수치해석 과제1

Fixed - point implementation

과목명	수치해석
담당교수	심동규 교수님
학과	컴퓨터정보공학부
학년	3학년
학번	2019202009
이름	서여지
제출일	21.10.04

KWANGWOON UNIVERSITY

1. 과제 개요

bisection method를 이용하여 주어진 방정식의 해를 구하는 프로그램을 작성한다. 이때 프로그램은 16 bits fixed-point를 사용하고, 구한 해는 'N >> n'의 형태로 출력한다.

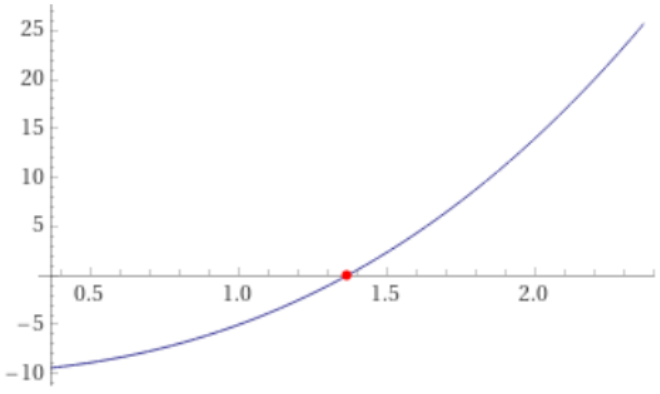
bisection method (이분법)은 초기 가정된 구간을 절반 크기의 구간으로 나누어서 해를 포함하는 구간을 찾는 방법으로 방정식의 해를 구한다. bracketing method(구간법)에 해당하며 수렴성이 보장되지만 수렴 속도가 느린 편이다.

fixed point(고정소수점)은 exponent의 소수점의 위치가 변화하는 floating point(부동소수점)과 달리 소수점의 위치를 변화시키지 않는 방법이다. 이번 과제에서는 32bit를 이용하는 int형 변수를 사용하여 하위 16bit을 소수부분으로, 상위 16bit을 정수부분을 표시하도록 사용하였다.

2. 과제 수행 방법

주어진 방정식의 해, 혹은 해가 포함된 구간을 구하는 프로그램을 작성한다. 주어진 문제와 해의 근사값은 다음과 같다.

$x^3 + 4x^2 - 10 = 0$ $\quad x \approx 1.36523001341410$



또한 fixed point를 이용한 방법과 floating point를 이용한 방법을 모두 구현하여 비교하도록 한다. 각각 고정소수점과 부동소수점을 이용하는 fixedPoint( )와 floatingPoint( )함수의 구조는 동일하다. 먼저 초기 구간의 중간 값인 c를 찾은 뒤, y(a)와 y(c) 값의 부호를 곱하여 다음에 사용할 구간을 찾는다. y(a)*y(c)가 음수인 경우 [a,c], 양수인 경우 [c,b]를 다음 계산에 사용한다. 구간의 크기가 tolerance 크기가 될 때까지 계산을 반복한다. tolerance의 크기는 1/2^16 값을 이용하였다. 또한 y(x)의 값이 [-tolerance, +tolerance] 사이에 위치한다면 0으로 판단하여 계산을 종료한다.

$\frac{1}{2^{16}} = 0.0000152587890625$ (fixed point에서 0x01, floating point에서 1/pow(2,16))

3. 결과

```
fixedPoint
------------------iter: 16
[a,b] = [15D7F >> 16, 15D80 >> 16]


floatingPoint
------------------iter: 16
[a,b] = [1.365219, 1.365234]
```

실험 결과는 위와 같이 나타났다.

15D7F >> 16 = 0001. 0101 1101 0111 1111, 15D80 >> 16 = 0001. 0101 1101 1000 0000

$$1+\frac{1}{2^2}+\frac{1}{2^4}+\frac{1}{2^5}+\frac{1}{2^6}+\frac{1}{2^8}+\frac{1}{2^9}-\frac{1}{2^{16}}=\frac{89471}{65536}=1.3652191162109375$$

$$1+\frac{1}{2^2}+\frac{1}{2^4}+\frac{1}{2^5}+\frac{1}{2^6}+\frac{1}{2^8}+\frac{1}{2^9}=\frac{699}{512}=1.365234375$$

fixed point와 floating point 모두 계산을 반복한 수는 16번으로 동일하게 나타났다. 두 함수 모두 방정식의 해 $x \approx 1.36523001341410$ 를 포함하는 구간을 출력하였다. 구간을 계속해서 반으로 나누는 이번 과제에서 fixed point의 경우 정확한 값을 표현할 수 있지만 floating point의 경우 근삿값을 이용하는 것을 확인할 수 있다.

a	1.36521912
b	1.36523438

4. 고찰

fixed point 방식의 경우 floating point보다 좁은 범위의 값을 표현하지만, 표현가능한 값 사이의 간격이 일정하다는 특징이 있다. 이 프로그램의 경우 1/2^16의 간격으로 수를 표현할 수 있었다. 따라서 정확한 계산이 필요한 경우에 floating point를 사용하는 것 보다 tolerance를 고려한 소수점 자릿수를 갖는 fixed point를 사용하는 것이 효과적이라는 것을 확인할 수 있다.

프로그램을 작성하는 과정에서 tolerance를 잘못 이용하여 결과가 정상적으로 출력되지 않는 오류가 발생했다. fixed point의 경우 x값의 구간에 대해 tolerance를 적용하고, floating point 함수에서는 f(x)의 값에 tolerance를 적용하여 나타난 오류였다. 두 함수의 조건을 동일하게 맞추어 문제를 해결하였다.