**고급소프트웨어실습1 Week4 HW**

컴퓨터공학 20172141 김미소

**[프로그램 구동 방법]**

Ctrl+F5를 눌러 콘솔 창을 띄운 후 Newton-Raphson 방법의 경우 초기값을 1개, Secant 방법과 Bisection 방법의 경우 초기값을 2개 입력하여야 한다. 근을 찾는 과정은 프로젝트 폴더 내 result.txt에 출력이 되고 근을 찾은 결과는 콘솔 창에 출력 된다.

실습 및 과제에서 사용한 입력 값은

텍스트이(가) 표시된 사진

자동 생성된 설명다음과 같다.

**[실습 1-1]**

**(2) 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.**

근을 정확히 찾고 있는지 확인하기 위해서는 찾아낸 근을 다시 함수에 대입해보는 방법이 있다. 찾아낸 근을 대입해 본 결과,

텍스트, 실내이(가) 표시된 사진

자동 생성된 설명

Newton-Raphson 방법으로 찾아낸 근 = 3.057103549998436e+00에 대하여

= 6.609823799408332e-12로 0에 굉장히 근접하고 임의로 설정한 Delta 값 0.000001보다 작으므로 이 해에 근접하다고 할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Secant 방법으로 찾아낸 근 = 3.057103549917540e+00에 대하여

= 1.379611980212303e-10으로 0에 굉장히 근접하고 임의로 설정한 Delta 값 0.000001보다 작으므로 이 해에 근접하다고 할 수 있다.

**(3) 위에서 산출한 결과를 볼 때, 각 방법의 근의 수렴 속도가 과연 앞에서 설명한 속도로 수렴하는지 비교 분석한 후 결과를 보고서에 기술하라.**

두 방법에 대하여 실제 근을 찾는 것은 어려우므로 반복문이 종료된 시점에서 구한 근을 실제 근으로 가정하고 절대 오차를 계산해보았다.

f1에 대하여

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

위의 결과는 Newton-Raphson 방법으로 구한 결과이다. 차례대로 반복 횟수, 실제 근, xn, 절대 오차이다. 반복 횟수가 3이고 위의 절대 오차를 가지고 c를 구하면 0.58정도이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 결과는 Secant 방법으로 구한 결과이다. 차례대로 반복 횟수, 실제 근, xn, 절대 오차이다. 반복 횟수는 8이고 위의 절대 오차를 가지고 c를 구하면 0.78정도이다.

f2에 대하여

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

위의 결과는 위의 결과는 Newton-Raphson 방법으로 구한 결과이다. 반복 횟수는 4이다. 위의 결과를 가지고 c를 구하면 1.82정도이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 결과는 Secant 방법으로 구한 결과이다. 반복 횟수는 6이다. 위의 결과를 가지고 c를 구하면 1.56정도이다.

Newton-Raphson 방법이 Secant 방법보다 오차가 줄어드는 속도가 높아 근의 수렴 속도가 빠르다.

**(4) 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 초기값 | 반복 횟수 | Secant 초기값 | 반복 횟수 |
| 3 | 3 | 2, 4 | 8 |
| 4 | 4 | 4, 5 | 6 |
| 6 | 6 | 6, 7 | 8 |
| 10 | 7 | 10, 12 | 9 |
| 100 | 10 | 100, 150 | 15 |
| 1000 | 14 | 1000, 1500 | 19 |
| 100000 | 20 | 123456, 135790 | 29 |
| 1000000 | 24 | 1234567, 3456789 | 35 |

모든 숫자에 대하여 테스트해 볼 수는 없겠지만 초기값을 임의로 설정하여 프로그램을 실행했을 때 모두 Newton-Raphson 방법이 Secant 방법에 비하여 근의 수렴 속도가 빨라 반복 횟수가 적었다. 그리고 100만이 넘는 초기값에 대해서도 반복 횟수가 50이 넘지 않고 빠르게 근을 찾을 수 있다.

**[실습 1-2]**

**지금 어떤 근 분리(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.**

각 구간의 중점(1.25, 2.16, 3.42, 4.22)을 초기값으로 설정하여 Newton-Raphson 방법을 통해 근을 구한 결과,

텍스트이(가) 표시된 사진

자동 생성된 설명

각 초기값에 대하여 위와 같은 실근 4개를 찾을 수 있었다. 근을 찾는 과정은

텍스트, 신문이(가) 표시된 사진

자동 생성된 설명

위의 과정을 갖는다. 실근은 1.099999997327960e+00, 2.199999837218696e+00, 3.300000000009910e+00, 4.400000091254436e+00이다.

**[실습 1-4]**

**다음 f1(x) = ln x−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 초기값 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.**

Double-precision 버전과 single-precision 버전 모두 초기값을 2로 설정하고 실행한 결과,

텍스트이(가) 표시된 사진

자동 생성된 설명

다음과 같은 값을 찾을 수 있었다. 실제 e = 2.718281828459045235360287471352···에 대하여 절대 오차는

텍스트이(가) 표시된 사진

자동 생성된 설명

다음과 같고, single-precision에 대한 오차가 double-precision에 대한 오차보다 큰 것이 확인되었다. Double 타입이 float 타입보다 정밀도가 높기 때문에 오차가 적다.

**[숙제 1]**

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

텍스트, 신문이(가) 표시된 사진

자동 생성된 설명텍스트, 신문이(가) 표시된 사진

자동 생성된 설명텍스트, 신문이(가) 표시된 사진

자동 생성된 설명

차례대로 f1(x), f2(x), f3(x)에 대한 Bisection 결과이다. Bisection 방법의 종료 조건은 근 에 대하여 이 Delta보다 작을 경우 반복문을 종료하게 된다. f1(x)에 대하여 a0=2, b0=4를 초기값으로 설정하였고 f2(x)에 대하여 a0=0.5, b0=1을 초기값으로 설정하였고 f3(x)에 대하여 a0=1, b0=1.5를 초기값으로 설정하였다. 각 식에 대한 근 을 넣은 결과 이 모두 Delta = 0.000001보다 작고 0과 근접하기 때문에 해와 근접한다고 할 수 있다.

**(4) Bisection 방법의 수렴 속도는 선형적인, 즉 εn+1 ≈ 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.**

Newton-Raphson 방법과 Secant 방법에 대해서는 앞에서 설명했으므로 Bisection에 대한 설명만 하겠다.

|  |  |  |  |
| --- | --- | --- | --- |
| n | xn1 | 절대오차 | C = εn+1 / εn |
| 0 | 3.00000000000000E+00 | 5.71031570434499E-02 | 7.75608330410784E+00 |
| 1 | 3.50000000000000E+00 | 4.42896842956550E-01 | 4.35534472697684E-01 |
| 2 | 3.25000000000000E+00 | 1.92896842956550E-01 | 3.51985247222754E-01 |
| 3 | 3.12500000000000E+00 | 6.78968429565501E-02 | 7.94859189550791E-02 |
| 4 | 3.06250000000000E+00 | 5.39684295655007E-03 | 4.79042233609416E+00 |
| 5 | 3.03125000000000E+00 | 2.58531570434499E-02 | 3.95625069165056E-01 |
| 6 | 3.04687500000000E+00 | 1.02281570434499E-02 | 2.36177156176626E-01 |
| 7 | 3.05468750000000E+00 | 2.41565704344993E-03 | 6.17054875646288E-01 |
| 8 | 3.05859375000000E+00 | 1.49059295655007E-03 | 3.10300703768546E-01 |
| 9 | 3.05664062500000E+00 | 4.62532043449926E-04 | 1.11134020621800E+00 |
| 10 | 3.05761718750000E+00 | 5.14030456550074E-04 | 5.00927643916092E-02 |
| 11 | 3.05712890625000E+00 | 2.57492065500742E-05 | 8.48148147886509E+00 |
| 12 | 3.05688476562500E+00 | 2.18391418449926E-04 | 4.41048034916312E-01 |
| 13 | 3.05700683593750E+00 | 9.63211059499258E-05 | 3.66336633616622E-01 |
| 14 | 3.05706787109375E+00 | 3.52859496999258E-05 | 1.35135135099420E-01 |
| 15 | 3.05709838867187E+00 | 4.76837157981080E-06 | 2.20000000093132E+00 |
| 16 | 3.05711364746093E+00 | 1.04904174800247E-05 | 2.72727272823484E-01 |
| 17 | 3.05710601806640E+00 | 2.86102295010693E-06 | 3.33333330901547E-01 |
| 18 | 3.05710220336914E+00 | 9.53674309744912E-07 | 1.00000001071020E+00 |
| 19 | 3.05710411071777E+00 | 9.53674319958964E-07 | 0.00000000000000E+00 |
| 20 | 3.05710315704345E+00 | 0.00000000000000E+00 |  |

이론 상의 수렴 속도 1/2에 비해 불안정한 값들을 도출한다. 그리고 Bisection 방법이 Newton-Raphson과 Secant에 비해 속도가 많이 떨어지는 것을 확인할 수 있다.

**[숙제 2]**

**여러분이 작성한 Newton-Raphson 방법을 사용하여 정확한 각도 α 값을 구하라.**

우선 식에 대한 선언을 function.cpp에 해주었다.

이고

이를 미분하면

이다.

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

텍스트, 화면, 스크린샷이(가) 표시된 사진

자동 생성된 설명

\_fp\_vehicle은 \_f\_vehicle을 미분한 함수이다. 삼각함수 계산을 할 때 인자로 들어가는 값을 라디안으로 변경한 후에 계산하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

α값은 Newton-Raphson 방법을 이용하여 구하였다. Newton-Raphson 방법은

이 방법을 사용하므로 while 반복문 안에 구현하였다. 그리고 종료 조건은

텍스트이(가) 표시된 사진

자동 생성된 설명

위와 같다. 따라서 위의 조건을 만족할 경우 break를 통해 while 루프를 빠져나올 수 있다.

α값으로 33을 넣어보면

텍스트이(가) 표시된 사진

자동 생성된 설명

다음과 같이 33의 값과 근접한 값이 나오지만 Nmax 조건에 부딪혀 50에 다다르면 근 찾기를 종료하게 된다.

텍스트, 신문이(가) 표시된 사진

자동 생성된 설명텍스트, 신문이(가) 표시된 사진

자동 생성된 설명

result.txt에 출력한 결과이다.

그러나 근에 직접 대입한 값이 Delta보다 작아 Nmax 조건을 지워봤는데 그 결과로

텍스트, 시계이(가) 표시된 사진

자동 생성된 설명

적절한 근을 찾는데 222번의 반복이 필요했다.

result.txt의 일부만 첨부하면 다음과 같다.

텍스트, 신문이(가) 표시된 사진

자동 생성된 설명