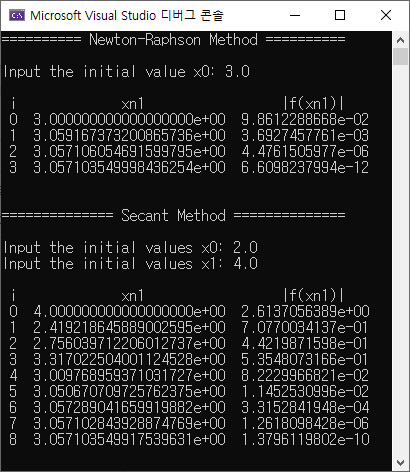
**고급 소프트웨어실험 4주차 과제**

**실습 문제 1-1**

1. 두 방법에 대한 초기값을 각각 x0 = 3.0과 x0 = 2.0, x1 = 4.0으로 설정하여 자신이 작성한 프로그램을 수행시켜 자신의 프로그램이 원하는 근을 정확히 찾고 있는지 분석하라. 과연 근이 맞는지 어떻게 확인할 수 있을까?

* 수행 결과



* 근이 맞는지 확인하기 위해서는 구해진 근을 식에 대입해보면 된다. 프로그램에서 f(xn1)의 값을 보면, Newton-Raphson Method 에서는 , Secant Method 에서는 로 0과 매우 가까우므로, 근을 잘 찾고 있다고 볼 수 있다.

1. 위에서 산출한 결과를 볼 때, 각 방법의 근의 수렴 속도가 과연 앞에서 설명한 속도로 수렴하는지 비교 분석한 후 그 결과를 보고서에 기술하라.

* Newton-Raphson Method에서 근의 수렴 속도를 확인하기 위해 를 계산해보면, n = 0, … ,2 일 때 각각 대략적으로 0.63, 0.59, 15.69으로 근이 εn+1 ≈ cεn2 (c≈0.6)의 속도로 수렴하는 것을 확인할 수 있다.

마찬가지로, Secant Method에서 근의 수렴 속도를 확인하기 위해 를 계산해보면, n = 0, …, 7 일 때 각각 대략적으로 0.70, 0.62, 1.81, 0.42, 0.90, 0.66, 0.78, 0.16으로 근이 εn+1 ≈ cεn2 (c≈0.7)의 속도로 수렴하는 것을 확인할 수 있다.

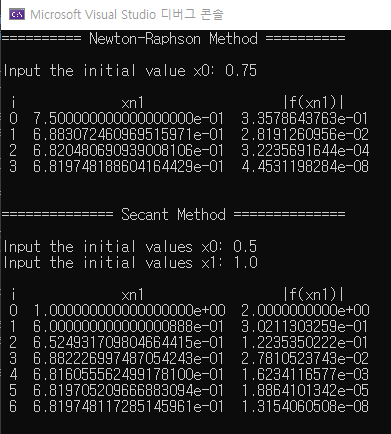
1. 위에서 제시한 초기값 외에 임의의 초기값들을 사용하여 자신이 작성한 프로그램을 수행한 후, 이 두 방법이 항상 임의의 초기값에 대해 빠르게 수렴하는지 보고서에 기술하라.

* Newton-Raphson Method에서 초기값을 3, 30, 300으로 주었을 때 근의 수렴 속도는 각 3, 8, 12 로 근과 초기값이 가까울수록 근이 빠르게 수렴하는 것을 확인할 수 있었다.

Secant Method의 경우 초기값이 구하고자 하는 근과 멀어질 경우 근을 산출해내지 못했다.

1. 위에서 작성한 프로그램을 사용하여 함수 f2(x) = x+1−2sinπx = 0에 대해 초기값을 각각 x0 = 0.75과 x0 = 0.5, x1 = 1.0으로 설정하여 위 문제를 반복하라.

* 수행 결과



* 근이 맞는지 확인하기 위해서는 구해진 근을 식에 대입해보면, Newton-Raphson Method에서는 , Secant Method 에서는 로 그 값이 0과 매우 가까우므로, 근을 잘 찾고 있다고 볼 수 있다.
* Newton-Raphson Method에서 근의 수렴 속도를 확인하기 위해 를 계산해보면, n = 0, … ,2 일 때 각각 대략적으로 0.45, 0.42, 0.42으로 근이 εn+1 ≈ cεn2 (c≈0.42)의 속도로 수렴하는 것을 확인할 수 있다.

마찬가지로, Secant Method에서 근의 수렴 속도를 확인하기 위해 를 계산해보면, n = 0, …, 5 일 때 각각 대략적으로 0.76, 0.56, 0.57, 0.59, 0.58, 0.58으로 근이 εn+1 ≈ cεn2 (c≈0.58)의 속도로 수렴하는 것을 확인할 수 있다.

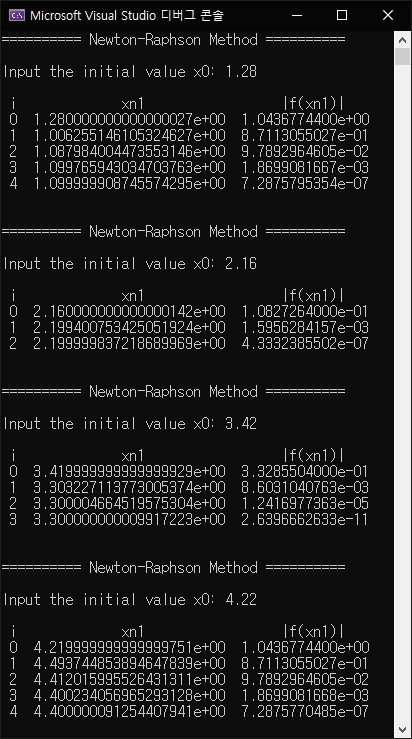
**실습 문제 1-2**

다음과 같은 다항식에 대한 방정식을 고려하자.

f3(x) = x 4 −11.0x 3 +42.35x 2 −66.55x+35.1384 = 0

지금 어떤 근분리(root separation) 방법을 사용하여, 이 방정식은 네 개의 서로 다른 실근을 가지며, 각 실근은 [1.02,1.48], [1.95,2.37], [3.11,3.73], [3.83,4.61] 구간에 존재한다는 사실을 밝혀냈다. 이 사실을 근거로 Newton-Raphson 방법을 사용하여 모든 실근을 구하라.

* 수행 결과



* 각 범위의 중간값을 초기값으로 주어 네 개의 근을 구하였다. 네 근을 주어진 식에 대입해보면 그 결과가 각 , , , 으로 그 값이 0과 매우 가까우므로, 근을 잘 찾았다고 볼 수 있다.

**실습 문제 1-4**

다음 f1(x) = lnx − 1 = 0과 같이 근을 알고 있는 비선형 방정식을 고려하 자 (잘 알다시피 근은 e = 2.718281828459045235360287471352···임). 이 방 정식의 근을 Newton-Raphson 방법을 사용하여 구하려 하는데, 적절한 초기 값 x0에 대해 자신이 작성한 double-precision 버전과 single-precision 버전 각각을 사용하여 근을 구하여 보자. 이때 부동 소수점 연산의 정밀도가 다 른 두 방법이 구한 근의 값이 정확한 근 e와 비교하여 어떤 차이가 있는지, 자신이 알아낸 사실을 보고서에 상세히 기술하라

* 수행결과

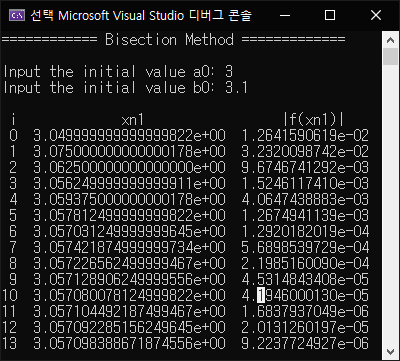
|  |  |
| --- | --- |
| Double-precision 버전 | Single-precision 버전 |
|  |  |

두 버전을 사용하여 구한 근은 각각 2.718281827760716141, 2.718281745910644531 으로, e = 2.718281828459045235… 와의 오차는 각각 6.9832939, 8.25484006이다. 두 값 모두 e값과의 오차가 10-7이하로 매우 작으나, double-precision 버전에서의 오차가 더욱 작은 것을 확인할 수 있다.

**과제 문제 1**

1. *f*1 (x)

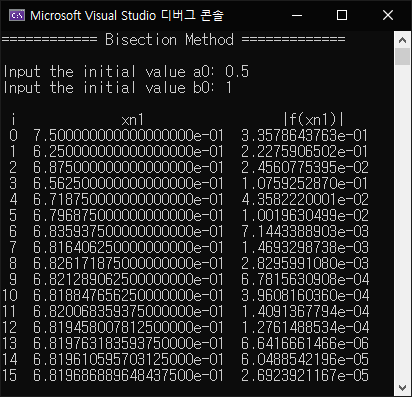
* 수행결과



* 구한 근을 주어진 식에 대입해보면 그 결과가 으로 그 값이 0과 매우 가까우므로, 근을 잘 찾았다고 볼 수 있다.
* 근의 수렴 속도를 확인하기 위해 를 계산해보면, 이론적인 수렴속도인 0.5로 나타나지는 않았다. 그러나 대부분의 경우에서 0과 1 사이의 값이 나타나는 것을 확인할 수 있었다.

1. *f*2 (x)

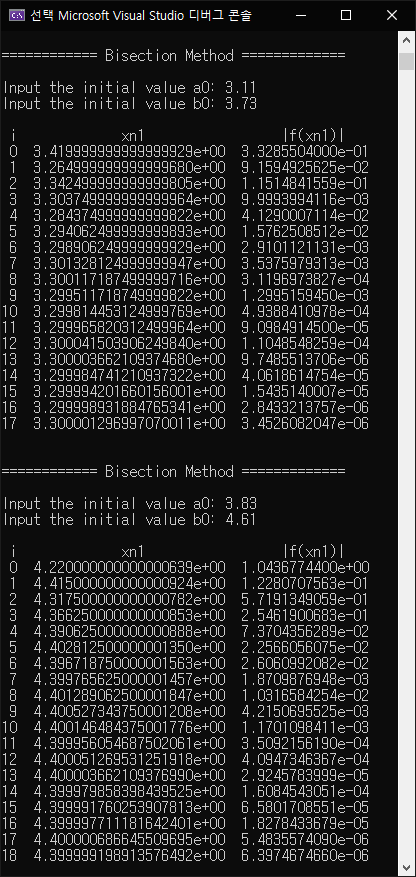
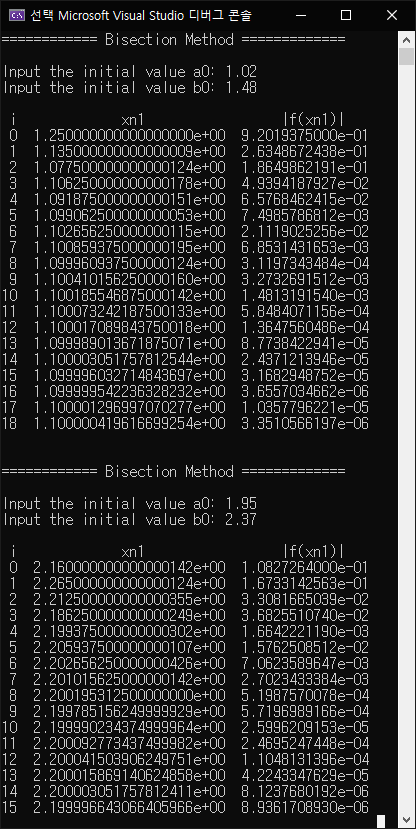
* 수행결과



* 구한 근을 주어진 식에 대입해보면 그 결과가 으로 그 값이 0과 매우 가까우므로, 근을 잘 찾았다고 볼 수 있다.
* 근의 수렴 속도를 확인하기 위해 를 계산해보면, 이론적인 수렴속도인 0.5로 나타나지는 않았다. 그러나 대부분의 경우에서 0과 1 사이의 값이 나타나는 것을 확인할 수 있었다.

1. *f*3 (x)

* 수행 결과

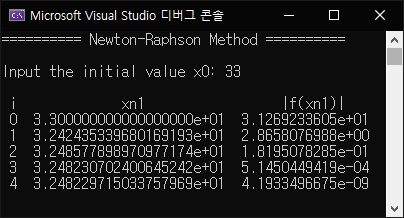


* 구한 근을 주어진 식에 대입해보면 그 결과가 각각 , , , 으로 그 값이 0과 매우 가까우므로, 근을 잘 찾았다고 볼 수 있다.
* 근의 수렴 속도를 확인하기 위해 를 계산해보면, 이론적인 수렴속도인 0.5로 나타나지는 않았다. 그러나 대부분의 경우에서 0과 1 사이의 값이 나타나는 것을 확인할 수 있었다.

**과제 문제 2**

1. 숙제 1에서 자신이 작성한 Newton-Rapson 방법을 사용하여 이 문제를 해결해라.

* 수행결과



1. 이를 위하여 어떠한 반복문을 사용하였는지 보고서에 관련 수식을 설명한 후, 자신이 구한 답에 수렴해가는 반복 결과와 함께 제출하라.



* 위와 같은 식을 사용하여 반복문을 통해 초기값 33에서부터 xn값을 구해나갔다. 현재 구한 xn+1 에 대해 함수 값이 충분히 작거나, 충분히 많은 회수만큼 반복문을 수행하였거나, 더 이상 의미있는 진전이 없는 경우 반복문을 탈출하도록 하였다.

반복결과는 위의 수행 결과에서 확인할 수 있다.