**고급소프트웨어실습I**

**4주차 보고서**

학번 : 20171663

이름 : 이도훈

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? 논리적으로 타당한 방법으로 분석한 내용을 보고서에 기술하라.

텍스트이(가) 표시된 사진

자동 생성된 설명

위에 있는 결과가 newton-raphson 방법으로 근을 구한 것이고 밑에 있는 결과가 secant 방법으로 근을 구한 것이다. 두 방법 모두 3.0571035에 근사한다는 것을 확인할 수 있다. 구한 값이 근이 맞는지 확인하는 방법은 도출해 낸 결과값이 특정 숫자 이하(예 : 0.000001)일 시 0에 수렴했다는 가정을 통해 확인할 수 있다.

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

Newton-raphson 방법을 통해 근을 구한 것은 반복문 3번 만에 근을 찾아낸 것을 확인할 수 있다. 이에 반해 secant 방법을 통해 근을 구한 것은 8번 만에 근을 구한 것을 확인할 수 있다. Newton-raphson 방법은 1차 도함수를 이용해서 해당 점에서의 접선의 방정식을 통해 다음 위치의 점을 구하고 이러한 방식을 반복하여 특정 조건에 도달하게 되면 그 위치의 점을 근으로 판단한다.

Secton은 도함수를 구할 필요는 없다. 그 대신 접선의 기울기를 해당 위치의 점을 포함하고 있는 두 점 사이의 기울기로 대체해서 사용하기 때문에 수렴하는 속도가 newton-raphson 방법보다 느리고 이를 반복문 도는 횟수를 통해 확인할 수 있다.

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

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

임의의 초기값을 사용하여 근을 구해보았는데 두 경우 모두 빠르게 수렴하는 것을 확인할 수 있었고 동시에 newton 방법이 secton 방법보다 빠른 것을 확인할 수 있었다.

4. 다음 f1(x) = lnx − 1 = 0과 같이 근을 알고 있는 비선형 방정식을 고려하자 (잘 알다시피 근은 e = 2.718281828459045235360287471352···임). 이 방정식의 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 다른 두 방법이 구한 근의 값이 정확한 근 e와 비교하여 어떤 차이가 있는지, 자신이 알아낸 사실을 보고서에 상세히 기술하라.

텍스트이(가) 표시된 사진

자동 생성된 설명

위에가 double type을 이용한 결과값이고 밑에가 float type 이용한 결과값이다. 두 경우 중에서 double type을 이용한 결과값이 e의 값(2.71828182845904…)에 더 근접한 것을 확인할 수 있다. C 언어에서 float type은 소수점 이하 6자리까지 표현이 가능하고 double type 같은 경우에는 소수점 이하 15자리까지 표현이 가능해서 정밀도의 차이가 있고 이러한 정밀도의 차이로 함수값을 구하는 반복문이 진행되면 진행될수록 비교적 더 정확한 값을 double-precision 버전이 구할 수 있다.

5. 프로그램이 완성되면, 실습 시간에 사용한 세 개의 함수 f1(x), f2(x), 그리 고 f3(x)에 대해 적절한 초기 구간을 사용하여 올바르게 근에 수렴하는지에 대한 분석 내용을 보고서에 기술하라.

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

왼쪽부터 차례대로 f1(x), f2(x), f3(x)에 bisection method를 이용하여 근을 구한 모습이다.

f1(x)의 경우 3.0571035라는 값에 수렴하는 모습을 보이고 있고, f2(x)의 경우에도 0.68197481이라는 값에 수렴하는 모습을 보이고 있고, f3(x)의 경우에도 해당 범위의 근인 1.1에 수렴하는 모습을 보이고 있다.

6. Bisection 방법의 수렴 속도는 선형적인, 즉 εn+1 ≈ 1/2 εn 형태를 보인다. 위의 세 함수에 대한 방정식을 대상으로 Newtown-Raphson 방법, Secant 방법, 그리고 Bisection 방법을 적용해보고, 과연 각 방법이 이론적인 수렴 속도를 보이는지를 분석하고 그 내용을 보고서에 기술하라.

Bisection 방법은 두 경계값의 중앙값을 이용하여 범위를 좁혀나가는 방식을 사용하기 때문에 문제에 적힌 것과 같이 선형적인 속도를 보인다. 따라서 같은 함수에서 같은 초기값을 이용하여 근을 구하더라도 f1(x)에서는 18번의 반복문을 돌고, f2(x)에서는 16번의 반복문을 돌고, f3(x)에서는 16번의 반복문을 도는 등 newton 방법과 secant 방법보다 오래 걸리는 모습을 보이고 있다.

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

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

초기값을 34로 잡았을 때 428번의 반복문을 돈 뒤 결과값인 33도에 수렴하는 값이 나오는 것을 확인할 수 있었다. Newton-raphson 방법을 사용하여 1차 도함수를 사용하여 해당 위치의 좌표에서의 접선을 통해 다음 위치의 좌표를 구하는 방법을 반복해서 결과값을 도출해냈다. 정지 조건으로는 현재 구한 xn+1에 대해 함수 값이 0.000001보다 작을 경우, 1000번 이상 반복문이 실행 됐을 경우, 현재 구한 xn+1이 xn에 비해 더 이상 의미 있는 전진을 하지 않을 경우 반복문을 중단했다.