Sketches of Solutions

Assignment 3 ROOTS OF EQUATIONS 2

1. Use ONE-POINT ITERATION method to find the root of following equation

$$f(x) = e^{-\frac{x}{4}}(2-x) - 1$$

with 2 initial guesses of 0 and the last digit of your student ID. For instance, if your student ID is 60-040626-1006-6, the initial guess will be 6 (there are 2 cases)

1.1 show your work for 4 iterations

Rearrange the given equation to

$$x = 2 - e^{\frac{x}{4}} \qquad (1)$$

Then rewrite equation (2) into iterative form

$$x_{i+1} = 2 - e^{\frac{x_i}{4}} \qquad (2)$$

From equation (2), we can apply One-point iteration method to approximate its root as follow

Case 1: initial guess; x = 0, ε_S = 0.000001%

Iteration	x_i	x_{i+1}	**ε(%)**
1	0.000000	1.000000	N/A
2	1.000000	0.715975	39.669762%
3	0.715975	0.803987	10.946979%
4	0.803987	0.777379	3.422732%
5	0.777379	0.785485	1.031944%
…	…	…	…
17	0.783596	0.783596	0.000001%
18	0.783596	0.783596	0.000000%

Case 2: initial guess; x = 6, ε_S = 0.000001%

Iteration	x_i	x_{i+1}	**ε(%)**
1	6.000000	-2.481689	N/A
2	-2.481689	1.462283	269.713361%

3	1.462283	0.558664	161.746497%
4	0.558664	0.850110	34.283395%
5	0.850110	0.763200	11.387670%
...	...	...	...
19	0.783596	0.783596	0.000001%
20	0.783596	0.783596	0.000000%

1.2 write a program to find it with the tolerance below 0.000001

```
// Node.js (JavaScript)
// One-point Iteration Method
var x, new_x, i = 0;
var approx_err, eps;
// equation new_x = 2 - e^(x/4)
eps = 0.000001;
// initial guess : x = 0
x = 0.00;
new_x = 2 - Math.exp(x/4);
approx_err = 100*Math.abs((new_x - x)/new_x);
console.log('iteration ' + i + ': x_i = ' + x + ', x_i+1 = ' + new_x + ', approx_err = ' + approx_err);
do {
    i++;
    x = new_x;
    new_x = 2 - Math.exp(x/4);
    approx_err = 100*Math.abs((new_x - x)/new_x);
    console.log('iteration ' + i + ': x_i = ' + x + ', x_i+1 = ' + new_x + ', approx_err = ' + approx_err);
} while(approx_err > eps);
```

2. Use Taylor series to approximate value of the following function

$$f(x) = \ln x$$

when $x = 4$ with an initial guess of $x_0 = 2$ using n from 0 to 3 for Taylor series. Also, show error obtained from using each n^{th} term in the approximation

$\ln 4 = 1.386294$, $x_0 = 2$

$n = 0;\ f(x) \cong f(x_0) = 0.693147$; ε = ln 4 – 0.693147 = 0.693147

$n = 1;\ f(x) \cong f(x_0) + (x - x_0)\dfrac{d(f(x)}{dx} = 1.693147$; ε = ln 4 – 1.693147 = -0.306853

$n = 2;\ f(x) \cong f(x_0) + (x - x_0)f'(x_0) + \dfrac{(x - x_0)^2}{2!}f''(x_0) = 1.193147$; ε = ln 4 – 1.193147 = 0.193147

$n = 3;\ f(x) \cong f(x_0) + (x - x_0)f'(x_0) + \dfrac{(x - x_0)^2}{2!}f''(x_0) + \dfrac{(x - x_0)^3}{3!}f'''(x_0) = 1.526485$;
ε = ln 4 – 1.526485 = -0.140861

3. Use Newton-Raphson method to find the value of $\sqrt{7}$ using x = 2.0 as the initial guess

3.1 show your work for 4 iterations

Rearrange the given equation to

$$f(x) = x^2 - 7 = 0 \qquad (1)$$

Then rewrite equation (2) into iterative form

$$x_{i+1} = x_i + \Delta x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^2 - 7}{2x_i} \qquad (2)$$

From equation (2), we can apply Newton-Raphson method to approximate its root as follow

Initial guess; x = 2, ε_S = 0.000001%

Iteration	x_i	$f(x_i)$	$f'(x_i)$	Δx_{i+1}	x_{i+1}	ε(%)
1	2.000000	-3.000000	4.000000	0.750000	2.750000	N/A
2	2.750000	0.562500	5.500000	-0.102273	2.647727	3.719008%
3	2.647727	0.010460	5.295455	-0.001975	2.645752	0.074601%
4	2.645752	0.000004	5.291504	-0.000001	2.645751	0.000028%
5	2.645751	0.000000	5.291503	0.000000	2.645751	0.000000%
6	2.645751	0.000000	5.291503	0.000000	2.645751	0.000000%

3.2 write a program to find it providing the tolerance is below 0.000001

```
// Node.js (JavaScript)
// Newton-Raphson Method
var x, new_x, delta_new_x, i = 0;
```

```
var approx_err, eps;
// equation : f(x) = x^2 - 7 = 0
eps = 0.000001;
// initial guess : x = 2.0
x = 2.00;
delta_new_x = -(x**2 - 7)/(2*x);
new_x = x + delta_new_x;
approx_err = 100*Math.abs(delta_new_x/new_x);
console.log('iteration ' + i + ': x_i = ' + x + ', x_i+1 = ' + new_x + ', approx_err = ' + approx_err);
do{
    i++;
    x = new_x;
    delta_new_x = -(x**2 - 7)/(2*x);
    new_x = x + delta_new_x;
    approx_err = 100*Math.abs(delta_new_x/new_x);
    console.log('iteration ' + i + ': x_i = ' + x + ', x_i+1 = ' + new_x + ', approx_err = ' + approx_err);
}while(approx_err > eps);
```