Math 4610 – HW 3

Jake Daniels A02307117

October 4, 2022

Task 1:

The 6 codes written up in C can be found on my github. From the link below click on Root Finding Code (C) to find the 6 codes I wrote up for this first task. In github you will also see 'fval.c' and 'fpval.c' which are my two function files which will be used below in task 5. They are all able to be compiled which can be seen in the screenshots below.

https://jake-daniels16.github.io/math4610/

Task 2:

The following code is my 'root finding test.c' file I created to test Newton's Method (note root finding test has underscores between the words, just freaks out latex so I'm excluding them):

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double newtonsMethod();
double fval(double);
double fpval(double);

int main()
{
    double a = -0.5;
    double b = 0.8;
    double tol = 0.00000001;
    double maxIter = 25.0;
    double r3 = newtonsMethod(fval, fpval, a, tol, maxIter);
    printf("\n newtons method root = %f\n", r3);
}
```

After building and running the file on my IDE, the output is as follows:

```
 newtons method root = -0.000000
```

Which is the exact same as what we got on previous homework, and is where the root should be, namely at $x = 0$.

Task 3:

The adjusted 'root finding test.c' file can be found in the same place as the original six codes found through the link. After building and running it on my IDE the output for this using the same values as above yields:

```
 bisect root value = -0.000000

 functional iteration root value = 0.000000

 newton hybrid root value = -0.000000

 newtons method root value = -0.000000

 secant method root value = 0.000000

 secant hybrid root value = 0.000000
```

Note that all the following codes approximate a number near $x = 0$ meaning they all are able to approximate the root of our function well because we know that there exists a root at $x = 0$.

Task 4:

The following are screenshots showing me creating the shared library in the temp folder.

```
jdsoc@Jakes-Laptop ~
$ cd /cygdrive/c

jdsoc@Jakes-Laptop /cygdrive/c
$ cd Users/jdsoc/OneDrive\ -\ USU/Fundamentals\ of\ Computational\ Math/RF\ Shared\ Library/

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ ls
bisect.c  fpval.c  functional_iteration.c  fval.c  main.c  newton_hybrid.c  newtons_method.c  root_finding_test.c  secant.c  secant_hybrid.c

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ gcc -c bisect.c

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ gcc -c functional_iteration.c

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ gcc -c newtons_method.c

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ gcc -c newton_hybrid.c

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ gcc -c secant.c

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ gcc -c secant_hybrid.c

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ ar rcv root_finding.a *.o
a - bisect.o
a - functional_iteration.o
a - newton_hybrid.o
a - newtons_method.o
a - secant.o
a - secant_hybrid.o

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ ranlib root_finding.a

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ ar tv root_finding.a
rw-r--r-- 2883003748/40605028   1382 Oct  5 11:47 2022 bisect.o
rw-r--r-- 2883003748/40605028   1065 Oct  5 11:48 2022 functional_iteration.o
rw-r--r-- 2883003748/40605028   1443 Oct  5 11:48 2022 newton_hybrid.o
rw-r--r-- 2883003748/40605028   1084 Oct  5 11:48 2022 newtons_method.o
rw-r--r-- 2883003748/40605028   1147 Oct  5 11:48 2022 secant.o
rw-r--r-- 2883003748/40605028   1603 Oct  5 11:48 2022 secant_hybrid.o
```

Task 5:

The following screenshot is me creating the test root finding.exe file and running it.

```
jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ gcc -o root_finding_test root_finding_test.c fval.c fpval.c root_finding.a

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ ls
bisect.c  fpval.c               functional_iteration.o  main.c          newton_hybrid.o   newtons_method.o  root_finding_test.c    secant.c  secant_hybrid.c
bisect.o  functional_iteration.c  fval.c                  newton_hybrid.c  newtons_method.c  root_finding.a    root_finding_test.exe  secant.o  secant_hybrid.o

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$ ./root_finding_test.exe

 bisect root = -0.000000

 functional iteration root = 0.000000

 newtons method root = -0.000000

 secant method root = 0.000000

 newton hybrid root = -0.000000

 secant hybrid root = 0.000000

jdsoc@Jakes-Laptop /cygdrive/c/Users/jdsoc/OneDrive - USU/Fundamentals of Computational Math/RF Shared Library
$
```

Notice that the output is exactly the same as when we ran it before.