Exercise 1. (Component Skill 7.1)

Consider the function $f(x) = 3x^3 - 2x^2 + 3x - 2$ over the interval $0 \leq x \leq 4.$

a. What is the exact root of f in this interval?

By factoring the function $f(x)$, we find that the exact root of the function in the interval $0 \leq x \leq 4$ is $x = \frac{2}{3}$.

b. In what interval is the root guaranteed to be after three iterations of the bisection method?

After three iterations of the bisection method, the root is guaranteed to be in the interval $[0.5, 1]$.

c. Based on your answer to (b), what is the best guess for the root of f after three iterations of the bisection method?

The best guess for the root of the function after three iterations of the bisection method is the midpoint of the interval $[0.5, 1]$, which is 0.75.

d. What is the maximum possible error of your answer to (c)?

The maximum possible error after three iterations is half the length of the final interval. In this case, the maximum possible error is $\frac{0.5}{2} = 0.25$.

e. How many iterations of the bisection method are needed to guarantee that the maximum possible error of the best guess of the root of f is less than 10^{-6}?

The error after n iterations of the bisection method can be expressed as $\frac{(b-a)}{2^n}$, where a and b are the endpoints of the initial interval, and n is the number of iterations. We want this to be less than 10^{-6}.

Setting up the equation $\frac{(b-a)}{2^n} < 10^{-6}$, where $a = 0$ and $b = 4$, we can solve for n:

$\frac{4}{2^n} < 10^{-6}$

$2^n > 4 \times 10^6$

$n > \log_2(4 \times 10^6)$

Therefore, the minimum n that satisfies this inequality is the smallest integer larger than $\log_2(4 \times 10^6)$. Using the fact that $\log_2(x) = \frac{\log(x)}{\log(2)}$ and that $\log$ is the natural logarithm:

$n > \frac{\log(4\times 10^6)}{\log(2)}$

$n >= 22$

Exercise 2. (Component Skills 7.2-7.3)

Consider again the function $f(x) = 3x^3 - 2x^2 + 3x - 2$.

a. Perform one iteration of Newton's method with initial guess $x_0 = 1$.

The formula for Newton's method is $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.

First, we compute the derivative of $f(x)$: $f'(x) = 9x^2 - 4x + 3$.

For $x_0 = 1$, we have $f(x_0) = 3(1)^3 - 2(1)^2 + 3(1) - 2 = 2$ and $f'(x_0) = 9(1)^2 - 4(1) + 3 = 8$.

Thus, the next estimate for the root, x_1, is given by $x_0 - \frac{f(x_0)}{f'(x_0)} = 1 - \frac{2}{8} = 0.75$.

b. A grad student is concerned that there is an initial guess x_0 such that $f'(x_k) = 0$ for some $k \geq 0$. In such a situation, Newton's method will fail. Explain why the grad student has nothing to worry about.

For Newton's method to fail, the derivative of the function at the point of interest would need to be zero, as this would result in division by zero in the iteration formula.

However, if we consider the values of x such that $f'(x) = 0$, we find:

$9x^2 - 4x + 3 = 0$

This is a quadratic equation with no real roots. Therefore, there is no value of x for which $f'(x) = 0$.

This means that Newton's method will not fail due to division by zero for this particular function.

c. Perform one iteration of the secant method with initial guesses $x_0 = 0$ and $x_1 = 1$.

The formula for the secant method is $x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$.

For $x_0 = 0$, we have $f(x_0) = 3(0)^3 - 2(0)^2 + 3(0) - 2 = -2$.

For $x_1 = 1$, we have $f(x_1) = 3(1)^3 - 2(1)^2 + 3(1) - 2 = 2$.

So, the next estimate for the root, x_2, is given by $x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} = 1 - \frac{2(1-0)}{2-(-2)} = 0.5$.

Exercise 3. (Component Skill 7.4)

Consider the function $f(x) = 3x^3 - 2x^2 + 3x - 2$.

a.

The grad student's iteration scheme is defined by $g_1(x) = 3x^3 - 2x^2 + 4x - 2$. The derivative $g_1'(x) = 9x^2 - 4x + 4$. Evaluating $g_1'(x)$ at the root $x = 1$ gives $g_1'(1) = 9 - 4 + 4 = 9$. Because $|g_1'(1)| > 1$, the iteration scheme does not converge for any initial guess x_0 not exactly at the root of f.

b.

The postdoc's iteration scheme is defined by $g_2(x) = -x^3 + \frac{2}{3}x^2 + \frac{2}{3}$. The derivative $g_2'(x) = -3x^2 + \frac{4}{3}x$. Evaluating $g_2'(x)$ at the root $x = 1$ gives $g_2'(1) = -3 + \frac{4}{3} = -\frac{5}{3}$. Because $|g_2'(1)| < 1$, the iteration scheme will converge for all initial guesses sufficiently close to the root of f.

c.

The professor's iteration scheme is defined by $g_3(x) = \left(\frac{2}{3}x^2 - x + \frac{2}{3}\right)^{\frac{1}{3}}$. The derivative $g_3'(x)$ is complex, if its absolute value at the root $x = 1$ is less than $|g_2'(1)| = \frac{5}{3}$, then this iteration scheme will converge faster than the postdoc's for all initial guesses sufficiently close to the root of f.

```
% Defining the function and its derivative
f = @(x) 6 - 3.*x.*(1 + exp(3.*(1-x)));
df = @(x) -3*(1 + exp(3*(1-x))) + 9*x*exp(3*(1-x));

% a. Use the MATLAB built-in function fzero with initial guess 0.1 to
 determine the value of x1.
A1 = fzero(f, 0.1);

% b. Use the MATLAB built-in function fzero with initial guess 1.9 to
 determine the value of x2.
A2 = fzero(f, 1.9);

% c. Bisection Method for 0 # x # 0.5
[A3, A4] = bisectionMethod(f, [0, 0.5], 10^-15);

% d. Bisection Method for 1.5 # x # 2
[A5, A6] = bisectionMethod(f, [1.5, 2], 10^-15);

% e. Newton's Method for x0 = 0.1
[A7, A8] = newtonMethod(f, df, 0.1, 10^-15);

% f. Newton's Method for x0 = 1.9
[A9, A10] = newtonMethod(f, df, 1.9, 10^-15);

% g. Secant Method for x0 = 0.1, x1 = 0.11
[A11, A12] = secantMethod(f, 0.1, 0.11, 10^-15);

% h. Secant Method for x0 = 1.9, x1 = 1.8
[A13, A14] = secantMethod(f, 1.9, 1.8, 10^-15);

% Defining the bisectionMethod function
function [x, N] = bisectionMethod(f, int, thresh)
    x = [int(1) mean(int) int(2)];
    N = 0;
    while x(3)-x(1) >= thresh
        fx = f(x);
        if fx(1)*fx(2) < 0
            x = [x(1) mean([x(1) x(2)]) x(2)];
        elseif fx(2)*fx(3) < 0
            x = [x(2) mean([x(2) x(3)]) x(3)];
        else
            x = [x(2) x(2) x(2)];
        end
        N = N +1;
    end
    x = x(2);
end

% Defining the newtonMethod function
function [x1, N] = newtonMethod(f, df, x0, thresh)
    x1 = x0 - f(x0)/df(x0);
    err = abs(x1-x0);
    N = 1;
    while err >= thresh
        x0 = x1;
        x1 = x0 - f(x0)/df(x0);
        err = abs(x1-x0);
        N = N+1;
    end
end

% Secant Method function
function [x2, N] = secantMethod(f, x0, x1, thresh)
    err = abs(x1-x0);
    N = 0;  % Initialize N to 0
    while err >= thresh
        x2 = x1 - (f(x1)*(x1-x0))/(f(x1)-f(x0)); % Update rule for Secant
 Method
        err = abs(x2-x1);
        x0 = x1;
        x1 = x2;
        N = N+1;
    end
end
```

Published with MATLAB® R2022b