Math 585 Homework 3
Due February 17th
By Marvyn Bailly

Problem 1 *T1: The equation $x^2 - a = 0$ (for the square root $\alpha = \sqrt{a}$) can be written equivalently in the form*

$$x = \varphi(x),$$

in many different ways, for example,

$$\varphi(x) = \frac{1}{2}\Big(x + \frac{a}{x}\Big), \quad \varphi(x) = \frac{a}{x}, \quad \varphi(x) = 2x - \frac{a}{x}.$$

Discuss the convergence (or nonconvergence) behavior of the iteration $x_{n+1} = \varphi(x_n), n = 0, 1, 2, \ldots$, for each of these three iteration functions. In case of converge, determine the order of convergence.

Solution.
Consider fixed point iteration of the form

$$x = \varphi(x),$$

applied to the equation $x^2 - a = 0$ (for the square root $\alpha = \sqrt{a}$) where $\varphi(x)$ are the following functions

$$\varphi(x) = \frac{1}{2}\Big(x + \frac{a}{x}\Big), \quad \varphi(x) = \frac{a}{x}, \quad \varphi(x) = 2x - \frac{a}{x}.$$

First let's consider when $\varphi(x) = \frac{1}{2}\big(x + \frac{a}{x}\big)$. Let $I_\epsilon = \{x \in \mathbb{R} : |x - \alpha| \leq \epsilon\}$ and then notice that

$$\max_{t \in I_\epsilon} |\varphi'(t)| = \max_{t \in I_\epsilon} \left|\frac{1}{2} - \frac{a}{2t^2}\right| < 1,$$

and thus $\varphi(x)$ converges by the Fixed Point Convergence Theorem. Next observe that

$$\varphi'(x)|_\alpha = \frac{1}{2} - \frac{a}{2x^2}\bigg|_\alpha = 0$$
$$\varphi''(x)|_\alpha = \frac{a}{x^3}\bigg| = a^{-1/2} \neq 0.$$

Thus we see that $\varphi(x)$ quadratically converges.
Next let's consider when $\varphi(x) = \frac{a}{x}$. Notice that application of fixed point iterations with some initial guess $x_0 \neq \alpha$ results in

$$x_1 = \varphi(x_0) = \frac{a}{x_0}$$
$$\implies x_2 = \varphi x_1 = \frac{a}{x_1} = \frac{a}{a/x_0} = x_0.$$

Thus the iteration is a cycle for any initial guess other than α and does not converge. Also note that

$$\varphi'(x)|_{\alpha} = -1 \neq 0.$$

Finally consider when $\varphi(x) = 2x - \frac{a}{x}$. Let $I_\epsilon = \{x \in \mathbb{R} : |x - \alpha| \leq \epsilon\}$ and then notice that

$$\max_{t \in I_\epsilon} |\varphi'(t)| = \max_{t \in I_\epsilon} \left|2 + \frac{a}{t^2}\right| \not< 1,$$

as t^2 and a are nonnegative. Thus $\varphi(x)$ does not convergence. Also note that

$$\varphi'(x)|_{\alpha} = 3 \neq 0.$$

□

Problem 2 *T2: Consider the function* $g(x) = x^2 + \frac{3}{16}$.

(a) This function has two fixed points. What are they?

(b) Consider the fixed point iteration $x_{n+1} = g(x_n)$ *for this* g. *For which of the points you have found in (a) can you be sure that the iterations will converge to that fixed point? Briefly justify your answer. You may assume that the initial guess is sufficiently close to the fixed point.*

(c) For the point or points you found in (b), roughly how many iterations will be required to reduce the convergence error by a factor of 10?

Solution.
Consider the function $g(x) = x^2 + \frac{3}{16}$.

(a) First let's find the fixed points of the function by observing that

$$\alpha = \alpha^2 + \frac{3}{16} \implies 0 = \alpha^2 - \alpha + \frac{3}{16} \implies \alpha \in \left\{\frac{1}{4}, \frac{3}{4}\right\}.$$

And thus we have found the two fixed points.

(b) Note that $g'(x) = 2x$. Then for $\alpha = \frac{1}{4}$ we have that $|2 \cdot 1/4| < 1$ which implies convergence by the Fixed Point Convergence Theorem but by the same logic, fixed point iteration for $\alpha = \frac{3}{4}$ may not converge.

(c) To find the approximate amount of iterations it will take to converge to $\alpha = \frac{1}{4}$, consider an initial guess sufficiently close to the fixed point meaning that $x_n = \frac{1}{4} + \epsilon$ where $\epsilon \ll 1$. Now observe that

$$x_{n+1} = \left(\frac{1}{4} + \epsilon\right)^2 + \frac{3}{16} = \epsilon^2 + \frac{1}{2}\epsilon + \frac{1}{4},$$

which implies that $x_{n+1} - \frac{1}{4} = \epsilon^2 + \frac{1}{2}\epsilon$. Now assuming that $\epsilon_n \to 0$ as $n \to \infty$, we can find the rate of convergence

$$\lim_{n\to\infty} \left| \frac{x_{n+1} - \frac{1}{4}}{x_n - \frac{1}{4}} \right| = \lim_{n\to\infty} \epsilon_n + \frac{1}{2} = \frac{1}{2}$$

Thus we have that this method converges at most linearly and for a sufficiently close guess with ϵ_0 being small $\left|x_n - \frac{1}{4}\right|$ is around $\frac{1}{2^n}\epsilon_0$. Thus to reduce the converge error by a factor of 10, we expect it to take four steps.

□

Problem 3 *T3: Assume $f(x)$ is sufficiently smooth. α is a root of $f(x)$ of multiplicity m, $m \geq 2$.*

(a) Show that Newton's method converges locally with order one.

(b) Set $g(x) = \frac{f(x)}{f'(x)}$, show that Newton's method applied to function $g(x)$ converges locally with (at least) order two.

(c) Consider the iteration

$$x_{n+1} = x_n - m\frac{f(x_n)}{f'(x_n)},$$

show that this method converges locally with (at least) order 2.

Solution.
Assume $f(x)$ is sufficiently smooth. α is a root of $f(x)$ of multiplicity m, $m \geq 2$.

(a) Since $f(x)$ is sufficiently smooth and α is a root of multiplicity m, note that

$$f(\alpha) = f'(\alpha) = \cdots = f^{(m-1)}(\alpha) = 0.$$

Now fixing x and Taylor Expanding $f(x)$ around $x = \alpha$ yields

$$\begin{aligned} f(x) &= f(\alpha) + f'(\alpha)(x-\alpha) + \cdots + \frac{f^{(m-1)}(\alpha)}{(m-1)!}(x-\alpha)^{m-1} + \frac{f^{(m)}(\xi)}{m!}(x-\alpha)^m \\ &= \frac{f^{(m)}(\xi)}{m!}(x-\alpha)^m. \end{aligned}$$

where $\xi \in [\alpha, x]$. And Taylor Expanding $f'(x)$ around $x = \alpha$ gives

$$\begin{aligned} f'(x) &= f'(\alpha) + \cdots + \frac{f^{(m-1)}(\alpha)}{(m-2)!}(x-\alpha)^{m-2} + \frac{f^{(m)}(\hat{\xi})}{(m-1)!}(x-\alpha)^{m-1} \\ &= \frac{f^{(m)}(\hat{\xi})}{(m-1)!}(x-\alpha)^{m-1}, \end{aligned}$$

where $\hat{\xi} \in [\alpha, x]$. Now looking at the $(n+1)^{\text{th}}$ step of Newton's method shows

$$\begin{aligned} x_{n+1} &= x_n - \frac{f(x)}{f'(x)} \\ &= x_n - \left(\frac{1}{m}\right)\left(\frac{f^{(m)}(\xi_n)}{f^{(m)}(\hat{\xi}_n)}\right)(x-\alpha). \end{aligned}$$

Subtracting α and dividing $x_n - \alpha$ from both sides shows that the error is given by

$$\frac{x_{n+1}-\alpha}{x_n-\alpha} = 1 - \frac{f^{(m)}(\xi_n)}{f^{(m)}(\hat{\xi}_n)}.$$

Then assuming that $x_n \to \alpha$ as $n \to \infty$ gives that $\xi_n, \hat{\xi}_n \to \alpha$ and thus

$$\lim_{n\to\infty} \left| \frac{x_{n+1} - \alpha}{x_n - \alpha} \right| = 1 - \frac{1}{m} < 1.$$

Therefore, Newton's Method converge locally with order one.

(b) Let $g(x) = \frac{f(x)}{f'(x)}$. Observe that for a fixed x we have that

$$g(x) = \frac{f^{(m)}(\xi)}{m f^{(m)}(\hat{\xi})}(x - \alpha),$$

which means that

$$g'(\alpha) = \lim_{x\to\alpha} \frac{g(x) - g(\alpha)}{x - \alpha} = \frac{1}{m} \neq 0.$$

Now applying Newton's Method on $g(x)$ is the same as fixed point iteration

$$\varphi(x) = x - \frac{g(x)}{g'(x)}.$$

Then

$$\varphi'(\alpha) = 1 - \frac{g'(\alpha)^2 - g(\alpha)g''(\alpha)}{g'(\alpha)^2} = \frac{g(\alpha)g''(\alpha)}{g'(\alpha)^2} = 0,$$

recalling that $g'(\alpha) \neq 0$. Therefore, Newton's Method applied to $g(x)$ converges locally with at least order two.

(c) Consider the iteration

$$\varphi(x) = x - m\frac{f(x)}{f'(x)}$$

which for a fixed x, simplifies to

$$\varphi(x) = x - \frac{f^{(m)}(\xi)}{f^{(m)}(\hat{\xi})}(x - \alpha).$$

Then

$$\begin{aligned}
\varphi(\alpha)' &= \lim_{x\to\alpha} \frac{\phi(x) - \phi(\alpha)}{x - \alpha} \\
&= \lim_{x\to\alpha} \frac{x - \frac{f^{(m)}(\xi)}{f^{(m)}(\hat{\xi})}(x - \alpha) - \alpha}{x - \alpha} \\
&= \lim_{x\to\alpha} 1 - \frac{f^{(m)}(\xi)}{f^{(m)}(\hat{\xi})} \\
&= 1 - 1 = 0.
\end{aligned}$$

Therefore, the iteration converges locally with at least order two by the fixed point iterated theorem.

□

Problem 4 *C1: For the equation*

$$\frac{1}{2}x - \sin(x) = 0,$$

it is easy to see that the only positive root is located in the interval $[\pi/2, \pi]$.

(a) Use the method of bisection on $[\pi/2, \pi]$ *to approximate the root to* $3, 7,$ *and* 15 *decimal places. report the number of iterations needed in each case.*

(b) Repeat the same task as in (a) but using Newton's method with $x_0 = \pi$ *as the initial guess.*

(c) Repeat the same task as in (b) but using the secant method with $x_0 = \pi/2$ *and* $x_1 = \pi$ *as the initial guess.*

Solution.
Consider the equation

$$\frac{1}{2}x - \sin(x) = 0,$$

on the interval $[\pi/2, \pi]$. The following MATLAB script in Listings 1 was used to run the entire problem.

Listing 1: **Main Script for C1**

```
format long

%driver code for coding problem one

syms f(x1);
f(x1) = 0.5*x1 - sin(x1);

%Part A
interval = [pi/2, pi];
resultsA = zeros(3,2);

resultsA(1,:) = bisection(f,interval,10e-3,1);
resultsA(2,:) = bisection(f,interval,10e-7,1);
resultsA(3,:) = bisection(f,interval,10e-15,1);

%Part B
x0 = pi; %set inital guess

resultsB = zeros(3,2);

resultsB(1,:) = newton(f,10e-3,x0, false);
```

```
resultsB(2,:) = newton(f,10e-7,x0, false);
resultsB(3,:) = newton(f,10e-15,x0, false);


%Part C
x0 = pi/2;
x1 = pi;
resultsC = zeros(3,2);

resultsC(1,:) = secant(f,10e-3,x0,x1);
resultsC(2,:) = secant(f,10e-7,x0,x1);
resultsC(3,:) = secant(f,10e-15,x0,x1);

format short
```

(a) First we wish to use the bisection method on the given interval to approximate the root to $3, 7,$ and 15 decimal places. To do so, I created the script in Listings 2 for the bisection method. Running the script to approximate the root up to 3 decimals iterated 9 times before converging to 1.892932292250881. To have 7 decimals the script ran 22 times and resulted with 1.895494294831430. Finally to gain 15 decimal approximation, the script ran 49 times and found the root to be 1.895494267033980.

Listing 2: **Bisection Method Script**

```
function output = bisection(f, interval, tol, iter)
    %Input: function, interval, tolerance, and number of
       iterations
    %Ouput: approximation using Bisection method and the
       number of iterations

    mid = (interval(1) + interval(2))/2; %get mid point

    if(f(mid) == 0 || interval(2)-interval(1) < tol) %we found
       the root
        output = [mid,iter];
    elseif(f(interval(1))*f(mid) < 0) %if signs are different
       on left
        output = bisection(f, [interval(1), mid], tol, iter+1)
           ;
    elseif(f(interval(2))*f(mid) < 0) %if signs are different
       on right
        output = bisection(f, [mid, interval(2)], tol, iter+1)
           ;
    else
```

```
14          output = "Can not find root";
15      end
16  end
```

(b) Next we wish to repeat the approximation but using the Newton's method with an initial guess of $x_0 = \pi$. I created the script in Listings 3 for Newton's method. To approximate up to 3 decimals gave 1.895494285255435 after 4 iterations. To approximate up to 7 decimals gave 1.895494267033981 after 5 iterations.To approximate up to 15 decimals gave 1.895494267033981 after 6 iterations.

Listing 3: **Newton's Method Script**

```
1  function [output,error] = newton(f, tol, x0, exact)
2      %Input: function, tolerance, initial guess, and exact
           solution
3      %Ouput: approximation using Newton method, the number of
           iterations, and errors
4
5      n = length(x0);              %get dim of system
6      syms x [1, n];               %create a list of symbols x0,x1
           ,...,xn
7
8      J = jacobian(f, x);          %create a jacobian function
9
10     y1 = x0;
11     y2 = y1 + (double(subs(J,x,y1)) \ double(-subs(f,x,y1))')
           ';
12
13     iter = 1;                    %set counter for iterations
14
15     %compute the error
16     error = false;
17     if(~(isa(exact,'boolean')))
18         error = [norm(y2 - exact)];
19     end
20
21
22
23     while norm(y2 - y1) > tol
24         y1 = y2;                 %update guess
25         y2 = y1 + (double(subs(J,x,y1)) \ double(-subs(f,x,y1)
               )')'; %compute next term
26
27
```

```
28          %compute the error
29          if(~(isa(exact,'boolean')))
30              error = [error, norm(y2 - exact)];
31          end
32
33          iter = iter + 1;
34      end
35
36      output = [y2, iter];
37  end
```

(c) Finally we repeat the same task as in the previous parts but using the Secant Method with the initial guesses of $x_0 = \pi/2$ and $x_1 = \pi$. I created the script in Listings 4 for Secant Method. Running the script to approximate the root up to 3 decimals iterated 4 times before converging to 1.895352767179123. To have 7 decimals the script ran 6 times and resulted with 1.895494267064823. Finally to gain 15 decimal approximation, the script ran 8 times and found the root to be 1.895494267033981.

Listing 4: **Secant Method Script**

```
1   function output = secant(f, tol, x0, x1)
2       %Input: function, tolerance, and initial guesses
3       %Ouput: approximation using Secant method and the number
            of iterations
4
5       y0 = x0;
6       y1 = x1;
7       y2 = y1 - double((y1 - y0)/(f(y1) - f(y0))*f(y1)); %do one
            iteration
8
9       iter = 1;
10
11      while norm(y2 - y1) > tol
12          y0 = y1; %update terms
13          y1 = y2;
14          y2 = y1 - double((y1 - y0)/(f(y1) - f(y0))*f(y1));
15          iter = iter + 1;
16      end
17      output = [y2, iter];
18  end
```

□

Problem 5 *C2: Consider the nonlinear system*

$$\begin{cases} (x_1+3)(x_2^3-7)+18=0, \\ \sin(x_2e^{x_1}-1)=0. \end{cases}$$

(a) Solve it using Newton's method. Set the initial guess as $x_0=(-0.5,1.4)^T$*. Use reasonable stopping criteria.*

(b) Solve it using Broyden's method. Set the same initial guess and stop criteria.

(c) Suppose you know the exact solution $x^*=(0,1)^T$*, compute the error* $\|x_k-x^*\|$ *during the iterations of both methods and compare their rate of convergence.*

Solution.
Consider the nonlinear system

$$\begin{cases} (x_1+3)(x_2^3-7)+18=0, \\ \sin(x_2e^{x_1}-1)=0. \end{cases}$$

The Following MATLAB script in Listings 5 was used to run the entire problem.

Listing 5: **Main Script for C2**

```
clc
format long

%Driver code for coding problem 2

syms f(x1,x2);
f(x1,x2) = [(x1 + 3)*(x2^3 - 7) + 18 sin(x2*exp(x1)-1)];

%Part A and C
x0 = [-0.5, 1.4];
exact = [0,1];

[output,error] = newton(f,10e-15,x0,exact)

convergence = error(2:end) ./ (error(1:end-1)) %goes to zero
convergence = error(2:end) ./ (error(1:end-1)).^2 %seems bounded
convergence = error(2:end) ./ (error(1:end-1)).^2.2 %seems to blow
    up

figure(1)
semilogy(error)
hold on
```

```
22
23 %Part B and C
24 [output,error] = broyden(f,10e-15,x0, exact)
25
26 convergence = error(2:end) ./ (error(1:end-1)) %goes to zero
27 convergence = error(2:end) ./ (error(1:end-1)).^2 %blows up
28
29 figure(2)
30 semilogy(error)
31 hold off
32
33
34 format short
```

(a) First we wish to solve the system using Newton's method and the initial guess of $x_0 = (-0.5, 1.4)^T$ and a stopping criteria of precision of 15 decimals. Using the Newton's method script in Listings 3 we get the solution to be approximately $(0, 1)$ after 5 iterations.

(b) Next we wish to use Broyden's method to compute the solution using the same initial guess and stopping criteria from before. I created the scripted in Listings 6 for Broyden's method. Using Broyden's method we get the solution to be approximately $(0, 1)$ after 9 iterations.

Listing 6: **Broyden's Method Script**

```
1  function [output,error] = broyden(f, tol, x0, exact)
2      %Input: function, tolerance, and initial guess
3      %Ouput: approximation using Newton method and the number
           of iterations
4
5
6      n = length(x0);          %get dim of system
7      syms x [1, n];           %create a list of symbols x0,x1
           ,...,xn
8
9      J = jacobian(f, x);       %create a jacobian function
10     y1 = x0;
11     Ai = inv(double(subs(J,x,y1))); %compute
12     y2 = y1 - (Ai*double(subs(f,x,y1)'))';
13     g = (double(subs(f,x,y2) - subs(f,x,y1)))'; %define
14     p = (y2 - y1)';
15
16     %compute the error
```

```
    error = [norm(y2 - exact)];

    iter = 1;                    %set counter for iterations

    while norm(y2 - y1) > tol
        Ai = Ai - ((Ai*g-p)*p'*Ai)/(p'*Ai*g); %compute
        y1 = y2;     %update
        y2 = y1 - (Ai*double(subs(f,x,y1)'))';
        g = (double(subs(f,x,y2) - subs(f,x,y1)))';
        p = (y2 - y1)';

        error = [error, norm(y2 - exact)];

        iter = iter + 1;
    end

    output = [y2 iter];
end
```

(c) Now we are given that the exact solution is $x^* = (0,1)^T$. Using the Newton's and Broyden's method scripts we can compute the error $e_n = \|x_n - x^*\|$ at the k^{th} step.

For Newton's Method we can test the convergence by computing

$$\frac{e_{n+1}}{e_n^p}$$

for each step. Setting $p = 1$, the $\lim_{n\to\infty} \frac{e_{n+1}}{e_n}$ appears to be zero. When setting $p = 2$, the error appears to still be bounded but due to the lack of iteration steps, it is difficult to tell the exact limiting behavior. We expect Newton's method to be at least quadratic and setting $p > 2$ shows divergent behavior.

For Broyden's Method we can test the convergence by computing

$$\frac{e_{n+1}}{e_n^p}$$

for each step. When setting $p = 1$, the $\lim_{n\to\infty} \frac{e_{n+1}}{e_n}$ appears to be zero. But when setting $p = 2$, the limit blows up to infinity. Thus the order of converge is superliner. This aligns with what we expect.

□