Homework#4

2015004302 곽상원

목차

---

1. Purpose of program

▶기존의 root finding 방법을 이용해
$f(R) = e^{-0.005R}\cos(\sqrt{2000 - 0.01R^2}\,0.05) - 0.01$에 대해, f(R)=0인 R을 구한다.
이 때, 각 방법 **iteration횟수**를 얻어내는 것이 목표이다.

2 .Experimental process

▶***Relative approximate error(r.e)가 10^-4(%), 10^-6(%)****일 때의 iteration 횟수를 구한다.*
▶Method는 Bisection, False-position, Newton-Raphson, Fail-safe, Secant, Muller method를 이용한다.

<Code>

//routine을 가리키기위한 pointer to function

```
typedef float(*rtMethod_1)(float(*)(float), float, float, float, int*);
typedef float(*rtMethod_2)(void(*)(float, float*, float*), float, float, float, int*);
```

//root finding routine을 담을 pointer to function. argument의 구성에 따라 두 종류로 나뉨

```
rtMethod_1 proper_routine[4];
rtMethod_2 another[2];
float root[6];// root는 해를 담을 변수
int a[6] = { 0, }, i;//a는 routine의 iteration횟수를 저장할 변수
float xacc;//Relative approximate Error(%) 크기(10^-4 or 10^-6)
```

//함수포인터에 함수 담기

```
proper_routine[0] = rtbis; proper_routine[1] = rtflsp; proper_routine[2] = rtsec;
proper_routine[3] = rtmull;
another[0] = rtnewt; another[1] = rtsafe;
```

//root와 iteration_num 구하기

```
for (i = 0; i < 4; i++) root[i] = proper_routine[i](E_C, 0, 400.0, xacc, &a[i]);
for (; i < 6; i++) root[i] = another[i-4](problem_EC, 0.0, 400.0, xacc, &a[i]);
```

3.Result

❶R.e = 10^-4(%)

```
Relative approximate error : 0.000100
<Bracketing Method>
[Bisection]   root:3.281513e+02  iteration:21
[False-position]   root:3.281517e+02  iteration:17



<Open method>
[Secant]   root:3.281514e+02  iteration:6
[Muller]   root:3.281514e+02  iteration:5
[Newton-Raphson]   root:3.281514e+02  iteration:6
[Fail-safe]   root:3.281514e+02  iteration:6
```

❷R.e = 10^-6(%)

```
Relative approximate error : 0.000001
<Bracketing Method>
[Bisection]   root:3.281514e+02  iteration:27
[False-position]   root:3.281514e+02  iteration:22



<Open method>
[Secant]   root:3.281514e+02  iteration:6
[Muller]   root:3.281514e+02  iteration:5
[Newton-Raphson]   root:3.281514e+02  iteration:7
[Fail-safe]   root:3.281514e+02  iteration:7
```

4.Discussion

❶R.e = 10^-4(%)

method	iterations
Bisection	21
False-position	17
Newton-Raphson	6
Fail-safe	6
Secant	6
Muller	5

❷R.e = 10^-6(%)

method	iterations
Bisection	27
False-position	22
Newton-Raphson	7
Fail-safe	7
Secant	6
Muller	5

⇒Open method의 converge 속도가 확연히 빠름을 확인할 수 있다.