NA-HW #04

2015004693_양상헌

What to do :

1.

[f(R) = e^(-0.005R)cos(0.05sqrt(2000-0.01R^2))-0.01]

- **Bisection Method**
- **Linear Interpolation Method(False Position)**
- **Secant Method**
- **Newton Raphson Method**
- **Newton Raphson with Bracketing.**
- **Muller Methods**

위 6가지 Root finding Methods를 이용하여 f(R)함수의 해를 구하고, 해를 구하는 과정에서 r.e.를 각각 0.0001%, 0.000001% 로 구분하여 Iteration이 각각 얼마나 일어나는지(Convergence Speed를) 비교.

2.

Solve the Exercise 8.32, 8.36

3.

Write the concept of "Pointer to function" and describe how it is used in homework #3.

1. Finding root of f(R) = e^(-0.005R)cos(0.05sqrt(2000-0.01R^2))-0.01

Process:

위 f(R) 함수와 f'(R)함수를 다음과 같이 선언해준다. f'(R)함수는 NewtonRaphson Method를 사용할 때 필요하므로 직접 구해서 선언해 준다. Hw03때 bessel function을 사용한 방법으로 선언한다.

```
float func01(float x){
    float ans = (expf(-0.005*x)*cosf(0.05*sqrtf(2000-0.01*x*x))-0.01);
    return ans;
}
float dfunc01(float x){
    float ans = (expf(-0.005*x)*((0.0005*x*sinf(0.05*sqrtf(2000-0.01*x*x))/sqrtf(2000-
0.01*x*x) )-0.005*cosf(0.05*sqrtf(2000-0.01*x*x))));
    return ans;
}
static float fx01(float x){
    return func01(x);
}
static void funcSet01(float x, float* fn, float* df){
    *fn = func01(x);
    *df = dfunc01(x);
}
```

이후로는 hw03에서와 같은 방법으로 실행하였는데 차이점은 xacc를 0.0001% 일때와 0.000001% 일때를 구분해준 점과 zbrak으로 구간을 나눌 때 1개의 큰 구간으로 나눠준 점이다. 밑에 코드는 Bisection Method부분의 일부이다.

```
//0.0001%
for (i=1;i<=nb;i++) {
       xacc=(1.0e-6)*(xb1[i]+xb2[i])/2.0;
       root=rtbis(fx01,xb1[i],xb2[i],xacc);
       printf("root %3d %14.9f %14.9f\n\n",i,root,fx01(root));
   }
. . .

//0.000001%
for (i=1;i<=nb;i++) {
       xacc=(1.0e-8)*(xb1[i]+xb2[i])/2.0;
       root=rtbis(fx01,xb1[i],xb2[i],xacc);
       printf("root %3d %14.9f %14.9f\n\n",i,root,fx01(root));
   }
```

나머지 Method들도 이와 같이 xacc의 경우를 두가지로 나누어서 실행시켜주었다.

Results:

Bisection Method

```
--------------------------------------------------------------------------------

--- < by Bisection Method > ---

Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.0001%
                    x            f(x)
Total iteration : 21
root   1  328.151336670   -0.000000090


Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.000001%
                    x            f(x)
Total iteration : 28
root   1  328.151428223   -0.000000001

--------------------------------------------------------------------------------
```

Linear Interpolation Method

```
--------------------------------------------------------------------------------

--- < by Linear Interpolation Method(False Position) > ---

Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.0001%
                    x            f(x)
Total Iteration : 18
root   1  328.151550293    0.000000129


Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.000001%
                    x            f(x)
Total Iteration : 22
root   1  328.151428223   -0.000000001

--------------------------------------------------------------------------------
```

Secant Method

```
--------------------------------------------------------------------------------

--- < by Secant Method > ---

Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.0001%
                    x            f(x)
Total Interpolation : 6
root   1  328.151428223   -0.000000001


Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.000001%
                    x            f(x)
Total Interpolation : 6
root   1  328.151428223   -0.000000001

--------------------------------------------------------------------------------
```

Newton Raphson Method

```
--------------------------------------------------------------------------------

--- < by Newton-Raphson Method > ---

Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.0001%
                    x            f(x)
Total Iteration : 5
root   1  328.151428223   -0.000000001


Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.000001%
                    x            f(x)
Total Iteration : 5
root   1  328.151428223   -0.000000001

--------------------------------------------------------------------------------
```

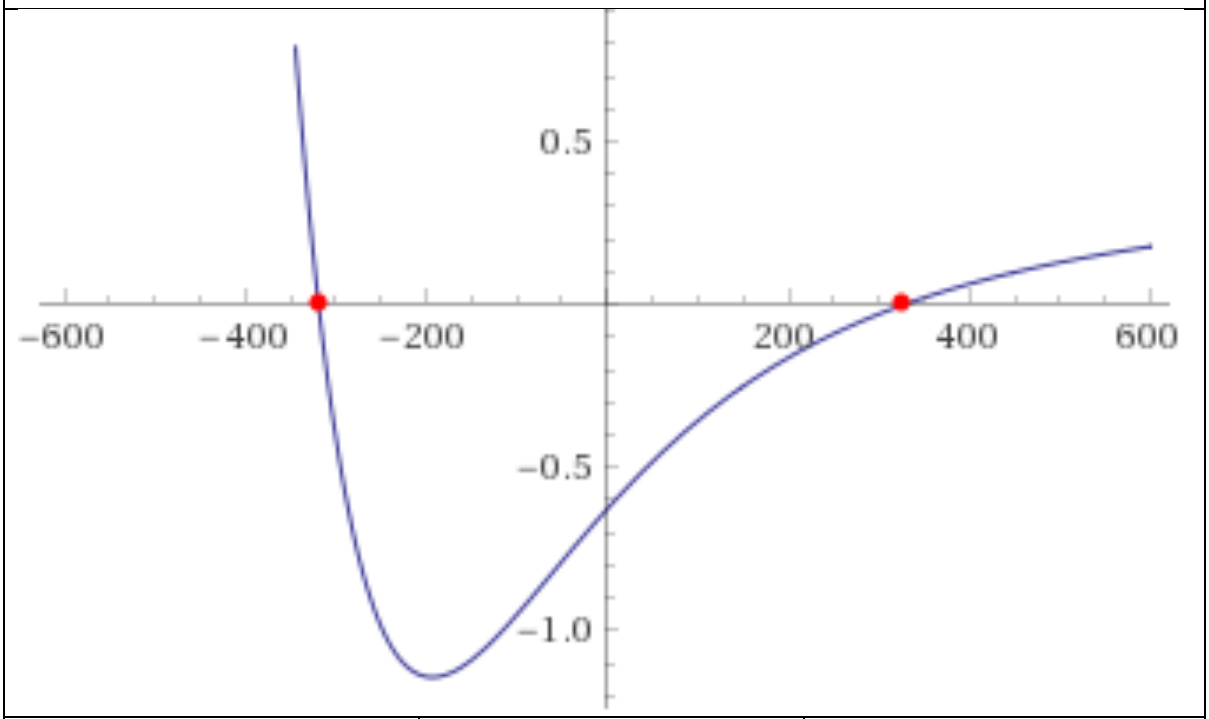
Newton Raphson with bracketing Method

```
--------------------------------------------------------------------------------

--- < by Newton w/ Bracketing Method > ---

Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.0001%
                    x            f(x)
Total Iterarion : 5
root   1  328.151428223   -0.000000001


Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.000001%
                    x            f(x)
Total Iterarion : 5
root   1  328.151428223   -0.000000001

--------------------------------------------------------------------------------
```

Muller Method

```
--------------------------------------------------------------------------------

--- < by Muller Method > ---

Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.0001%
                    x            f(x)
Total Iteration : 4
root   1  328.151428223   -0.000000001


Roots of e^(-0.005*x)*cos(0.05*sqrt(2000-0.01*x^2))-0.01 on [0, 400] with r.e. 0.000001%
                    x            f(x)
Total Iteration : 4
root   1  328.151428223   -0.000000001

--------------------------------------------------------------------------------
```

Overall Results



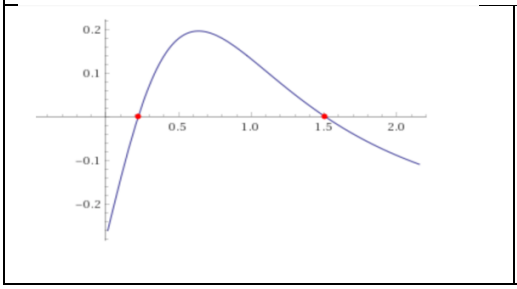
Root: x = 328.151428	Total Iteration for 0.0001%	Total Iteration for 0.000001%
>Bisection Method:	21	28
>Linear Interpolation Method:	18	22
>Secant Method:	6	6
>Newton-Raphson Method:	5	5
>Newton with Bracketing:	5	5
>Muller Method:	4	4

2. Solving Exercise 8.32 & 8.36

문제에 나와 있는 함수를 Bisection Method를 이용하여 풀었다.

Exercise 8.32:

$$f(x) = \frac{x}{(x^2 + 0.81)^{3/2}} - 0.0885\,\pi$$



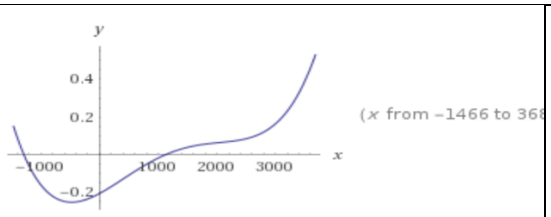

```
---------------------------------------------------
--- < by Bisection Method > ---

Roots of Ex8.32 on [0, 100] with r.e. 0.0001%
                    x            f(x)
Total iteration : 21
root   1    0.221349716   -0.000000405

Total iteration : 20
root   2    1.509785652   -0.000000107

---------------------------------------------------
```

Solution:

x = 0.2213, x = 1.5099.

Exercise 8.36:

$$f(x) = 1.952\times10^{-14}\,x^4 - 9.5838\times10^{-11}\,x^3 + 9.7215\times10^{-8}\,x^2 + 0.0001671\,x - 0.20597$$




```
---------------------------------------------
--- < by Bisection Method > ---

Roots of Ex8.36 on [0,2000] with r.e. 0.0001%
                    x            f(x)
Total iteration : 15
root   1 1126.009521484   -0.000000030

---------------------------------------------
```

Solution

x = 1126.0095.

3. “Pointer to Function”

함수를 포인터를 이용하여 선언하고 사용하는 이유는 C의 경우, 함수를 인자로 받아 다른 일을 하는 함수를 사용하기 위함이다. C에서는 포인터를 사용하지 않고 어떤 함수 내에서 다른 함수를 사용하게 되면, 만약 함수 내에서 사용되는 함수가 다른 함수로 변경이 되어야하는 경우 이를 쉽게 변경하기 힘들고 전체 함수를 고쳐야 되는 일이 발생한다. 이러한 불편함을 해결하기 위해 함수를 포인터로 선언하여 이를 다른 함수의 인자로 넘겨 사용한다.

Ex) in hw#03

```
. . .

float func01(float x){
    float ans = (10.0*expf(-1*x)*sin(2.0*M_PI*x) - 2.0);
    return ans;
}

float dfunc01(float x){
    float ans = (10*-expf(-1*x)*(sin(2*M_PI*x)-2*M_PI*cos(2*M_PI*x)));
    return ans;
}

static float fx01(float x){
    return func01(x);
}

static void function01(float x, float* fn, float* df){
    *fn = func01(x);
    *df = dfunc01(x);
}

. . .

static float fx(float x)
{
       return bessj0(x);
}

static void funcd(float x,float *fn, float *df)
{
       *fn=bessj0(x);
       *df = -bessj1(x);
}

float rtsafe(void (*funcd)(float, float *, float *), float x1, float x2, float xacc) {

. . .

}

int main (){

. . .

       root=rtsafe(function01,xb1[i],xb2[i],xacc);

. . .

}
```

위 방식대로 Pointer to Function을 이용하여 함수를 인자로 받아 계산을 수행해는 함수를 실행하였다. 위 코드에서 **function01()** 함수 내에서 **func01()** 함수와 **dfunc01()** 함수를 포인터로 선언해주고, 이를 **rtsafe()** 의 인자로 사용하는 방식이다.