chapter 1: 1.8.2 Programming assignments

Nan Ziqian 3210104676

September 25, 2023

1 B

after running “test_bisection.cpp”

```
f1:
root: 0.860334
width of interval: 7.13406e-13
number of iteration: 40
f(root): 3.84937e-12

f2:
root: 0.641186
width of interval: 1.81717e-12
number of iteration: 38
f(root): -1.47882e-13

f3:
root: 1.82938
width of interval: 3.63798e-12
number of iteration: 38
f(root): 2.41585e-13

f4:
root: 4
width of interval: 8.8221e-13
number of iteration: 41
```

It can be seen that the 4th function does not have a root on [0,4].

2 C

after running “test_Newton.cpp”

```
root near 4.5:
root: 4.49341
f(root): 8.88178e-16
number of iteration: 4

root near 7.7:
root: 4.49341
f(root): 8.88178e-16
number of iteration: 4
```

3 D

```
f1:
root_n: 3.14159
root_n_1: 3.14159
f(root): -5.25802e-13
number of iteration: 29

f2:
root_n: 1.30633
root_n_1: 1.30633
f(root): -6.52367e-12
number of iteration: 11

f3:
root_n: -0.188685
root_n_1: -0.188685
f(root): 2.08722e-14
number of iteration: 8
```

You can find new root if you change f1's initial points, this is because $\sin(x/2) - 1$ is periodic by 4π.

If you set the initial points bigger, Secant Method won't find any root. But if you set them smaller, you can find a root every π. This is because e^x is sufficiently small to affect root and $\tan(x)$ is periodic by π.

If you set the second initial point x_1 to 0.5, you can find the other root around 0.455.

4 E

```
bisection:
root: 0.166166
width of interval: 1.81899e-12
```

```
number of iteration: 38
f(root): -3.64153e-13

newton method:
root: 0.166166
f(root): 2.34479e-13
number of iteration: 8

secant method:
root_n: 0.166166
root_n_1: 0.166166
f(root): 0
number of iteration: 5

f(c-0.005) = 0.0986519
f(c+0.005) = -0.0985668
```

All three method find the root 0.166166, and you can find that $f(c-0.005)*f(c+0.005) < 0$, therefore the root must be in $[c-0.005, c+0.005]$ and $|c-root| < 0.01$.

5 F

5.1 (a)

```
root: 0.575473
f(root): 9.59233e-13
number of iteration: 2
```

5.2 (b)

```
root: 0.578907
f(root): 0
number of iteration: 3
```

5.3 (c)

```
root_n: 0.578907
root_n_1: 0.578907
f(root): -6.98464e-12
number of iteration: 6

root_n: 2.56269
```

```
root_n_1: 2.56269
f(root): 1.42109e-14
```

As you can see, if you set both the initials around 2.5, the secant method will find the root 2.56269. But if you keep the 33° as one initial, you can only find 0.578907, this is because 33° is already very close to the root 0.578907, and you can't change the primary variable too much.