np
#A = np.vander([145.0, 155, 165, 175, 185])[:,-1::-1]
N = 5
A = np.ones((N,N))
cs = [145., 155., 165., 175., 185.]
b = [7,17,32,51,60]
for i in range(N):
    for j in range(1,N):
        A[i,j] = cs[i]**j
print(A)
```

- Appropriate imports . . . . [0.5 mark]
- Appropriate initialisation . . . . [1 mark]
- Correct for loop . . . . [1 mark]

(ii) Find the determinant of the matrix A by writing down both the Python command and the result. (1 mark)

Ans. np.linalg.det(A) . . . . [0.5 mark]

2879999999999.994 . . . . [0.5 mark]

(iii) Solve $A\boldsymbol{a} = \boldsymbol{b}$ where $\boldsymbol{a}$ is the vector of the unknown coefficients a_0, a_1, a_2, a_3, a_4. (1 mark)

Ans. np.linalg.solve(A,[7,17,32,51,60]) . . . . [0.5 mark]

$a_0 = -3.41103672e+04$, $a_1 = 8.64637500e+02$, $a_2 = -8.20395833e+00$, $a_3 = 3.45000000e-02$, $a_4 = -5.41666667e-05$ . . . . [0.5 mark]

(iv) Identify the problem of using the polynomial $y = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$ to fit the 2D data points and propose a solution to solve the problem you state. [Hint: The answer provided must be relevant to scientific computing.] (0.5 mark)

Ans. The determinant of the matrix is too large. . . . . [0.2 mark]

A possible solution: Use $y = a_0 + a_1(x-\bar{x}) + a_2(x-\bar{x})^2 + a_3(x-\bar{x})^3 + a_4(x-\bar{x})^4$ and it is possible to have a matrix with smaller determinant. . . . [0.3 mark]

(h) Given the following 3×3 matrices of integers:

$$A = \begin{bmatrix} 1 & -1 & 6 \\ 7 & -7 & -2 \\ 8 & 9 & 5 \end{bmatrix}, \quad B = \begin{bmatrix} -6 & -7 & -4 \\ -8 & -7 & -2 \\ 8 & 7 & -5 \end{bmatrix} C = \begin{bmatrix} -5 & -6 & -5 \\ 5 & 7 & -1 \\ 2 & -3 & 1 \end{bmatrix}$$

(i) Write down a single command to form the following 6×6 matrix from A, B and C:

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

(0.8 mark)

Ans. `D = np.vstack((np.hstack((A,B)),np.hstack((B,C))))` [0.8 mark]

Remark: I learn "`D = np.bmat([[A,B],[B,C]])`" from student.

For example, if I want to stack B to the "back" of A, then

```
np.stack([A.T,B.T],axis=2).T
```

(ii) Write down a single line of Python command to count the number of integers which are odd, and the output of the command. (0.7 mark)

Ans. `np.sum(D % 2 == 1)` $\Rightarrow$ 21 ....[0.5+0.2=0.7 mark]

(iii) Write down a single line of Python command to change all the negative entries in matrix D to 0. (0.5 mark)

Ans. `D[D<0]=0` .... [0.5 mark]