21-671 A Homework Assignment

1(a) Given, $f(n) = \log(n!) + n2^n + 5n^4$

Taking each part of $f(n)$ separately so as to find the value of big-O more easily.

$f_1(n) = \log(n!)$

$f_2(n) = n2^n$

$f_3(n) = 5n^4$

Taking $f_1(n)$

$f_1(n) = \log(n!)$ ... ①

We know

$n! = n(n-1)(n-2)\ldots 3.2.1$

which can be compared similarly to the expression

$\approx \underbrace{n.n.n \ldots n.n.n}_{n \text{ times}}$

$\therefore O(n!) \approx n^n$ ... ②

Plugging the value of ② in ① we get

$f_1(n) = \log(n^n)$

$\therefore f_1(n) = O(n \log(n)) \ldots ③$

Taking $f_2(n)$

$f_2(n) = n\,2^n$

We can take $f_2(n)$ as a function made up of two components

We know n grows linearly we also know that comparing the rate of growth of both components that the rate of growth of $2^n > n$

$\therefore O(f_2(n)) = O(2^n) \ldots ④$

Taking $f_3(n)$

$f_3(n) = 5n^4$

$O(f_3(n)) = O(n^4) \ldots ⑤$

Comparing the values of all three parts will give us the value of the big-O of the given function

From comparison we know that

$$n \log n < n^4 < 2^n$$

$\therefore$ The value of big-O for the given equation is $O(2^n)$

(b) Given,

$$f(n) = n\cos(n^3) + n^2 + n + 1$$

Taking and splitting the function we get

$$f_1(n) = n\cos(n^3)$$

$$f_2(n) = n^2$$

$$f_3(n) = n$$

$$f_4(n) = 1$$

Taking $f_1(n)$,

$$f_1(n) = n\cos(n^3)$$

We know that the value of anything inside a cosine function only ranges between $[-1, 1]$

So we can take the value $\cos(n^3)$ as 1

$$\therefore O(f_1(n)) = O(n.1)$$
$$= O(n)$$

Taking $f_2(n)$

$$f_2(n) = n^2$$

$$\therefore O(f_2(n)) = O(n^2)$$

Taking $f_3(n)$

$f_3(n) = n$

$\therefore O(f_3(n)) = O(n)$

Taking $f_4(n)$

$f_4(n) = 1$.

$\therefore O(f_4(n) = O(1)$

Comparing all the values produced by each seperation we get

$$O(1) < O(n) = O(n) < O(n^2)$$

$\therefore O(f(n)) = O(n^2)$

(c) Given,

$$f(n) = \frac{1}{100} n^3 + 100n^2 + 1000n$$

Taking it as three seperate parts we get

$$f_1(n) = \frac{1}{100} n^3$$

$$f_2(n) = 100n^2$$

$$f_3(n) = 1000n$$

Finding the big O of $f_1(n)$

$$f_1(n) = \frac{1}{100} n^3$$

$$O(f_1(n)) = O(n^3)$$

Finding the big O of $f_2(n)$

$$f_2(n) = 100n^2$$

$$O(f_2(n)) = O(n^2)$$

Finding the big O of $f_3(n)$

$$f_3(n) = 1000n$$

$$O(f_3(n)) = O(n)$$

Comparing the values of big-O received from each part of the function we get

$$O(n) < O(n^2) < O(n^3)$$

∴ The big-O of the given function is $O(n^3)$

2 Given

$$y = f(x) = e^x$$

$$\hat{f}(x) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$$

To find the actual, forward and backward error when $x = 1$ and $x = 10$

1] When $x = 1$

i] Actual value

$$y = f(1) = e^1 \approx 2.71828$$

ii] Approximate value

$$\hat{y} = f(\hat{x}) = \hat{f}(x)$$

$$= \hat{f}(1)$$

$$\hat{y} = 1 + 1 + \frac{1}{2} + \frac{1}{6}$$

$$\hat{y} = 2 + \frac{3+1}{6}$$

$$\hat{y} = 2 + \frac{4}{6}$$

$$\hat{y} \approx 2.66667$$

iii] Forward error

$$\Delta y = |\hat{y} - y|$$

$$= |2.66667 - 2.71828|$$
$$= |-0.05161|$$
$$= 0.05161$$

iv] Backward error

$$\Delta x = |\hat{x} - x|$$

$$\hat{y} = f(\hat{x})$$

$$\therefore \hat{x} = f^{-1}(y)$$

$$\hat{x} = \log_e (2.66667)$$

$$\hat{x} \approx 0.98083$$

$$\Delta x = |0.98083 - 1|$$
$$\Delta x = |-0.01917|$$
$$\therefore \Delta x = 0.01917$$

2] When $x = 10$

i] Actual value

$$y = f(x) = e^{10} = 22026.4658$$

ii] Approximate value

$$\hat{y} = f(\hat{x}) = \hat{f}(x) = 1 + 10 + \frac{100}{2} + \frac{1000}{6}$$

$$\hat{y} = 11 + 50. + \frac{500}{3}$$

$$\hat{y} = 61 + 166.6667$$

$$\hat{y} = 227.6667$$

iii] Forward error

$$\Delta y = |\hat{y} - y|$$

$$\Delta y = |227.6667 - 22026.4658|$$

$$\Delta y = |-21,798.7991|$$

$$\Delta y = 21,798.7991$$

iv] Backward error

$$\Delta x = |\hat{x} - x|$$

$$\hat{x} = f^{-1}(\hat{y})$$

$$\hat{x} = \log_e (227.6667)$$

$$\hat{x} \approx 5.4279$$

$$\Delta x = |\hat{x} - x|$$
$$= |5.4279 - 101|$$
$$\Delta x = |-4.5721|$$
$$\Delta x = 4.5721$$

In class we saw that the condition number for the function $f(x) = e^x$ was $|x|$

From the values received after utilising $|\tilde{f}(x)|$ I can see that forward error increases by a factor much larger than x. So it works slightly more effectively for smaller number but that isn't true for larger numbers

(3) To find the conditional number the formula is

$$\text{Conditional number} = \left| \frac{x f'(x)}{f(x)} \right|$$

Where $f'(x)$ is the differentiated form of $f(x)$

Given $f(x) = \cos(x^2) + (1+2x)^3$

$$f'(x) = \frac{d f(x)}{dx}$$

$$= \frac{d(\cos(x^2))}{dx} + \frac{d(1+2x)^3}{dx}$$

$$= -\sin(x^2) \frac{d(x^2)}{dx} + 3(1+2x)^2 \frac{d}{dx}(1+2x)$$

$$f'(x) = -2x \sin(x^2) + 6(1+2x)^2$$

Finding out the formula of the condition matrix

$$\text{condition number} = \left| \frac{x[-2x\sin(x^2) + 6(1+2x)^2]}{\cos(x^2) + (1+2x)^3} \right|$$

$$\text{condition number} = \left| \frac{-2x^2\sin(x^2) + 6x(1+2x)^2}{\cos(x^2) + (1+2x)^3} \right|$$

Estimating the value of the condition number when $x = 10$

$$\text{condition number} = \left| \frac{-2(10)^2(\sin(10^2)) + 6(10)(1+2(10))^2}{\cos(10^2) + (1+2(10))^3} \right|$$

$$\text{condition number} = \left| \frac{-2(100)\sin(100) + 60(1+20)^2}{\cos(100) + (1+20)^3} \right|$$

$$\text{condition number} = \left| \frac{-200(-0.5064) + 60(21)^2}{0.8623 + 21^3} \right|$$

$$\text{condition number} = \left| \frac{101.28 + 60(441)}{0.8623 + 9261} \right|$$

$$\text{condition number} = \left| \frac{101.28 + 26,460}{9261.8623} \right|$$

$$\text{condition number} = \left| \frac{26561.28}{9261.8623} \right|$$

Condition number = 2.8678

The condition number when $x = 10$ is 2.8678

Question 4:

Code:

```
'''import numpy as np
A = np.array([[25, 30, 45, 60], [43, 44, 12, 32]])
B = np.array([[12, 21, 32], [42,34, 53], [78,19, 90], [93,37,89]])
def multiply(A:np.array, B:np.array):
    if np.shape(A)[1] != np.shape(B)[0]:
        return "The matrices are not able to be multiplied"
    C = np.zeros([np.shape(A)[0], np.shape(B)[1]])
    for i in range(np.shape(A)[0]):
        for j in range(np.shape(B)[1]):
            for k in range(np.shape(A)[1]):
                C[i][j] += A[i][k]*B[k][j]
    return C
C=np.matmul(A, B)
print(multiply(A,B))
print(multiply(B, A))
def test(A:np.array, B:np.array, C:np.array):
    if np.array_equal(C, multiply(A,B)):
        print("Both Matrices are the same")
D = np.array([[12, 32, 41], [43, 54, 12], [12, 32, 45]])
E = np.array([[1, 3, 4], [4, 2, 5], [9, 4, 12]])
F = np.matmul(D, E)
test(A, B, C)
test(D, E, F)
'''
```

Output of the code: