MATH 151A - Numerical Methods - Homework 2

Darren Tsang, Discussion 1B

Produced on Wednesday, Jul. 08 2020 @ 08:39:11 PM

Please note that all the code the functions fixedPoint(), newtonMethod(), and secantMethod() are shown at the end of this document.

Question #1(ab)

$$f(x) = x^4 + 2x^2 - x - 3 \tag{1}$$

Part A

Setting (1) to 0 and doing some manipulation:

$$x^4 + 2x^2 - x - 3 = 0 \longrightarrow x^4 = x + 3 - 2x^2 \longrightarrow x = (x + 3 - 2x^2)^{1/4}$$

We obtain $g_1(x) = (x + 3 - 2x^2)^{1/4}$, as desired.

Part B

Setting (1) to 0 and doing some manipulation:

$$x^4 + 2x^2 - x - 3 = 0 \longrightarrow 2x^2 = x + 3 - x^4 \longrightarrow x^2 = \frac{x + 3 - x^4}{2} \longrightarrow x = \left(\frac{x + 3 - x^4}{2}\right)^{1/2}$$

We obtain $g_2(x) = (\frac{x+3-x^4}{2})^{1/2}$, as desired.

Question #2

Part A

```
g1 <- function(x){ (x + 3 - 2*x*x)^.25 }
g2 <- function(x){ ((x + 3 - x^4) / 2)^.5 }
```

Here are the results when applying the Fixed Point Iteration on $g_1(x) = (x + 3 - 2x^2)^{1/4}$, with $p_0 = 1$:

```
fixedPoint(f = g1, initial = 1, iter = 4, tol = -Inf)
```

```
##      n      p_n |f(p_n) - p_n|
## [1,] 0 1.000000     0.18920712
## [2,] 1 1.189207     0.10914936
## [3,] 2 1.080058     0.06961368
## [4,] 3 1.149671     0.04185090
## [5,] 4 1.107821     0.02611175
```

Here are the results when applying the Fixed Point Iteration on $g_2(x) = (\frac{x+3-x^4}{2})^{1/2}$, with $p_0 = 1$:

```
fixedPoint(f = g2, initial = 1, iter = 4, tol = -Inf)
```

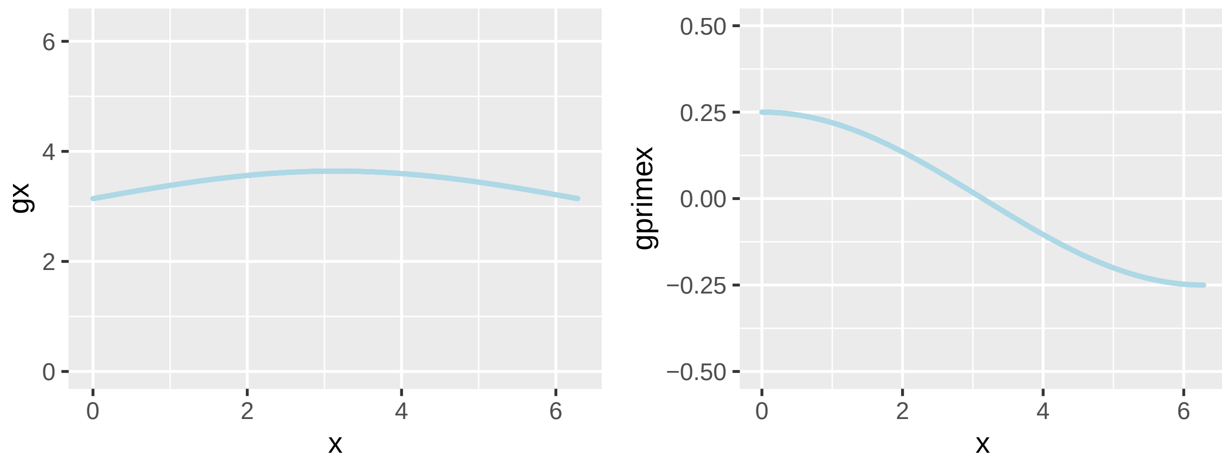
```
##      n       p_n |f(p_n) - p_n|
## [1,] 0 1.0000000      0.2247449
## [2,] 1 1.2247449      0.2310787
## [3,] 2 0.9936662      0.2349025
## [4,] 3 1.2285686      0.2410622
## [5,] 4 0.9875064      0.2446770
```

Part B

Using $g_1(x)$ is much better because the absolute error is actually getting smaller, as compared to using $g_2(x)$, where the absolute error is increasing.

Question #7

Below are the plots of $g(x) = \pi + \frac{1}{2}sin(\frac{x}{2})$ and $g'(x) = \frac{1}{4}cos(\frac{x}{2})$. We notice that $g \in C[0, 2\pi]$ such that $g(x) \in [0, 2\pi], \forall x \in [0, 2\pi]$. Furthermore, $|g'(x)| \leq k = 1/4 < 1, \forall x \in (0, 2\pi)$. Then, for any $p_0 \in [0, 2\pi]$, the sequence defined by $p_n = g(p_{n-1}), n \geq 1$ will converge to the fixed point p in $[0, 2\pi]$.




```r
g7 <- function(x){ pi + .5*sin(x/2) }
```

Here are the results when applying the Fixed Point Iteration method on $g(x) = \pi + \frac{1}{2}sin(\frac{x}{2})$, with $p_0 = \pi$:

```r
fixedPoint(f = g7, initial = pi, iter = Inf, tol = 10^-2)
```

```
##      n      p_n |f(p_n) - p_n|
## [1,] 0 3.141593    0.500000000
## [2,] 1 3.641593    0.015543789
## [3,] 2 3.626049    0.000946758
```

From class, we know that

$$|p_n - p| \leq k^n max(p_0 - a, b - p_0), \tag{1}$$

and

$$|p_n - p| \leq \frac{k^n}{1-k}|p_1 - p|, \forall n \geq 1 \tag{2}$$

Using (1), we solve the following:

$$k^n max(p_0 - a, b - p_0) = (1/4)^n max(\pi - 0, 2\pi - pi) = \pi(1/4)^n = 10^{-2} \longrightarrow n = log_{1/4}(\frac{10^{-2}}{\pi})$$

Using (2), we solve the following:

$$\frac{k^n}{1-k}|p_1 - p| = \frac{(1/4)^n}{1-(1/4)}|3.64159 - 3.14159| = \frac{(1/4)^n}{(3/4)}|1/2| = \frac{(1/4)^n}{3/2} = 10^-2 \longrightarrow n = log_{1/4}(10^{-2}(3/2))$$

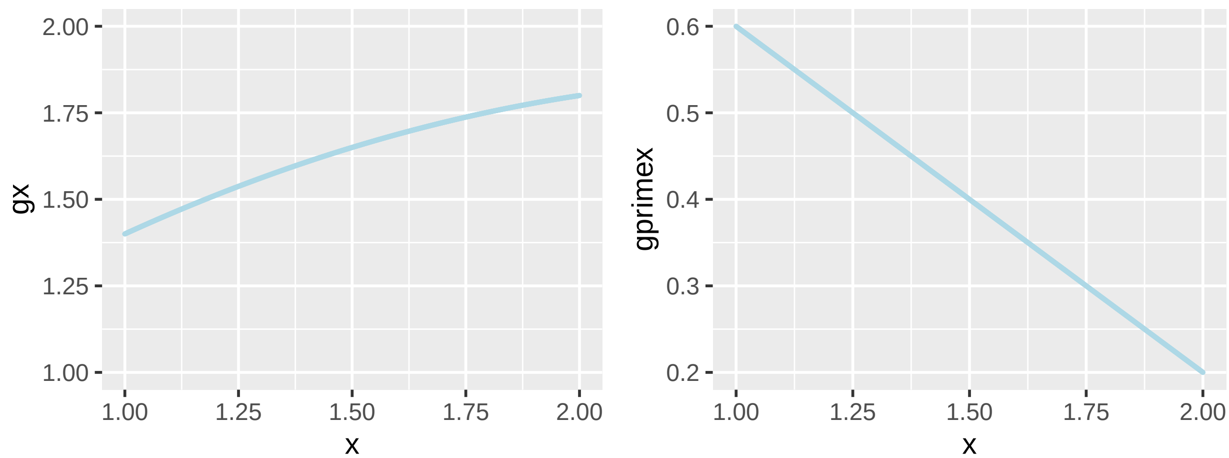
Thus, we can say:

$$n = ceil[min(log_{1/4}(\frac{10^{-2}}{\pi}), log_{1/4}(10^{-2}(3/2))] = ceil[min(4.14768, 3.02945)] = 4$$

We notice that when actually running the Fixed Point algorithm, $|p_2 - p| \leq 10^{-2}$, which means it took less iterations than the bound (as expected).

Question #9

We are trying to find an approximation for $\sqrt{3}$. We can define $f(x) = x^2 - 3$, but we could not run the Fixed Point Iteration method on f. Thus, we define $g(x) = x + cf(x) = x + c(x^2 - 3)$, where c is some constant. We let $c = -\frac{1}{5}$. Below are the graphs of $g(x) = x - \frac{1}{5}(x^2 - 3)$ and $g'(x) = 1 - \frac{2}{5}x$ for $x \in [1, 2]$. We notice that $g(x) \in [1, 2], \forall x \in [1, 2]$ and $|g'(x)| \leq k = .7 < 1, \forall x \in (1, 2)$. Thus, $g(x)$ satisfies the Fixed-Point Theorem over the interval $[1, 2]$.




```
g9 <- function(x){ x - .2*(x^2 - 3) }
```

Here are the results when applying Fixed Point Iteration method to $g(x) = x - \frac{1}{5}(x^2 - 3)$, with $p_0 = 1$:

```
fixedPoint(f = g9, initial = 1, iter = Inf, tol = 10^-4)
```

```
##       n      p_n |f(p_n) - p_n|
##  [1,] 0 1.000000   4.000000e-01
##  [2,] 1 1.400000   2.080000e-01
##  [3,] 2 1.608000   8.286720e-02
##  [4,] 3 1.690867   2.819362e-02
##  [5,] 4 1.719061   8.965978e-03
##  [6,] 5 1.728027   2.784676e-03
##  [7,] 6 1.730811   8.583271e-04
##  [8,] 7 1.731670   2.639388e-04
##  [9,] 8 1.731934   8.110292e-05
```

From Homework 1, Question 5, we obtained an approximation of $\sqrt{3}$ with a tolerance of 10^{-2} (1.731995) after 14 iterations of the Bisection Method. As we can see above, we obtained an approximation of $\sqrt{3}$ with a tolerance of 10^{-2} (1.731934) after 8 iterations of the Fixed Point Iteration method. It is clear that the Fixed Point Iteration method was faster in this particular situation. Also, from a calculator $\sqrt{3} = 1.732050808$. This means that the result from the Fixed Point Iteration method was slightly closer to the real value of $\sqrt{3}$.

Question 2.3, #2

```r
f2 <- function(x){ -(x^3) - cos(x) }
f2prime <- function(x){ -3*x^2 + sin(x) }
```

Here are the results when applying Newton's Method on $f(x) = -x^3 - cos(x)$, with $p_0 = -1$:

```r
newtonMethod(f = f2, fprime = f2prime, initial = -1, iter = 2, tol = -Inf)
```

```
##      n        p_n     |f(p_n)|
## [1,] 0 -1.0000000 0.4596976941
## [2,] 1 -0.8803329 0.0453511546
## [3,] 2 -0.8656842 0.0006323134
```

Using $p_0 = 0$ does not work because $f'(0) = -3(0^2) + sin(0) = 0$, which leads to division by 0.

Question 2.3, #4(a)

```r
f4 <- function(x){ -(x^3) - cos(x) }
```

Here are the results when applying Secant Method on $f(x) = -x^3 - cox(x)$, with $p_0 = -1, p_1 = 0$:

```r
secantMethod(f = f4, initials = c(-1, 0), iter = 3, tol = -Inf)
```

```
##      n        p_n  |f(p_n)|
## [1,] 0 -1.0000000 0.4596977
## [2,] 1  0.0000000 1.0000000
## [3,] 2 -0.6850734 0.4528502
## [4,] 3 -1.2520765 1.6495236
```

Question 2.3, #6(a)

```r
f6 <- function(x){
  exp(x) + 2^(-x) + 2*cos(x) - 6
}

f6prime <- function(x){
  exp(x) - log(2) / 2^x - 2*sin(x)
}
```

Here are the results when applying Newton's Method on $f(x) = e^x + 2^{-x} - 2cos(x) - 6$, with $p_0 = 1.5$:

```r
newtonMethod(f = f6, fprime = f6prime, initial = 1.5, iter = Inf, tol = 10^-5)
```

```
##      n      p_n     |f(p_n)|
## [1,] 0 1.500000 1.023283e+00
## [2,] 1 1.956490 5.797014e-01
## [3,] 2 1.841533 5.034095e-02
## [4,] 3 1.829506 5.021213e-04
## [5,] 4 1.829384 5.151614e-08
```

Here are the results when applying Secant Method on $f(x) = e^x + 2^{-x} - 2cos(x) - 6$, with $p_0 = 1, p_1 = 2$:

```r
secantMethod(f = f6, initials = c(1, 2), iter = Inf, tol = 10^-5)
```

```
##      n      p_n     |f(p_n)|
## [1,] 0 1.000000 1.701114e+00
## [2,] 1 2.000000 8.067624e-01
## [3,] 2 1.678308 5.456738e-01
## [4,] 3 1.808103 8.573869e-02
## [5,] 4 1.832298 1.198455e-02
## [6,] 5 1.829331 2.150281e-04
## [7,] 6 1.829383 5.247854e-07
```

```
fixedPoint
```

```
## function(f, initial, iter, tol){
##   results <- c(initial)
##
##   count <- 1
##   while(count <= iter & abs(f(results[count]) - results[count]) > tol){
##     results <- c(results, f(results[count]))
##     count <- count + 1
##   }
##   cbind("n" = 0:(length(results)-1),
##         "p_n" = results,
##         "|f(p_n) - p_n|" = abs(f(results) - results))
## }
## <bytecode: 0x7fe9b2340950>
```

```
newtonMethod
```

```
## function(f, fprime, initial, iter, tol){
##   results <- c(initial)
##
##   count <- 1
##   while(count <= iter & abs(f(results[count])) > tol){
##     results <- c(results, results[count] - f(results[count])/fprime(results[count]))
##     count <- count + 1
##   }
##
##   cbind("n" = 0:(length(results) - 1),
##         "p_n" = results,
##         "|f(p_n)|" = abs(f(results)))
## }
## <bytecode: 0x7fe9b2346c28>
```

```
secantMethod
```

```
## function(f, initials, iter, tol){
##   results <- c(initials)
##
##   count <- 2
##   while(count <= iter & abs(f(results[count])) > tol){
##     old <- results[count]
##     older <- results[count - 1]
##
##     fold <- f(old)
##     folder <- f(older)
##
##     newValue <- old - fold * (old - older)/(fold - folder)
##     results <- c(results, newValue)
##
##     count <- count + 1
##   }
##
```

```
##   cbind("n" = 0:(length(results) - 1),
##         "p_n" = results,
##         "|f(p_n)|" = abs(f(results)))
## }
## <bytecode: 0x7fe9b1a0b7f8>
```