Assignment 3
MATH391

Connor Braun

November 22, 2021

Problem 1

a) Function newton_bisect() for numerically approximating root of some $f(x) = 0$ bracketed by some interval $[a, b]$. The function combines the bisection and Newton's methods. Function specifications are as described in the assignment.

```
function [x, fx, nf, af, bf] = newton_bisect(a, b, fname, tolx, nmax)

    % Fix arguments pro re nada
    if a > b
        a_c = a;
        a = b;
        b = a_c;
    end
    tolx = abs(tolx);
    nmax = round(abs(nmax));
    fname_p = matlabFunction(diff(sym(fname)));

    % Variable initialization
    x = a;
    nf = 2; % Two function evaluations during initialization
    af = a;
    bf = b;

    % Evaluate functions at endpoints
    fa = fname(a);
    fb = fname(b);
    fx = fa;

    % Return condition: either initial endpoint is a root
    if fa == 0
        disp('newton_bisect return condition: a is a root')
        return
    elseif fb == 0
        x = b;
        disp('newton_bisect return condition: b is a root')
        return
    else % Assert the guaranteed existence of a root on (a,b)
        assert(fa*fb <= 0, 'Root may not exist; ensure that the product of function evaluated at
    end
```

```
    % Root finding algorithm main loop
    for i = 1:nmax

        % Return condition: interval length less than allowed tolerance
        if abs(b - a) <= tolx
            disp('newton_bisect return condition: |b - a| < tolx')
            return
        end

        fx_p = fname_p(x); % Compute derivative

        % First, try iterating using Newton's algorithm
        x = x - fx/fx_p;
        fx = fname(x);

        if x >= a && x <= b % If the next iterate is on the current interval
            if fx*fa < 0 % If (a,x) brackets the root
                b = x;
            else % Otherwise, (x,b) must bracket the root
                a = x;
            end

        % Next, if the iterate left the interval, resort to bisection
        else
            x = (a + b)/2;
            fx = fname(x);

            if fx*fa < 0 % If (a,x) brackets the root
                b = x;
            elseif fx*fb < 0 % If (x,b) brackets the root
                a = x;
            end
        end
        fa = fname(a);
        fb = fname(b);
        af = a;
        bf = b;
        nf = nf + (i - 1)*5; % 5 function evaluations per iteration
    end
end
```

Problem 2

a) Let $f(x) = x^3 - a$ for some $a \in \mathbb{R}$ and consider the equation $x^3 - a = 0$. Write down the secant algorithm for $f(x) = 0$.

Let $x_1 = b$ and $x_2 = c$ be real numbers. The secant algorithm generates a sequence of iterates $\{x_n\}_{n\geq 1}$ approximating $r \in \mathbb{R}$ such that $f(r) = 0$, where r is called a root of $f(x)$. The n^{th} iterate x_n where $x \geq 3$ is given by

$$x_n = x_{n-1} - f(x_{n-1})\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}.$$

We can simplify this equation as follows

$$\begin{aligned} x_n &= x_{n-1} - (x_{n-1}^3 - a)\frac{x_{n-1} - x_{n-2}}{(x_{n-1}^3 - a) - (x_{n-2}^3 - a)} \\ &= x_{n-1} - \frac{(x_{n-1}^3 - a)(x_{n-1} - x_{n-2})}{x_{n-1}^3 - x_{n-2}^3} \\ &= \frac{x_{n-1}(x_{n-1}^3 - x_{n-2}^3) - (x_{n-1}^3 - a)(x_{n-1} - x_{n-2})}{x_{n-1}^3 - x_{n-2}^3}. \end{aligned}$$

However, $x_{n-1}^3 - x_{n-2}^3 = (x_{n-1} - x_{n-2})(x_{n-1}^2 + x_{n-1}x_{n-2} + x_{n-2}^2)$, so we can expand the expression above into

$$\begin{aligned} x_n &= \frac{x_{n-1}(x_{n-1} - x_{n-2})(x_{n-1}^2 + x_{n-1}x_{n-2} + x_{n-2}^2) - (x_{n-1}^3 - a)(x_{n-1} - x_{n-2})}{(x_{n-1} - x_{n-2})(x_{n-1}^2 + x_{n-1}x_{n-2} + x_{n-2}^2)} \\ &= \frac{x_{n-1}(x_{n-1}^2 + x_{n-1}x_{n-2} + x_{n-2}^2) - x_{n-1}^3 + a}{x_{n-1}^2 + x_{n-1}x_{n-2} + x_{n-2}^2} \\ &= \frac{x_{n-1}^3 + x_{n-1}^2x_{n-2} + x_{n-2}^2x_{n-1} - x_{n-1}^3 + a}{x_{n-1}^2 + x_{n-1}x_{n-2} + x_{n-2}^2} \\ &= \frac{x_{n-1}^2x_{n-2} + x_{n-2}^2x_{n-1} + a}{x_{n-1}^2 + x_{n-1}x_{n-2} + x_{n-2}^2} \\ &= \frac{x_{n-1}x_{n-2}(x_{n-2} + x_{n-1}) + a}{x_{n-1}^2 + x_{n-1}x_{n-2} + x_{n-2}^2}. \end{aligned}$$

Finally, we can write the secant algorithm for $x^3 - a = 0$ as

$$\begin{cases} x_1 = b \\ x_2 = c \\ x_n = \frac{x_{n-1}x_{n-2}(x_{n-2}+x_{n-1})+a}{x_{n-1}^2+x_{n-1}x_{n-2}+x_{n-2}^2}, \qquad n \geq 3. \end{cases}$$

b) Let $a = 3$, $b = 1$ and $c = 2$, then use the secant algorithm defined in a) to compute the first 10 iterates. Comment on the rate of convergence of the sequence.

Table 1. The sequence of approximations of r such that $f(r) = r^3 - 3 = 0$ generated by the secant algorithm. The computation for x_3 using the result in a) is shown, and the rest were computed to 19 digits using the MATLAB code shown below the table. We know that $r = 3^{\frac{1}{3}} = 1.442249570307408382\ldots$, and so also tabulate k_n, where k_n is the number of consecutive correct digits in x_n.

n	x_n	k_n
1	1	-
2	2	-
3	$x_3 = \frac{2(3)+3}{4+2+1} = \frac{9}{7} = 1.285714285714285714\ldots$	1
4	$1.392059553349875931\ldots$	1
5	$1.448265350468348823\ldots$	3
6	$1.442035892099417439\ldots$	4
7	$1.442248681418064631\ldots$	6
8	$1.442249570439115908\ldots$	10
9	$1.442249570307408301\ldots$	17
10	$1.442249570307408382\ldots$	19
11	$1.442249570307408382\ldots$	19
12	$1.442249570307408382\ldots$	19

MATLAB code to generate results:

```
digits(19)
a = 3;
xn = [vpa(1); vpa(2)];
for i = numel(xn) + 1:numel(xn) + 10
    xi = (xn(i - 1)*xn(i - 2)*(xn(i - 2) + xn(i - 1)) + a)/...
        (xn(i - 1)^2 + xn(i - 1)*xn(i - 2) + xn(i - 2)^2);
    xn(i) = xi;
end
xn
```

The sequence appears to have a faster than linear rate of convergence, since the number of correct decimal places increases nonlinearly across the tabulated sequence of iterations. We next endeavor to approximate the rate of convergence. To motivate the approximation, first suppose that p is the rate of convergence of the sequence $\{x_n\}_{n\geq 1}$ with $x_1 = b$ and $x_2 = c$ and x_n, $n \geq 3$ as defined in part a). Then there exists some $d > 0$ such that

$$\lim_{n\to+\infty} \frac{|x_{n+1} - r|}{|x_n - r|^p} = d$$
$$\Rightarrow \frac{|x_{n+1} - r|}{|x_n - r|^p} \approx d$$

We can now manipulate this expression to reveal an approximation for p, which we will call p_n where $n \geq 3$.

$$|x_{n+1} - r| \approx d|x_n - r|^p$$
$$\Rightarrow \frac{|x_{n+1} - r|}{|x_n - r|} \approx \frac{d|x_n - r|^p}{d|x_{n-1} - r|^p}$$
$$\Rightarrow \ln\left(\frac{|x_{n+1} - r|}{|x_n - r|}\right) \approx \ln\left(\frac{|x_n - r|^p}{|x_{n-1} - r|^p}\right)$$

$$\Rightarrow p_{n+1} = \frac{\ln\left(\frac{|x_{n+1}-r|}{|x_n-r|}\right)}{\ln\left(\frac{|x_n-r|}{|x_{n-1}-r|}\right)} \approx p$$

This approximation, while true, is numerically unstable since, as seen in table 1, the difference between subsequent iterates rapidly approaches zero, which results in division by zero when precision is limited. To avoid this, we most seriously consider $5 \leq n \leq 8$, since the detectable difference between iterates diminishes beyond this point (indeed, we gain 7 correct decimal places for $n = 9$, 2 for $n = 10$, and 0 thereafter).

Table 2. Approximated values of p using the last 3 elements of $\{x_i\}$ for $i \leq n$, where sequence $\{x_i\}$ is generated using the secant algorithm as defined in a) to approximate root r of $x^3 - 3 = 0$.

n	p_n
5	1.86505038093259
6	1.57331139189452
7	1.64253909945344
8	1.60830639153794
9	2.17371661563057
10	−0.25350888214223
11	$8.25611634512994 \times 10^{-12}$
12	−1.00000000004011
13	0
14	NaN

For $n < 9$ (i.e., before the onset of numerical instability as described above) we have that $p \approx 1.6 > 1$, so the rate of convergence of the sequence $\{x_n\}_{n\geq 1}$ generated by the secant algorithm defined in a) to find r for $a = 3$ appears to be superlinear as per the finding that we appear to have $1 < p < 2$.

Problem 3

a) Prove that if $g(x) \in C^0[a, b]$ and if $\forall_x \in [a, b]$, $a \leq g(x) \leq b$, then $g(x)$ has at least on fixed point in $[a, b]$. Justify whether or not the fixed point is necessarily unique.

Proof. Suppose that $g(x)$ is a continuous function on $[a, b]$ for some $a, b \in \mathbb{R}$. Suppose also that $\forall_x \in [a, b]$, we have that $a \leq g(x) \leq b$. First, we define a new function $f(x) = g(x) - x$, which is itself continuous on $[a, b]$ since both $g(x)$ and x are. Next, we can show that $f(x)$ has a root r on $[a, b]$.

$$\begin{aligned} &a \leq g(a) \leq b, \qquad \text{since } a \leq a \leq b \\ &\Rightarrow 0 \leq g(a) - a \leq b - a \\ &\Rightarrow 0 \leq f(a). \end{aligned}$$

Furthermore,

$$\begin{aligned} &a \leq g(b) \leq b, \qquad \text{since } a \leq b \leq b \\ &\Rightarrow a - b \leq g(b) - b \leq 0 \\ &\Rightarrow f(b) \leq 0. \end{aligned}$$

Now, since $f(x)$ is continuous on $[a, b]$ and $f(b) \leq 0 \leq f(a)$, by the intermediate value theorem we guarantee the existence of some $\xi \in [a, b]$ such that $f(\xi) = 0$. Let r be this value so that $f(r) = 0$. Then r is a fixed point of $g(x)$, since

$$f(r) = g(r) - r$$

$$\Rightarrow g(r) = r$$

where r is guaranteed to exist under the conditions provided.

Such a fixed point need not be unique, however.

Proof. Let $g(x) = x$, $a = 0$, $b = 1$ and $c \in [a, b]$. Then $g(x) \in C^0[a, b]$ and $a \leq c \leq b \Rightarrow a \leq g(c) \leq b$. Despite this, we have both that $g(0) = 0$ and $g(1) = 1$, so both $0, 1 \in [a, b]$ are fixed points of $g(x)$.

Problem 4

a) Consider the equation $f(x) = 0$, where $f(x) = e^x - ex$. Show that $f(x)$ has a unique root r and find it's multiplicity p.

First we seek the critical points of $f(x)$.

$$\begin{aligned} f'(x) &= e^x - e = 0 \\ \Rightarrow e^x &= e \\ \Rightarrow x\ln(e) &= ln(e) \\ \Rightarrow x &= 1. \end{aligned}$$

Hence $x = 1$ is the only critical point of $f(x)$, so computing $f'(x_a)$ for $x_a = 1 + 1 = 2 > 1$ and $f'(x_b)$ for $x_b = 1 - 1 = 0 < 1$ shows us that

$$\begin{aligned} f'(x_a) = f'(2) = e^2 - e = e(e-1) > 0 \\ f'(x_b) = f'(0) = e^0 - e = 1 - e < 0. \end{aligned}$$

From this, $f(x)$ is decreasing for all $x < 1$, and increasing for all $x > 1$. Furthermore, $f(1) = e^1 - e = 0$, so $r = 1$ is the only root of $f(x) = 0$. Now, since $r = 1$ is a root of $f(x)$, for some unknown function $h(x)$ we can write

$$\begin{aligned} f(x) &= (x-1)h(x) \\ \Rightarrow f'(x) &= h(x) + (x-1)h'(x) \\ \Rightarrow f'(1) &= e^1 - e = 0 = h(1) \end{aligned}$$

where now $r = 1$ is also a root of $h(x)$. Then for some function $g(x)$ we can write

$$\begin{aligned} f(x) &= (x-1)^2 g(x) \\ \Rightarrow f'(x) &= 2(x-1)g(x) + (x-1)^2 g'(x) \\ \Rightarrow f''(x) &= 2g(x) + 4(x-1)g'(x) + (x-1)^2 g''(x) \\ \Rightarrow f''(1) &= e^1 = 2g(1) \\ \Rightarrow g(1) &= e/2 \neq 0 \end{aligned}$$

So 1 is not a root of $g(x) = 0$ and we can write that

$$f(x) = (x-1)^2 g(x)$$

implying that root $r = 1$ of $f(x) = 0$ has multiplicity 2.

b) Let $\{x_n\}_{n\geq 1}$ be the Newton sequence with initial guess $x_1 = 0$. Compute 20 iterates and and verify the linear convergence of the sequence.

Table 3. First 20 iterates of the Newton sequence as described above. The error $e_n = |x_n - 1|$ and approximation of the rate of convergence k_n are also tabulated, where the latter uses the same approximation technique as was used to compute p_n in table 2. Note that the n^{th} approximation of the rate of convergence, k_n uses error of iterates n, $n-1$ and $n-2$. All calculations were done in MATLAB with 15 points of precision. The code used to generate this data is included below the table.

n	x_n	e_n	k_n
1	0	1	–
2	0.5819767068693262	0.418023293130674	–
3	0.805508075137019	0.194491924862981	0.877242161526464
4	0.905904311079007	0.0940956889209927	0.94894036925479
5	0.953689879905671	0.0463101200943289	0.976412058588285
6	0.977023652500333	0.022976347499667	0.988636008307362
7	0.988555818575133	0.0114441814248673	0.994419554185397
8	0.994288823371104	0.00571117662889575	0.997234473252792
9	0.997147129812271	0.00285287018772873	0.998623327220866
10	0.998574243145011	0.00142575685498925	0.999313175954041
11	0.999287290970944	0.00071270902905618	0.999656964676004
12	0.99964368781498	0.00035631218502008	0.999828576660796
13	0.99982185448684	0.00017814551315997 7	0.999914307580552
14	0.999910929887816	0.0000890701121841753	0.999957161749242
15	0.999955465603578	0.0000445343964224909	0.999978539377947
16	0.999977732967117	0.0000222670328825503	0.999989342051222
17	0.999988866519551	0.0000111334804494545	0.999993952227437
18	0.999994433260763	0.00000556673923679529	0.999995592183484
19	0.999997216609442	0.00000278339055759247	0.999988890662244
20	0.999998608271376	0.00000139172862412273	0.999976286169489

MATLAB code to generate results:

```
n = 20;
xn = zeros(n, 1);
en = ones(n, 1);
kn = zeros(n, 1);
f_x = @(x) exp(x) - exp(1)*x;
f_x_p = @(x) exp(x) - exp(1);

for i = 2:n
    xn(i) = xn(i - 1) - f_x(xn(i - 1))/f_x_p(xn(i - 1));
    en(i) = abs(xn(i) - 1);
    if i >= 3
        kn(i) = log(en(i)/en(i - 1))/log(en(i - 1)/en(i - 2));
    end
end
disp(xn) % Newton sequence
disp(en) % Absolute error of Newton sequence
disp(kn) % Approximation of rate of convergence using n, n-1 and n-2
```

The results of table 3 show three key trends. Firstly, x_n appears to approach 1 as n increases, so the algorithm appears to be converging on the root r. Equivalently, the error e_n appears to be approaching zero. Finally,

letting k be the rate of convergence of the Newton sequence of interest and $d > 0$, by definition we have

$$\lim_{n\to+\infty} \frac{|x_{n+1} - r|}{|x_n - r|^k} = d.$$

We found in problem 2.b that k can be approximated by the formula

$$k_n = \frac{\ln\left(\frac{|x_{n+1}-r|}{|x_n-r|}\right)}{\ln\left(\frac{|x_n-r|}{|x_{n-1}-r|}\right)} \approx k$$

Where $\lim_{n\to+\infty} k_n = k$. In table 3, k_n is very nearly 1 so we argue that $k \approx 1$. By these three trends we have compelling reason to believe that the Newton sequence $\{x_n\}_{n\geq 1}$ is converging to $r = 1$ with a linear rate of convergence.

c) Show that the rate of convergence of $\{x_n\}_{n\geq 1}$ is, in fact, linear.

Proof. Let $g(x)$ be a continuous function with continuous derivatives up to order two and a unique root r so that $g(r) = 0$. Suppose also that this root has multiplicity 2. Then we know that for some at least twice differentiable function $h(x)$, $g(x)$ can be written as

$$\begin{aligned} g(x) &= (x-r)^2 h(x) \\ \Rightarrow g'(x) &= 2(x-r)h(x) + (x-r)^2 h'(x) \\ \Rightarrow g''(x) &= 2h(x) + 4(x-r)h'(x) + (x-r)^2 h''(x) \\ \Rightarrow g''(r) &= 2h(r). \end{aligned}$$

Where in particular we have that $g(r) = 0$, $g'(r) = 0$, $g''(r) = 2h(r) \neq 0$. Next, define the sequence $\{x_n\}_{n\geq 1}$ to be the one generated by the Newton algorithm such that

$$\begin{cases} x_1 = a \\ x_{n+1} = x_n - \frac{g(x_n)}{g'(x_n)}, \qquad n \geq 1. \end{cases}$$

Finally, suppose we know that in the limit the sequence converges to root r. We seek to show that this convergence is linear, which by the definition of linear rate of convergence would imply that

$$\lim_{n\to+\infty} \frac{|x_{n+1} - r|}{|x_n - r|} = k, \qquad k > 0.$$

By Newton's algorithm, we have that

$$\begin{aligned} x_{n+1} &= x_n - \frac{g(x_n)}{g'(x_n)} \\ \Rightarrow x_{n+1} - r &= (x_n - r) - \frac{g(r) - g(x_n)}{g'(r) - g'(x_n)}. \end{aligned}$$

Next, by substituting the Taylor expansion of $g(r)$ and $g'(r)$, we derive a new expression for $x_{n+1} - r$.

$$\begin{aligned} x_{n+1} - r &= (x_n - r) - \frac{(r - x_n)g'(x_n) + \frac{1}{2}(r - x_n)^2 g''(\theta_1)}{(r - x_n)g''(\theta_2)} \\ &= (x_n - r) - \left(\frac{g'(x_n) + \frac{1}{2}(r - x_n)g''(\theta_1)}{g''(\theta_2)}\right) \end{aligned}$$

where θ_1, $\theta_2 \in I[x_n, r]$. Note that $lim_{n\to+\infty} I[x_n, r] = \{r\}$, so in the limit, $\theta_1, \theta_2 \in \{r\} \Rightarrow \theta_1, \theta_2 = r$. Then the desired result can be shown by

$$\frac{x_{n+1} - r}{x_n - r} = 1 - \left(\frac{g'(x_n) + \frac{1}{2}(r - x_n)g''(\theta_1)}{g''(\theta_2)(x_n - r)}\right)$$

$$
\begin{aligned}
&= 1 + \frac{g''(\theta_1)}{2g''(\theta_2)} - \frac{g'(x_n)}{g''(\theta_2)(x_n - r)} \\
&= 1 + \frac{g''(\theta_1)}{2g''(\theta_2)} - \frac{2(x_n - r)h(x_n) + (x_n - r)^2 h'(x_n)}{g''(\theta_2)(x_n - r)} \\
&= 1 + \frac{g''(\theta_1)}{2g''(\theta_2)} - \frac{2h(x_n) + (x_n - r)h'(x_n)}{g''(\theta_2)} \\
\Rightarrow \lim_{n \to +\infty} \left| \frac{x_{n+1} - r}{x_n - r} \right| &= \left| 1 + \frac{g''(r)}{2g''(r)} - \frac{2h(r)}{g''(r)} \right| \\
&= \left| 1 + \frac{1}{2} - \frac{g''(r)}{g''(r)} \right| \\
&= \frac{1}{2} > 0.
\end{aligned}
$$

So by definition, the rate of convergence of sequence $\{x_n\}_{n \geq 1}$ is linear.

Note that this theorem applies to $f(x) = e^x - ex$ from part a) since $f(x)$ has two continuous derivatives and a unique root of multiplicity 2.

d)Let $\{z_n\}_{n \geq 1}$ be the sequence generated by the modified Newton's algorithm

$$
z_1 = 0, \qquad z_{n+1} = z_n - p \frac{f(z_n)}{f'(z_n)}, \qquad n \geq 1
$$

where p is the multiplicity of the root r of $f(z)$. Verify that the convergence of $\{z_n\}_{n \geq 1}$ is quadratic.

To verify this, we consider the original equation $f(z) = e^z - ez$, which has root $r = 1$ with multiplicity $p = 2$. By precisely the same method as in 4.b, we numerically approximate the rate of convergence k of $\{z_n\}_{n \geq 1}$ by the formula

$$
k_{n+1} = \frac{\ln\left(\frac{|z_{n+1} - r|}{|z_n - r|}\right)}{\ln\left(\frac{|z_n - r|}{|z_{n-1} - r|}\right)} \approx k.
$$

Table 4. Approximation of rate of convergence $k_n \approx k$ of $\{z_n\}_{n \geq 1}$. MATLAB code used to generate the results is given beneath the table.

n	k_n
1	–
2	–
3	1.9911702809902
4	1.99987570511093
5	1.84752119194048
6	−1.21522541247082
7	−0.714058334947209
8	0
9	NaN

MATLAB code used to generate results

```
format longg
n = 20;
xn = zeros(n, 1);
en = ones(n, 1);
```

```
kn = zeros(n, 1);
f_x = @(x) exp(x) - exp(1)*x;
f_x_p = @(x) exp(x) - exp(1);

for i = 2:n
    xn(i) = xn(i - 1) - 2*f_x(xn(i - 1))/f_x_p(xn(i - 1));
    en(i) = abs(xn(i) - 1);
    if i >= 3
        kn(i) = log(en(i)/en(i - 1))/log(en(i - 1)/en(i - 2));
    end
end
disp(xn) % Newton sequence
disp(en) % Absolute error of Newton sequence
disp(kn) % Approximation of rate of convergence using n, n-1 and n-2
```

As was the case in 2.b, approximations k_n are numerically unstable when working with limited precision – in this case the approximation includes division by zero after only 9 iterations. By iteration 5 the error decreases from $\sim 10^{-6}$ to $\sim 10^{-12}$, and remains relatively constant thereafter. For this reason, we consider only the values above the second horizontal line which are computed before the onset of numerical istability. Interestingly, these approximations indicate that $k \approx 1.999$, suggesting that $\{z_n\}_{n\geq 1}$ is exhibiting quadratic convergence.

e)Show that the rate of convergence of $\{z_n\}_{n\geq 1}$ is, in fact, quadratic.

Proof. Let $g(z)$ be an a continuous function with continuous derivatives up to at least order three and a unique root r so that $g(r) = 0$ and $g'''(r) \neq 0$. Suppose also that root r has multiplicity 2. Then we know that for some at least thrice differentiable function $h(z)$, $g(z)$ can be written as

$$\begin{aligned} g(z) &= (z-r)^2 h(z) \\ \Rightarrow g'(z) &= 2(z-r)h(z) + (z-r)^2 h'(z) \\ \Rightarrow g''(z) &= 2h(z) + 4(z-r)h'(z) + (z-r)^2 h''(z) \\ \Rightarrow g'''(z) &= 6h'(z) + 6(z-r)h''(z) + (z-r)^2 h'''(z) \\ \Rightarrow g''(r) &= 2h(r) \qquad \text{and} \qquad g'''(r) = 6h'(r) \end{aligned}$$

Where in particular we have that $g(r) = 0$, $g'(r) = 0$, $g''(r) = 2h(r) \neq 0$ and $g'''(r) = 6h'(r) \neq 0$. Next, define the sequence $\{z_n\}_{n\geq 1}$ to be the one generated by the modified Newton algorithm such that

$$\begin{cases} z_1 = a \\ z_{n+1} = z_n - 2\frac{g(z_n)}{g'(z_n)}, \qquad n \geq 1. \end{cases}$$

Finally, suppose we know that in the limit the sequence converges to root r. We seek to show that this convergence is quadratic, which by the definition of quadratic rate of convergence would imply that

$$\lim_{n\to+\infty} \frac{|z_{n+1} - r|}{|z_n - r|^2} = k, \qquad k > 0.$$

By the modified Newton's algorithm, we have that

$$\begin{aligned} z_{n+1} &= z_n - 2\frac{g(z_n)}{g'(z_n)} \\ \Rightarrow z_{n+1} - r &= (z_n - r) - 2\frac{g(r) - g(z_n)}{g'(r) - g'(z_n)}. \end{aligned}$$

Next, by substituting the Taylor expansion of $g(r)$ and $g'(r)$, we derive a new expression for $z_{n+1} - r$.

$$z_{n+1} - r = (z_n - r) - 2\frac{(r - z_n)g'(z_n) + \frac{1}{2}(r - z_n)^2 g''(\theta_1)}{(r - z_n)g''(\theta_2)}$$

$$= (z_n - r) - 2\left(\frac{g'(z_n) + \frac{1}{2}(r - z_n)g''(\theta_1)}{g''(\theta_2)}\right)$$

where θ_1, $\theta_2 \in I[z_n, r]$. Note that $lim_{n\to+\infty} I[x_n, r] = \{r\}$, so in the limit, $\theta_1, \theta_2 \in \{r\} \Rightarrow \theta_1, \theta_2 = r$. Then the desired result can be shown by

$$\begin{aligned}
\frac{z_{n+1} - r}{(z_n - r)^2} &= \frac{1}{z_n - r} - 2\left(\frac{g'(z_n) + \frac{1}{2}(r - z_n)g''(\theta_1)}{g''(\theta_2)(z_n - r)^2}\right) \\
&= \frac{1}{z_n - r} - 2\left(\frac{2(z_n - r)h(z_n) + (z_n - r)^2 h'(z_n) + \frac{1}{2}(r - z_n)g''(\theta_1)}{g''(\theta_2)(z_n - r)^2}\right) \\
&= \frac{1}{z_n - r} - 2\left(\frac{2h(z_n) + (z_n - r)h'(z_n) - \frac{1}{2}g''(\theta_1)}{g''(\theta_2)(z_n - r)}\right) \\
&= \frac{g''(\theta_2) + g''(\theta_1) - 4h(z_n) - 2(z_n - r)h'(z_n)}{g''(\theta_2)(z_n - r)} \\
\Rightarrow \lim_{n\to+\infty}\left|\frac{z_{n+1} - r}{(z_n - r)^2}\right| &= \left|\frac{2g''(r) - 4h(r) - 2(r - r)h'(r)}{g''(r)(r - r)}\right| \\
&= \left|\frac{2g''(r) - 2g''(r) - 2(r - r)h'(r)}{g''(r)(r - r)}\right| \\
&= \left|\frac{-2(r - r)h'(r)}{(r - r)g''(r)}\right| \\
&= \left|\frac{-2h'(r)}{g''(r)}\right| \\
&= 2\left|\frac{h'(r)}{g''(r)}\right| = \frac{1}{3}\left|\frac{g'''(r)}{g''(r)}\right| > 0, \qquad \text{since } g'''(r) \neq 0 \text{ and } g''(r) \neq 0.
\end{aligned}$$

Therefore, we have that $\{z_n\}_{n\geq 1}$ converges to root r of $g(z) = 0$ with quadratic order of convergence.

Note that this theorem applies to $f(x) = e^x - ex$ from part a) since $f(x)$ has three continuous derivatives and a unique root r of multiplicity 2 with $f'''(r) = f'''(1) = e \neq 0$.