# EM314 - NUMERICAL METHODS ASSIGNMENT - 1

Hisni Mohammed M.H.

E/15/131

Theory

Q1. use taylor series $\sin x$,

$$f(x) = f(a) + (x-a) f'(a) + \frac{(x-a)^2}{2!} f''(a) + \frac{(x-a)^3}{3!} f'''(a) + \dots$$

$f(x) = \sin x$.

$a = 0$

$$\therefore \sin x = 0 + x \cdot \cos(0) + \frac{x^2}{2!} \cdot 0 + \frac{x^3}{3!}(-\cos 0) + \dots$$

$$\sin x = 0 + x + 0 + E_3(f, \xi)$$

$$|\sin x - x| = |E_3(f, \xi)|$$

$$= \left| \frac{\cos(\xi) \cdot x^3}{3!} \right|$$

if approximation has to give correct result rounded to six decimal place then.

$$\text{error} < 10^{-6}$$

$$\left| \frac{\cos(\xi) \cdot x^3}{3!} \right| < 10^{-6}$$

$$|\cos \xi \cdot x^3| < 10^{-6} \times 6$$

$$|x^3| < 6 \times 10^{-6} \qquad \because |\cos \xi| < 1$$

$$x < 1.8171 \times 10^{-2}$$

$$x < 0.018171$$

Q2. any $x \in \mathbb{F}$ can be represented as,

$$x = \pm(.d_1 \ldots d_t) \cdot \beta^E$$

the sign bit can only assume 2 values.
$d_1$ cannot assume 0. therefore it can only assume $\beta-1$ values. but each of digits $d_2, d_3, \ldots d_t$ can assume $\beta$ different values. therefore mantissa assumes $(\beta-1) \times \beta^{t-1}$ values.
Range of exponent is $[L, U]$ therefore exponent can assume $(U-L+1)$ values.

$\therefore$ number of elements, $= 2 \times (\beta-1)\beta^{t-1} \times (U-L+1)$

$\therefore$ set $\mathbb{F}$ contains $2(\beta-1)\beta^{t-1}(U-L+1)$ elements.

Q3. Taylor series,

$$f(x) = \sum_{k=0}^{N} \frac{1}{k!} \cdot f^{(k)}(x_0)(x-x_0)^k + E_{N+1}(f, \xi)$$

where $E_{N+1}(f, \xi) = \frac{1}{(N+1)!} \cdot f^{(N+1)}(\xi)(x-x_0)^{N+1}$

$\therefore$ $f(x+h) = f(x) + h f'(x) + \frac{h^2}{2} \cdot f''(\xi)$ where $x < \xi < x+h$

$$f'(x) = \frac{f(x+h) - f(x) - h^2/2 \cdot f''(\xi)}{h}$$

$$= \frac{f(x+h) - f(x)}{h} - \frac{h f''(\xi)}{2} = f_h'(x) - \frac{h \cdot f''(\xi)}{2}$$

$$\therefore \quad E_h(x) = \left| f'(x) - f_h'(x) \right| = \left| h \cdot \frac{f''(\xi)}{2} \right|$$

$$\lim_{h \to 0} \frac{|E_h(x)|}{|h|} = \lim_{h \to 0} \frac{|f''(\xi)|}{2} = \frac{|f''(\xi)|}{2} = \text{constant}.$$

$\therefore$ $E_h = O(h)$

## Computer Experiments

**Q4. <u>OCTAVE code (q4.m)</u>**

**(a) , (b), (c), (d)**

```
n=500:500:5000;
for i=500:500:5000
        A = rand(i);
        st = cputime;
        det(A);
        t(i/500)= cputime - st;
end

N = log(n); T = log(t);
hold on;
plot( N, T, 'ro' );
co = polyfit( N, T, 1 )
y = co(1)*N + co(2);
plot( N, y );
xlabel('logarithm scaled Matrix Size');
ylabel('logarithm scaled Computation Time');
```

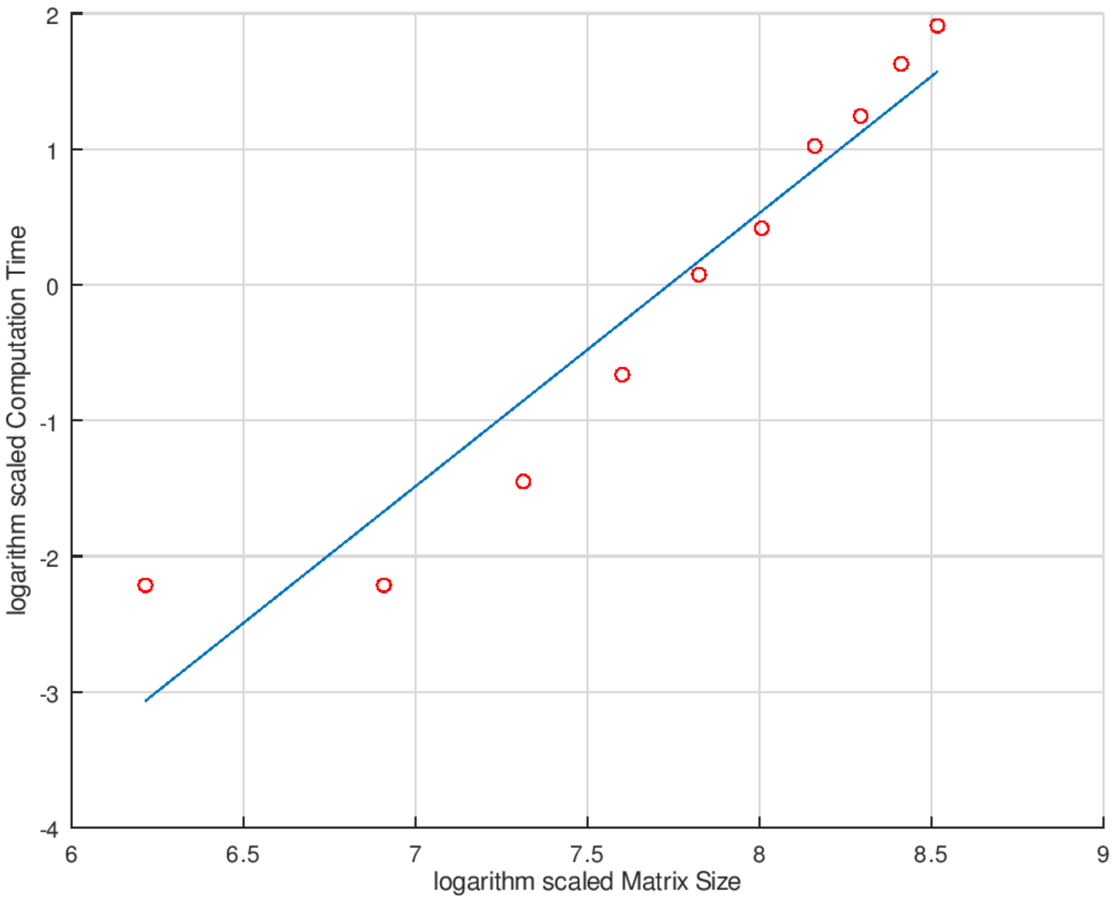
**<u>Output of the code</u>**

```
>> q4

co =

        2.3552  -18.2233
```

**Output Graph ( log t vs log n )**




**(e)** Hence here α = 2.3552

**(f)** Theoretical value > computing value

The theoretical value was 3. Here alpha is 2. 3552. It's because the time taken for all arithmetic was assumed to be constant. But real computation time will vary.

**Q5.** When n = 10

**OCTAVE code (q5.m)**

```
N = 10;
k = 1:N;
hk = 1./(2.^k);
x = 3;

dfhk = ( log(x+hk) - log(x) ) ./ hk;
Ehk = abs( (1/x) - dfhk );

fprintf('k\t hk\t\t f\''hk\t\t Ehk\n' )
for i = 1:N
        fprintf('%d\t %f\t %f\t %ld\n',k(i),hk(i),dfhk(i),Ehk(i))
end

hold on;
loglog(hk,Ehk);
co = polyfit( log(hk), log(Ehk), 1 )
xlabel('logarithm scaled hk value');
ylabel('logarithm scaled Ehk value');
```

**(a) Output of the code**

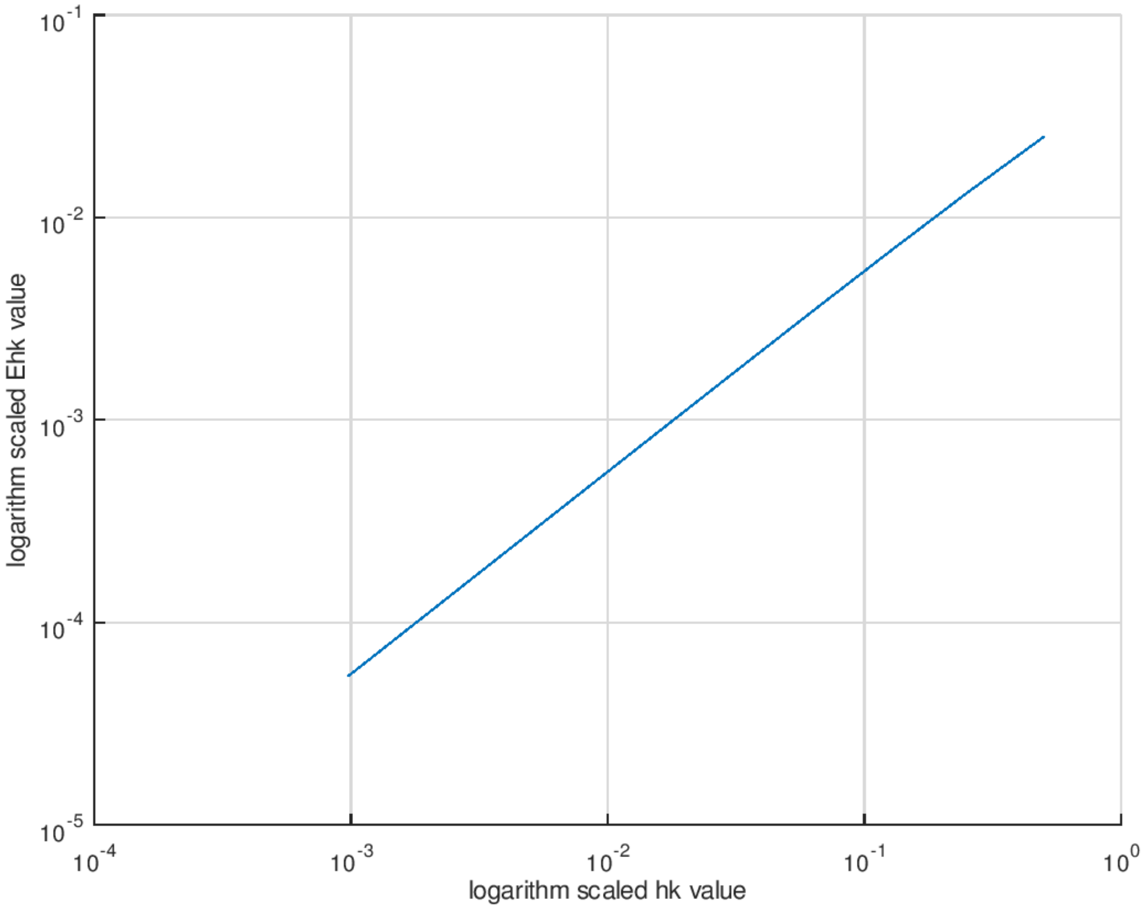
```
>> q5

k     hk          f'hk        Ehk
1     0.500000    0.308301    0.025032
2     0.250000    0.320171    0.0131625
3     0.125000    0.326576    0.00675738
4     0.062500    0.329909    0.00342474
5     0.031250    0.331609    0.00172415
6     0.015625    0.332468    0.000865053
7     0.007812    0.332900    0.000433276
8     0.003906    0.333117    0.000216826
9     0.001953    0.333225    0.00010846
10    0.000977    0.333279    5.42417e-05
```

co =

0.98706  -2.96094

**Output Graph ( log Ehk vs log hk )**




**(b)** Ehk tends to zero (Ehk → 0)

Hence according to the graph ϒ = 0. 98706

Eh/h is tents to a constant while h tents to infinity. Hence Eh = O(h)

**(c)** When n = 40

**Mathlab code (q6.m)**

```
N = 40;
k = 1:N;
hk = 1./(2.^k);
x = 3;

dfhk = ( log(x+hk) - log(x) ) ./ hk;
Ehk = abs( (1/x) - dfhk );

fprintf('k\t hk\t\t f\''hk\t\t Ehk\n' )
for i = 1:N
        fprintf('%d\t %f\t %f\t %ld\n',k(i),hk(i),dfhk(i),Ehk(i))
end

hold on;
loglog(hk,Ehk,'b');
xlabel('logarithm scaled hk value');
ylabel('logarithm scaled Ehk value');
```

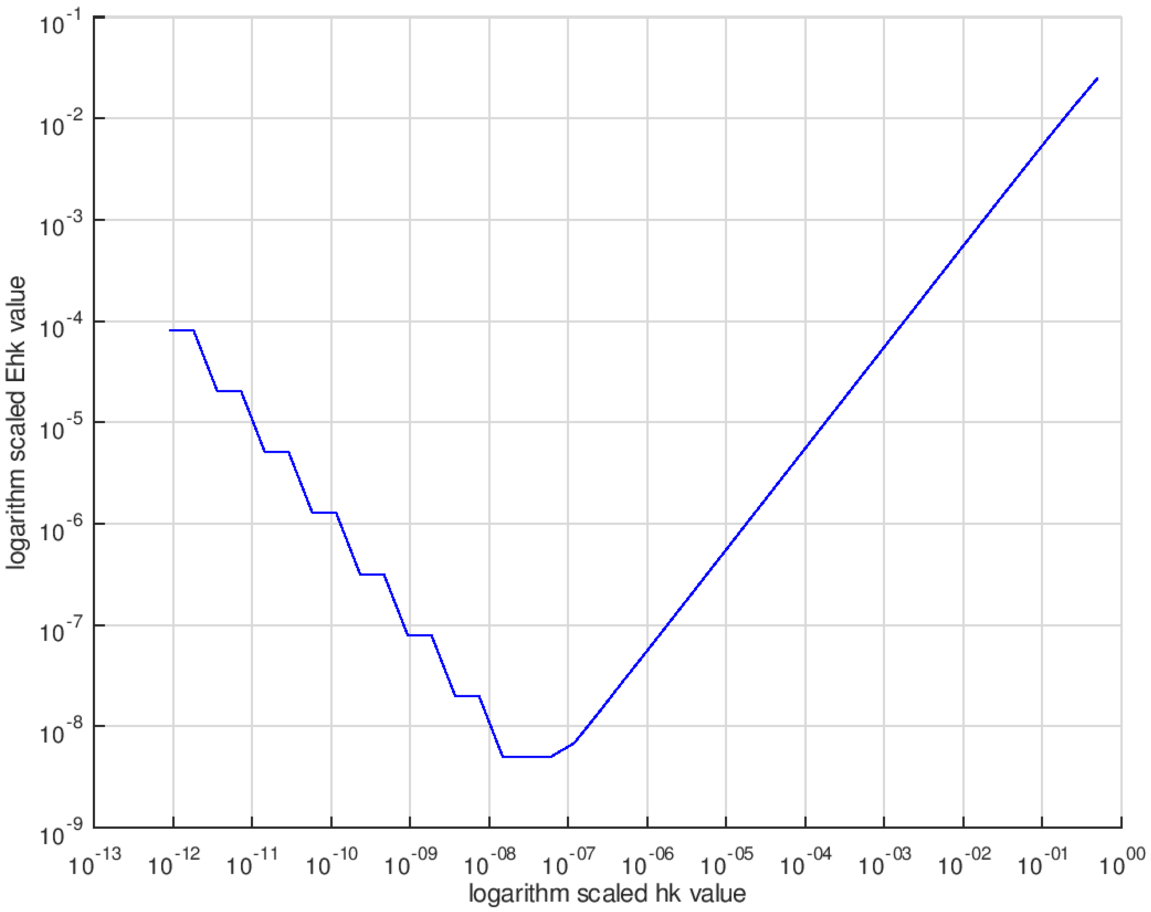
**Output of the code**

```
>> q5

k      hk           f'hk         Ehk
1      0.500000     0.308301     0.025032
2      0.250000     0.320171     0.0131625
3      0.125000     0.326576     0.00675738
4      0.062500     0.329909     0.00342474
5      0.031250     0.331609     0.00172415
6      0.015625     0.332468     0.000865053
7      0.007812     0.332900     0.000433276
8      0.003906     0.333117     0.000216826
9      0.001953     0.333225     0.00010846
10      0.000977     0.333279     5.42417e-05
```

| | | | |
|---|---|---|---|
| 11 | 0.000488 | 0.333306 | 2.71238e-05 |
| 12 | 0.000244 | 0.333320 | 1.35626e-05 |
| 13 | 0.000122 | 0.333327 | 6.7815e-06 |
| 14 | 0.000061 | 0.333330 | 3.3908e-06 |
| 15 | 0.000031 | 0.333332 | 1.69541e-06 |
| 16 | 0.000015 | 0.333332 | 8.47712e-07 |
| 17 | 0.000008 | 0.333333 | 4.23858e-07 |
| 18 | 0.000004 | 0.333333 | 2.11953e-07 |
| 19 | 0.000002 | 0.333333 | 1.06016e-07 |
| 20 | 0.000001 | 0.333333 | 5.3163e-08 |
| 21 | 0.000000 | 0.333333 | 2.68531e-08 |
| 22 | 0.000000 | 0.333333 | 1.3349e-08 |
| 23 | 0.000000 | 0.333333 | 6.8297e-09 |
| 24 | 0.000000 | 0.333333 | 4.96705e-09 |
| 25 | 0.000000 | 0.333333 | 4.96705e-09 |
| 26 | 0.000000 | 0.333333 | 4.96705e-09 |
| 27 | 0.000000 | 0.333333 | 1.98682e-08 |
| 28 | 0.000000 | 0.333333 | 1.98682e-08 |
| 29 | 0.000000 | 0.333333 | 7.94729e-08 |
| 30 | 0.000000 | 0.333333 | 7.94729e-08 |
| 31 | 0.000000 | 0.333333 | 3.17891e-07 |
| 32 | 0.000000 | 0.333333 | 3.17891e-07 |
| 33 | 0.000000 | 0.333332 | 1.27157e-06 |
| 34 | 0.000000 | 0.333332 | 1.27157e-06 |
| 35 | 0.000000 | 0.333328 | 5.08626e-06 |
| 36 | 0.000000 | 0.333328 | 5.08626e-06 |
| 37 | 0.000000 | 0.333313 | 2.03451e-05 |
| 38 | 0.000000 | 0.333313 | 2.03451e-05 |
| 39 | 0.000000 | 0.333252 | 8.13802e-05 |
| 40 | 0.000000 | 0.333252 | 8.13802e-05 |

**Output Graph ( log Ehk vs log hk )**




**(d)** when N increases Ehk decreases and reach it's minimum value and then increases. Therefore, Ehk not tends to zero when N → 0. It happens because machine can handle only up to certain precision value. When Ehk values too small machine will round off. Therefore after some point roundoff constant dominates In that time Ehk value increases.

**(e)** From the graph Ehk minimum is 4.96705e-09,
when k = 26 ( hk = 0.000000).
Therefore $k_{min}$ = 26 minimizes the Ehk.