Linghao Shi

① (a) 1. $f(x) = 5x - 3$

Forward Error $= |x_e - x_a| = |\frac{3}{5} - 0.61| = 0.01$

Backward Error $= |f(x_a)| = |5(0.61) - 3| = 0.05$

EMF $= \frac{RFE}{RBE} = \frac{0.01}{0.05} = \frac{1}{5}$

2. $f(x) = (5x-3)^2$

Forward Error $= |x_e - x_a| = |\frac{3}{5} - 0.61| = 0.01$

Backward Error $= |f(x_a)| = |(5 \cdot 0.61 - 3)^2| = 0.0025$

EMF $= \frac{RFE}{RBE} = \frac{0.01}{0.0025} = 4$

3 $f(x) = (5x-3)^3$

Forward Error $= |x_e - x_a| = |\frac{3}{5} - 0.61| = 0.01$

Backward Error $= |f(x_a)| = |(5 \cdot 0.61 - 3)^3| = 0.000125$

EMF $= \frac{RFE}{RBE} = \frac{0.01}{0.000125} = 80$

4. $f(x) = (5x-3)^7$

Forward Error $= |x_e - x_a| = |\frac{3}{5} - 0.61| = 0.01$

Backward Error $= |f(x_a)| = |(5 \cdot 0.61 - 3)^7| = 7.8125 \times 10^{-10}$

EMF $= \frac{RFE}{RBE} = \frac{0.01}{7.8125 \times 10^{-10}} = 1.28 \times 10^7$

5. $f(x) = (5x-3)^{\frac{1}{3}}$

Forward Error $= |x_e - x_a| = |\frac{3}{5} - 0.61| = 0.01$

Backward Error $= |f(x_a)| = |(5 \cdot 0.61 - 3)^{\frac{1}{3}}| \approx 0.3684$

EMF $= \frac{RFE}{RBE} = \frac{0.01}{0.3684} \approx 0.02714417$

(b) Using bisect.m from previous hw, interval $[0, 1]$, tol $= 10^{-12}$, MaxIteration $= 100$

Function output result: 0.3335593

FE $= |\text{Output} - \frac{1}{3}| \approx 2.2238 \times 10^{-4}$

BE $= |f(\text{output})| \approx 8.6736 \times 10^{-18}$

EMF $= \frac{FE}{BE} \approx 2.564 \times 10^{14}$

Now, use fzeor to compute the function

Output = 0.3330

$FE = \left| 0.3330 - \frac{1}{3} \right| \approx 3.392 \times 10^{-4}$

$BE = |f(0.3330)| \approx 5.2 \times 10^{-18}$

$EMF = \frac{FE}{BE} \approx 6.52 \times 10^{13}$

Since both EMF are huge numbers and condition number is also big, then we can conclude that both are ill-conditioned

(2) (a) for linear function $f(x) = a + bx$ when $b \neq 0$

we can tell that our true root is $x = x_0 = -\frac{b}{a}$

According to Newton's method: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

Substitute values into this formula, we get:

$$x_1 = x_0 - \frac{ax_0 + b}{a}$$

$$x_1 = \cancel{x_0} - \cancel{x_0} - \frac{b}{a}$$

$$x_1 = -\frac{b}{a} \quad \text{(it matches our true root calculated above)}$$

Thus, part (a) is proved

(b) $f(x) = x^3 - a = 0$

since $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$ substituting using i and $x^3 - a$

$$x_{i+1} = x_i - \frac{x_i^3 - a}{3x_i^2}$$

$$x_{i+1} = \frac{2x_i^3 + a}{3x_i^2}$$

x_1 (at $x=1, a=8$) $= \frac{2+8}{3} = \frac{10}{3}$

x_2 (at $x=\frac{10}{3}, a=8$) ≈ 2.46

x_3 (at $x=2.46, a=8$) ≈ 2.08

x_4 (at $x=2.08, a=8$) ≈ 2.0031

} using calculator

(c) since $x_{i+1} = \frac{2x_i^3 + a}{3x_i^2}$ (when $p=3$ $f(x) = x^3 - a$)

It implies that at $f(x) = x^p - a = 0$

$$f'(x) = px^{p-1}$$

thus, we can conclude our formula:

$$x_{i+1} = \frac{(p-1) \cdot x_i^p + a}{p x_i^{p-1}}$$

③ (a) At N steps, we can calculate the values using $x_n = x_e + e_n$

First term is the exact root, second term is the error at step N.

Then use Taylor series to expand using $x_n = x_e + e_n$

$f(x_n) = f(x_e) + e_n \cdot f'(x_e) + \frac{e_n^2}{2!} f''(x_e) + \dots$

As we approaching to the true root, e_n will become smaller and smaller.

Then, for Taylor expansion at $f(x_n)$, we just need to consider the first two terms.

$$\frac{f(x_n)}{f'(x_e)} \approx \frac{f(x_e) + e_n \cdot f'(x_e)}{f'(x_e)} = \underline{\frac{f(x_e)}{f'(x_e)}} + \frac{e_n \cdot \cancel{f'(x_e)}}{\cancel{f'(x_e)}} \quad (\text{reduced to } e_n)$$

Since x_e is our true root, $f(x_e) = 0$, we can ignore first term.

$\frac{f(x_n)}{f'(x_e)} \approx e_n$ proved

(b) Using conclusion from part (a)

since $e_n \approx \frac{f(x_n)}{f'(x_e)}$, then $e_{n+1} \approx \frac{f(x_{n+1})}{f'(x_e)}$

$$\frac{e_{n+1}}{(e_n)^p} = \frac{\frac{f(x_{n+1})}{f'(x_e)}}{\left(\frac{f(x_n)}{f'(x_e)}\right)^p} = \frac{f(x_{n+1}) \cdot [f'(x_e)]^{p-1}}{[f(x_n)]^p} \approx M = \text{constant}$$

From the given info, we assume $f'(x_e)$ is a constant

Thus, we can conclude that $\frac{f(x_{n+1})}{[f(x_n)]^p} \approx N = \text{constant}$

(c) Given that $R_n = \frac{|f(x_n)|}{|f(x_{n-1})|}$

From part (b), we know that $\underline{\frac{f(x_{n+1})}{[f(x_n)]^p} \approx N}$

$$\Rightarrow (R_n)^p \approx \frac{|f(x_n)|^p}{|f(x_{n-1})|^p} = \frac{|f(x_n)|^p}{|f(x_n)|} \cdot N \qquad (1)$$

$$R_n = \frac{|f(x_n)|}{|f(x_{n-1})|}$$

$$\Rightarrow R_{n+1} = \frac{|f(x_{n+1})|}{|f(x_n)|} \qquad (f(x_{n+1}) \approx N f(x_n)^p \text{ from part b})$$

$$\Rightarrow R_{n+1} = \frac{|N f(x_n)^p|}{|f(x_n)|} \qquad ②$$

We can see that $(R_n)^p \approx R_{n+1}$ $\quad ① \approx ②$

$$\Rightarrow p \cdot \log(R_n) \approx \log(R_{n+1})$$

$$p \approx \frac{\log(R_{n+1})}{\log(R_n)}$$

Solution:

4. (a) Here is my Matlab code for solveEquationByNewton.m,

Listing 1: solveEquationByNewton.m

```
%  Find an approximate solution to a scalar equation
%         f(x)=0
%  by  Newtons  method
%
% f (input) : function f(x)
% fx (input) : function that defines f(x)
% x0 (input) : initial guess
% tol (input) : convergence tolerance
% xc (output) : approximate solution
% maxIterations (input) : maximum number of iterations
function [xc] = solveEquationByNewton(f,fx,x0,tol,maxIterations)
k = 1;
xc = x0;
fc = f(xc);
while k <= maxIterations
    %Check if tolerance condition is satified
    if tol > abs(fc)
        break
    end

    %Calculate current x value and ratio
    xk = xc - f(xc)/fx(xc);
    currratio = abs(f(xk)) / abs(fc);

    %Output result
    if k==1  % do not print p first time
        fprintf("Newton: it=%2d: x=%13.6e f(x)=%9.2e ratio=%9.2e\n",k,xc,fc,currratio);
    else
        %Calculate order of convergence
        p = log(currratio)/log(Oldratio);
        fprintf("Newton: it=%2d: x=%13.6e f(x)=%9.2e ratio=%9.2e p=%4.2f\n",k,xc,fc,currratio,p);
    end

    %Store the current value for future computation
    Oldratio = currratio;
    xc = xk;
    fc = f(xc);
    k = k + 1;

    %Stop at maxIterations
    if k > maxIterations
        fprintf("Maximun number of iteration is reached")
        break
    end
end
return
```

(b) Use Newton solver by finding $\sqrt{7}$ by looking for a root to f(x) = x^2 - 7 = 0 using an initial guess $x_0 = 1$

From the below output result, we can see that at the last few steps, we can see an approximate quadratic convergence and estimated convergence rate is mostly above 2. Thus, we can conclude that this is quadratically convergence and my Newton solver worked.

Listing 2: Results from solveEquationByNewton.m

```
>> solveEquationByNewton.m
>> f = @(x) x.^2 - 7
>> fx = @(x) 2.*x
>> solveEquationByNewton(f, fx, 1, 10.^-12, 20)
Newton: it= 1: x= 1.000000e+00 f(x)=-6.00e+00 ratio= 1.50e+00
Newton: it= 2: x= 4.000000e+00 f(x)= 9.00e+00 ratio= 1.41e-01 p=-4.84
Newton: it= 3: x= 2.875000e+00 f(x)= 1.27e+00 ratio= 3.83e-02 p=1.66
Newton: it= 4: x= 2.654891e+00 f(x)= 4.84e-02 ratio= 1.72e-03 p=1.95
Newton: it= 5: x= 2.645767e+00 f(x)= 8.33e-05 ratio= 2.97e-06 p=2.00
Newton: it= 6: x= 2.645751e+00 f(x)= 2.48e-10 ratio= 3.59e-06 p=0.99
ans =  2.645751
```

(c)

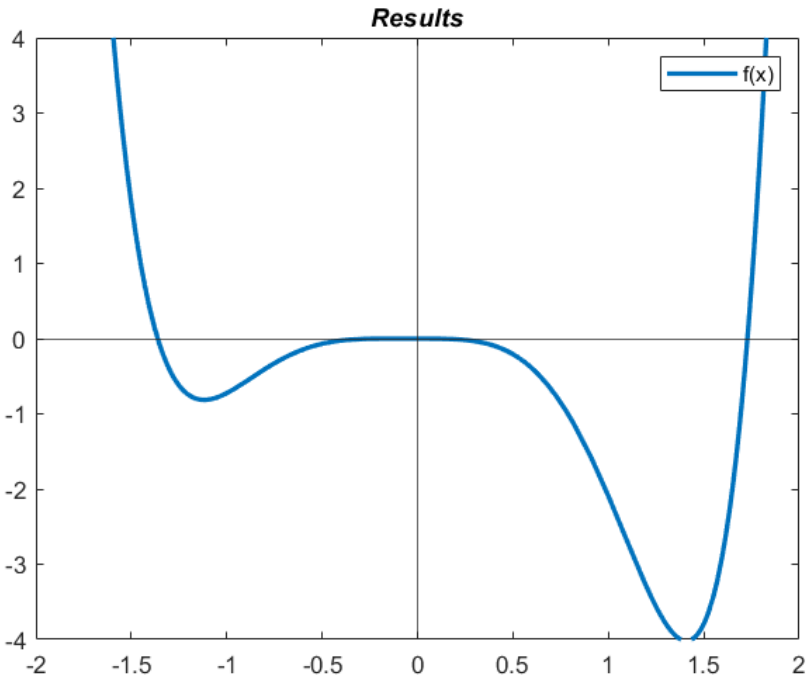

Figure 1: graph for f(x) to find four roots

From the graph, we can see that in the interval given, there are three roots, we can distinguish them by determining initial guesses and use my Newton solver to calculate them for each root. A reasonable guess for each should be x = -1.5, -0.25 and 1.5.

Listing 3: initial guess x = -1.5

```
>> solveEquationByNewton.m
>> f =@(x)1/2*exp((sin(x))^3)+x.^6-2.5.*x.^4-x.^3-1/2
>> fx = @(x) 6.*x.^5-10.*x.^3-3.*x.^2+3/2*exp((sin (x)).^3).*(sin(x)).^2.*cos (x)
>> solveEquationByNewton(f,fx,-1.5,10^(-12),20)
Newton: it= 1: x=-1.500000e+00 f(x)= 1.79e+00 ratio= 2.20e-01
Newton: it= 2: x=-1.403112e+00 f(x)= 3.95e-01 ratio= 1.07e-01 p=1.48
Newton: it= 3: x=-1.366611e+00 f(x)= 4.21e-02 ratio= 1.64e-02 p=1.84
Newton: it= 4: x=-1.361697e+00 f(x)= 6.88e-04 ratio= 2.82e-04 p=1.99
Newton: it= 5: x=-1.361614e+00 f(x)= 1.94e-07 ratio= 7.10e-08 p=2.01
ans = -1.361614
```

Listing 4: initial guess x = -0.25

```
>> solveEquationByNewton.m
>> solveEquationByNewton( f,fx,-0.25,10^(-12),20)
Newton: it= 1: x=-2.500000e-01 f(x)=-1.41e-03 ratio= 2.28e-01
Newton: it= 2: x=-2.220647e-01 f(x)=-3.22e-04 ratio= 1.27e-01 p=1.40
Newton: it= 3: x=-2.107807e-01 f(x)=-4.08e-05 ratio= 2.55e-02 p=1.78
Newton: it= 4: x=-2.088833e-01 f(x)=-1.04e-06 ratio= 7.08e-04 p=1.98
Newton: it= 5: x=-2.088323e-01 f(x)=-7.36e-10 ratio= 4.53e-07 p=2.01
ans = -0.2088323
```

Listing 5: initial guess x = 1.5

```
>> solveEquationByNewton.m
>> solveEquationByNewton( f,fx,1.75,10^(-12),20)
Newton: it= 1: x= 1.750000e+00 f(x)= 7.13e-01 ratio= 5.00e-02
Newton: it= 2: x= 1.729654e+00 f(x)= 3.56e-02 ratio= 2.95e-03 p=1.94
Newton: it= 3: x= 1.728524e+00 f(x)= 1.05e-04 ratio= 8.79e-06 p=2.00
Newton: it= 4: x= 1.728521e+00 f(x)= 9.25e-10 ratio= 3.84e-06 p=1.07
ans = 1.728521
```

Looking at the output result for each root, we can see that at the final two iterations before convergence, correct digits to the right of the decimal point are roughly doubling for all of them (P is round 2). Thus, we can conclude that all three roots converge quadratically using my Newton Solver.

5. The matlab code is given below.

Listing 6: Code for solveEquationBySecantMethod.m

```
% Find an approximate solution to a scalar equation
% f(x)=0
% by the Secant Method.
%
% f (input) : function f(x)
% x0,x1 (input) : initial guesses
% tol (input) : convergence tolerance
% xc (output) : approximate solution
% maxIterations (input) : maximum number of iterations
function [xc] = solveEquationBySecantMethod( f,x0,x1,tol,maxIterations )
k = 1;
fa = f(x0);
fb = f(x1);
while k <= maxIterations
    %Calculate the new x value and ratio
    xc = x1 - ((x1 - x0)/(fb - fa)) * fb;
    currratio = abs(fa)/ abs(fb);

    %Check if satisfies the tolerance
    fc = f(xc);
    error = abs(fc);
    if tol > error
        break
    end
    nextratio = abs(fc)/ abs(fb);
    p = log(nextratio)/log(currratio);
    fprintf("Secant: it=%2d: x=%13.6e f(x)=%9.2e ratio=%9.2e p=%4.2f\n",k,xc,fa,nextratio,p);

    %store the value for next iteration calculation
    x0 = x1;
    x1 = xc;
    fa = fb;
    fb = fc;
    k = k + 1;

    %Check if reaches the max iterations
    if k > maxIterations
        fprintf("Maximun iteration has reached")
    end
end
return
```

(b) Use Newton solver by finding $\sqrt{3}$ by looking for a root to f(x) = x^2 - 3 = 0 using an initial guess $x_0 = 1$ and $x_1 = 2.0$. By looking at the output result, we can tell that this is superlinearly convergence because it is roughly less than 2 but always greater than 1 (also p value proves it) and we can conclude that my secant-method solver worked.

Listing 7: Computation using solveEquationBySecantMethod.m

```
>> f = @(x) x.^2 - 3
>> solveEquationBySecantMethod( f,1,2,10.^-12,20)
Secant: it= 1: x= 1.666667e+00 f(x)=-2.00e+00 ratio= 2.22e-01 p=-2.17
Secant: it= 2: x= 1.727273e+00 f(x)= 1.00e+00 ratio= 7.44e-02 p=-1.73
```

```
Secant: it= 3: x= 1.732143e+00 f(x)=-2.22e-01 ratio= 1.93e-02 p=-1.52
Secant: it= 4: x= 1.732051e+00 f(x)=-1.65e-02 ratio= 1.38e-03 p=-1.67
Secant: it= 5: x= 1.732051e+00 f(x)= 3.19e-04 ratio= 2.66e-05 p=-1.60
ans = 1.732051
```

(c) Looking at the graph from previous question, we can roughly determine three intervals:[-1.5, -1.25], [-0.25, 0.25], [1.5 ,2].

Listing 8:]interval [-1.5, -1.25]

```
>> f = @(x)1/2*exp((sin(x))^3)+x.^6-2.5.*x.^4-x.^3-1/2
>> solveEquationBySecantMethod( f,-1.5,-1.25,10^(-12),20)
Secant: it= 1: x=-1.314419e+00 f(x)= 1.79e+00 ratio= 5.33e-01 p=-0.59
Secant: it= 2: x=-1.387973e+00 f(x)=-6.23e-01 ratio= 7.18e-01 p=-0.53
Secant: it= 3: x=-1.357225e+00 f(x)=-3.32e-01 ratio= 1.50e-01 p=-5.73
Secant: it= 4: x=-1.361238e+00 f(x)= 2.39e-01 ratio= 8.67e-02 p=-1.29
Secant: it= 5: x=-1.361619e+00 f(x)=-3.58e-02 ratio= 1.51e-02 p=-1.72
Secant: it= 6: x=-1.361614e+00 f(x)=-3.11e-03 ratio= 1.28e-03 p=-1.59
Secant: it= 7: x=-1.361614e+00 f(x)= 4.68e-05 ratio= 1.92e-05 p=-1.63
ans =  -1.361614
```

Listing 9:]Interval [-0.25, 0.25]

```
>> solveEquationBySecantMethod( f,-0.25,0.25,10^(-12),20)
Secant: it= 1: x=-2.938057e-01 f(x)=-1.41e-03 ratio= 2.64e-01 p=0.53
Secant: it= 2: x=-4.886641e-01 f(x)=-1.75e-02 ratio= 1.33e+01 p=1.94
Secant: it= 3: x=-2.779446e-01 f(x)=-4.62e-03 ratio= 5.23e-02 p=1.14
Secant: it= 4: x=-2.663215e-01 f(x)=-6.14e-02 ratio= 7.36e-01 p=-0.10
Secant: it= 5: x=-2.338822e-01 f(x)=-3.21e-03 ratio= 3.00e-01 p=-3.93
Secant: it= 6: x=-2.199873e-01 f(x)=-2.36e-03 ratio= 3.73e-01 p=-0.82
Secant: it= 7: x=-2.117222e-01 f(x)=-7.09e-04 ratio= 2.32e-01 p=-1.48
Secant: it= 8: x=-2.092280e-01 f(x)=-2.64e-04 ratio= 1.32e-01 p=-1.38
Secant: it= 9: x=-2.088477e-01 f(x)=-6.13e-05 ratio= 3.88e-02 p=-1.61
Secant: it=10: x=-2.088324e-01 f(x)=-8.10e-06 ratio= 5.49e-03 p=-1.60
ans = -0.2088323
```

Listing 10:]Interval [1.5, 2]

```
>> solveEquationBySecantMethod( f,1.5,2,10^(-12),20)
Secant: it= 1: x= 1.593151e+00 f(x)=-3.79e+00 ratio= 1.78e-01 p=1.17
Secant: it= 2: x= 1.654488e+00 f(x)= 1.66e+01 ratio= 6.48e-01 p=-0.25
Secant: it= 3: x= 1.767496e+00 f(x)=-2.94e+00 ratio= 7.10e-01 p=-0.79
Secant: it= 4: x= 1.720577e+00 f(x)=-1.91e+00 ratio= 1.80e-01 p=-5.00
Secant: it= 5: x= 1.727748e+00 f(x)= 1.35e+00 ratio= 9.92e-02 p=-1.35
Secant: it= 6: x= 1.728537e+00 f(x)=-2.44e-01 ratio= 2.12e-02 p=-1.67
Secant: it= 7: x= 1.728521e+00 f(x)=-2.42e-02 ratio= 2.03e-03 p=-1.61
Secant: it= 8: x= 1.728521e+00 f(x)= 5.12e-04 ratio= 4.28e-05 p=-1.62
ans = 1.728521
```

Looking at the output result, we can tell that this is superlinearly convergence because it is converging mostly less than 2 and greater than 1 (also p value proves it).