**고급소프트웨어실습1 - 4주차 과제**

비선형 방정식의 풀이 기초

|  |  |  |
| --- | --- | --- |
| **분반** | **:** | 3 |
| **학번** | **:** | 20121635 |
| **이름** | **:** | 장종석 |

1. **실습 문제 1-1,** 
   1. *두 방법에 초기값을 각각 과 으로 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.*

다음은 실습 문제 1-1에 해당하는 식의 근을 Newton-Raphson 방법, Secant 방법으로 구한 결과이다. (는 주어진 식의 근 중 하나로 여기서는 3.057103549994738077이다.)

* Newton-Raphson 방법

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 3.000000000000000000e+00 | 9.8612288668e-02 |
| 1 | 3.059167373200865700e+00 | 3.6927457761e-03 |
| **2** | **3.057106054691599800e+00** | **4.4761505977e-06** |

* Secant 방법

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 4.000000000000000000e+00 | 2.6137056389e+00 |
| 1 | 2.419218645889003000e+00 | 7.0770034137e-01 |
| 2 | 2.756039712206013200e+00 | 4.4219871598e-01 |
| 3 | 3.317022504001122300e+00 | 5.3548073166e-01 |
| 4 | 3.009768959371032200e+00 | 8.2229966821e-02 |
| 5 | 3.050670709725761900e+00 | 1.1452530996e-02 |
| 6 | 3.057289041659920800e+00 | 3.3152841948e-04 |
| 7 | 3.057102843928874800e+00 | 1.2618098428e-06 |
| **8** | **3.057103549917539600e+00** | **1.3796119802e-10** |

실습문제 1-1 은 로 인해 에서만 정의되고 약 1.412391172023884516과 약3.057103549994738077 두 개의 근을 갖는다.  
실험 내용과 비교하면, Newton-Raphson method , Secant method 모두 연산횟수가 많지 않아도 근에 근접하는 것을 알 수 있다.  
만약, 실제 근의 값을 모르는 상황에서 위의 방법을 신뢰할 수 있을까?   
 Newton-Raphson method 의 경우 임의의 좌표 에서의 함수 값인 에 접선을 그었을 때 그 접선의 절편을 으로 하는 과정을 반복하여 값이 점점 찾고자 하는 근에 수렴하도록 하는 방법이다. 근에서의 접선의 방정식의 절편은 곧 함수의 근이기 때문에 위 식을 적용했을 때 이 되므로 임의의 좌표 에서 출발하여 알고리즘을 진행하다 보면 이 됨을 짐작할 수 있다. 즉 값이 커질수록 은 근에 수렴하게 된다  
 그러나, Newton-Raphson method 의 경우 아래와 같은 문제점을 가지고 있다.

* 값이 0인 경우 더 이상 진행할 수 없다.
* 근에서 함수를 미분할 수 없는 경우 이 방법을 사용할 수 없다.
* 이 특정 값들을 무한히 반복하는 경우가 있다. 예를 들어, 함수 는 0 또는 1에서 이 알고리즘을 시작했을 경우 값이 0과 1을 반복한다.[[1]](#footnote-2)

위와 같이 Newton-Raphson method 를 사용할때 문제가 있는 경우, 연산을 지속으로 수행하여도, 근의 값을 구하지 못한다

그러므로 Newton-Raphson 방법이 정상적으로 적용이 되었다면 값이 커질수록 의 값이 충분히 작아지고 의 값이 충분히 작아져야 한다. 즉 다음 조건을 **모두 만족하면** Newton-Raphson 방법을 통해 구한 값을 근이라고 신뢰할 수 있다.

조건 1. (이 실습 문제에선 0.000001사용)

조건 2. (이 실습 문제에선 0.00001 사용)

이 때, 와 값은 사용자가 원하는 정밀도에 따라 값을 조정할 수 있는 충분히 작은 값이다. 또한 충분히 많은 횟수를 반복했음에도 위 조건을 만족하지 못한다면 근을 구하지 못했다고 판단할 수 있다.

즉 실습 문제 1-1에 주어진 방정식의 근을 Newton-Raphson 방법으로 구한 값이 위 두 조건을 만족하므로 **근으로 신뢰할 수 있다**

Secant method 는 Newton-Raphson method 와 달리, 접선의 기울기를 이용하는 것이 아닌, 두 점의 기울기를 활용하여, 계산하는 방법으로 실제로 미분을 하지 않아도 된다는 장점이 있지만, Newton-Raphson method에 비해 오차가 더 크게 작용을 하기 때문에, 연산 횟수가 많다는 단점이 존재한다.  
Secant 방법은 과 두 점을 이은 선의 절편을 으로 한다 이를 식으로 표현하면 아래와 같이 표현이 가능하다.

Secant 방법도 Newton-Raphson 방법과 마찬가지로 다음 조건을 모두 만족하면 Secant 방법을 통해 구한 값을 근으로 신뢰할 수 있다  
조건 1. (이 실습 문제에선 0.000001사용)

조건 2. (이 실습 문제에선 0.00001 사용)

즉 실습 문제 1-1에 주어진 방정식의 근을 Secant 방법으로 구한 값이 위 두 조건을 만족하므로 근으로 신뢰할 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 방법은 의 형태로 실제 정답과의 오차가 감소하는 반면, Secant 방법은 의 형태로 그 오차가 감소한다고 알려져 있다. 즉 Newton-Raphson method의 경우 차수 2의 속도로, Secant 방법은 차수 1.62의 속도로 오차가 감소하므로 Newton-Raphson method가 Secant method에 비해 빠르게 근에 접근해야 한다. 위의 실험에서 나온 결과 또한 Newton-Raphson method가Secant method에 비해 훨씬 빠른 속도로 값을 구해냈다.

* 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 방법 | | Secant 방법 | | |
| 초기값() | 연산횟수 | 초기값() | 초기값() | 연산횟수 |
| 100.0 | 10 | 100.0 | 99.0 | 14 |
| 200.0 | 11 | 200.0 | 199.0 | 16 |
| 300.0 | 12 | 300.0 | 299.0 | 16 |
| 1000.0 | 14 | 1000.0 | 999.0 | 19 |
| 10000.0 | 17 | 10000.0 | 9999.0 | 24 |
| 100000.0 | 20 | 100000.0 | 99999.0 | 29 |
| 1000000.0 | 24 | 1000000.0 | 999999.0 | 33 |
| 10000000.0 | 27 | 10000000.0 | 9999999.0 | 38 |
| 100000000.0 | 30 | 100000000.0 | 99999999.0 | 43 |
| 1000000000.0 | 33 | 1000000000.0 | 999999999.0 | 48 |

두 방법 모두 초기값이 주어진 식의 해에 가까울수록 연산횟수가 적고 멀어질수록 연산횟수가 많아지는 경향을 보인다. 하지만 경우에서 해를 구할 수 있는 것은 아니다. 하지만 예외가 존재한다. 예를 들어 한 점에서의 미분 값이 0이 되는 경우가 생기면 Newton-Raphson 방법은 더 이상 이용할 수 없다. 문제에 주어진 함수는 값이 (약 2.22474487139158904909864) 일 때 미분 값이 0인데 프로그램에 이 값을 넣고 수행을 해보면 근의 값을 구할 수 없었다.

Secant 방법도 예외가 존재한다. 어떤 과 에 해당 하는 함수 값을 이었을 때 그 직선의 기울기가 0이 되는 경우가 생기면 Secant 방법을 이용해 근을 구할 수 없다. 값이 약 0.0024545491704989, 약 5.419052648328639272 일 때 주어진 함수의 값이 약 10으로 같은데 이 두 값을 프로그램에 넣고 수행해 보면 근의 값을 구할 수 없었다.

결론적으로 두 방법 모두 일부 예외를 제외하면 대부분의 임의의 초기값에 대해서 아주 빠르게 근으로 수렴한다

1. **실습 문제 1-3**
   1. *두 방법 각각에 대하여 공정한 비교가 가능하도록 초기값 을 100.0부근에서 적절히 동일하게 설정한 후, 근에 충분히 수렴할 때까지 반복을 하면서, 각 줄에 a) 반복 회수 (0부터 시작), b) 현재까지 추정한 값 , 그리고 c) 정확한 값에 대한 절대 오차 를 출력하라. 에 대한 정확한 값으로 다음 값을 사용하고,*

*double 타입의 숫자를 출력할 때에는 십진수 유효숫자 15개를 출력하라.*

두 방법에 대해 공정한 비교를 하기 위해서는 Newton-Raphson method 의 경우는 순간 기울기, Secant method 의 경우 평균기울기의 값이 비슷하게 시작을 하여야 한다.  
이를 위하여 Newton-Raphson 방법의 초기값은 으로 하였고, Secant 방법의 초기값은 로 하였다

* 1. *보고서에 각 방법에 대한 결과를 첨부하고, 각 방법의 근에 대한 수렴 속도가 이론적으로 습득한 내용과 일치하는지를 비교 분석하여 기술하라.*
* Newton-Raphson 방법

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 0 | 1.000000000000000e+02 | 9.858578643762691e+01 | -nan(ind) |
| 1 | 5.001000000000000e+01 | 4.859578643762691e+01 | 2.861669273034480e-02 |
| 2 | 2.502499600079984e+01 | 2.361078243842674e+01 | 4.373414327850527e-02 |
| 3 | 1.255245804674590e+01 | 1.113824448437281e+01 | 6.643209243240769e-02 |
| 4 | 6.355894694931140e+00 | 4.941681132558045e+00 | 9.954778231608553e-02 |
| 5 | 3.335281609280434e+00 | 1.921068046907339e+00 | 1.443662595386860e-01 |
| 6 | 1.967465562231149e+00 | 5.532519998580538e-01 | 1.921249378290680e-01 |
| 7 | 1.492000889689723e+00 | 7.778732731662807e-02 | 2.029427785334303e-01 |
| 8 | 1.416241332038944e+00 | 2.027769665848611e-03 | 1.269832052647335e-01 |
| **9** | **1.414215014050053e+000** | **1.451676957975323e-06** | **3.345798759650934e-02** |

* Secant 방법

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 0 | 9.998999999999999e+01 | 9.857578643762690e+01 | -nan(ind) |
| 1 | 5.000999949999716e+01 | 4.859578593762406e+01 | 2.862139546707708e-02 |
| 2 | 3.334999911119873e+01 | 3.193578554882564e+01 | 5.915450908187882e-02 |
| 3 | 2.003159149107379e+01 | 1.861737792870069e+01 | 6.807517987986750e-02 |
| 4 | 1.255214673941698e+01 | 1.113793317704388e+01 | 9.761950784479080e-02 |
| 5 | 7.778096976695405e+00 | 6.363883414322310e+00 | 1.282031658826897e-01 |
| 6 | 4.900669957337246e+00 | 3.486456394964151e+00 | 1.739218914556451e-01 |
| 7 | 3.164178850173626e+00 | 1.749965287800531e+00 | 2.314019257090062e-01 |
| 8 | 2.170728385432810e+00 | 7.565148230597150e-01 | 3.055693276877713e-01 |
| 9 | 1.662366833197602e+00 | 2.481532708245064e-01 | 3.899740851214191e-01 |
| 10 | 1.463190072754846e+00 | 4.897651038175055e-02 | 4.683186269376010e-01 |
| 11 | 1.418102047404884e+00 | 3.888485031789291e-03 | 5.152288034514260e-01 |
| 12 | 1.414279659255878e+00 | 6.609688278236270e-05 | 5.305626564290439e-01 |

두 방법 모두 빠르게 로 수렴하는 모습을 보인다. 둘 중 Newton-Raphson method 의 경우가, Secant method 에 비해 빠른 속도를 보이고 있다.

그리고 Newton-Raphson 방법이 조금 더 빠른 속도를 보인다. 이론에 따르면 각 방법의 값(혹은 )이 연산을 수행할수록 특정 값에 수렴하는 모습을 보여야 한다 Newton-Raphson method 는 의 값이 약 3.53으로 근접하는 모습을 보이고 Secant 방법은 의 값이 약 5.43으로 근접하는 모습을 보인다. 즉 이론에 따르는 모습을 알 수 있다.

1. **숙제 1**
   1. *프로그램이 완성되면, 실습 시간에 사용한 세 개의 함수 그리고 에 대해 적절한 초기 구간을 사용하여 올바르게 근에 수렴하는지에 대한 분석 내용을 보고서에 기술하라.*

Bisection method는 범위 에서 시작해 또는 와 같이 이분법적으로 그 범위를 줄여가며 근을 구하는 방법이다. 이 방법은 중간값 정리를 활용한 방법으로, 임의의 어떤 와 의 부호가 다를 경우 폐구간 에 적어도 한 개의 근이 존재한다는 특성을 사용한다. 즉, 초기값 와 의 부호가 달라야 하며 다음 구간을 정하는 알고리즘은 아래와 같다.

* 인 경우,
* 인 경우,
* / 초기 범위 : [2.0, 100.0]

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 5.100000000000000000e+01 | 2.3970681744e+03 |
| 1 | 2.650000000000000000e+01 | 5.9697285527e+02 |
| 2 | 1.425000000000000000e+01 | 1.4740574309e+02 |
| 3 | 8.125000000000000000e+00 | 3.5420679272e+01 |
| (중략) | | |
| 18 | 3.057103549994737400e+00 | 1.5543122345e-15 |
| 19 | 3.057103549994740100e+00 | 2.8865798640e-15 |
| 20 | 3.057080268859863300e+00 | 4.1605147233e-05 |
| **21** | **3.057103633880615200e+00** | **1.4991246600e-07** |

* / 초기 범위 : [-100.0, 100.0]

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 0.000000000000000000e+00 | 3.057103549994737900e+00 |
| 1 | -5.000000000000000000e+01 | 5.305710354999473600e+01 |
| 2 | -2.500000000000000000e+01 | 2.805710354999473900e+01 |
| 3 | -1.250000000000000000e+01 | 1.555710354999473700e+01 |
| (중략) | | |
| 21 | -9.999752044677734375e-01 | 1.8189760301e-04 |
| 22 | -9.999990463256835938e-01 | 8.2529651089e-06 |
| 23 | -1.000010967254638672e+00 | 7.8569353909e-05 |
| **24** | **-1.000005006790161133e+00** | **3.5158194406e-05** |

* / 초기 범위 : [2.0, 100.0]

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 5.100000000000000000e+01 | 5.4128334384e+06 |
| 1 | 2.650000000000000000e+01 | 3.1646103840e+05 |
| 2 | 1.425000000000000000e+01 | 1.7090829806e+04 |
| 3 | 8.125000000000000000e+00 | 7.4809762852e+02 |
| (중략) | | |
| 21 | 4.399984836578369141e+00 | 1.2109202486e-04 |
| 22 | 4.399996519088745117e+00 | 2.7798395870e-05 |
| 23 | 4.400002360343933105e+00 | 1.8849780709e-05 |

초기구간에 대하여 적절한 값이 입력이 된다면, Bisection method를 이용하여 해를 구할 수 있다.

* 1. *Bisection 방법의 수렴 속도는 선형적인, 즉 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.*

모든 방법에 대해서 근으로 신뢰하기 위한 조건으로 다음을 사용하였다.

조건 1.

조건 2.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Newton-Raphson 방법  초기값 = 3.0 | Secant 방법  초기값 = [2.0, 4.0] | Bisection 방법  초기값 = [2.0, 4.0] |
| 근에 수렴하기 까지의  연산 횟수 | 3 | 8 | 17 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Newton-Raphson 방법  초기값 = 0.75 | Secant 방법  초기값 = [0.5, 1.0] | Bisection 방법  초기값 = [0.5, 1.0] |
| 근에 수렴하기 까지의  연산 횟수 | 3 | 6 | 15 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Newton-Raphson 방법  초기값 = 4.0 | Secant 방법  초기값 = [3.5, 4.5] | Bisection 방법  초기값 = [3.5, 4.5] |
| 근에 수렴하기 까지의  연산 횟수 | 11 | 9 | 16 |

Newton-Raphson 방법은 의 형태로, Secant 방법의 수렴 속도는 의 형태로, Bisection 방법은 의 형태로 근에 수렴 하는 오차가 감소하기 때문에 연산 횟수는 대체로 와 같은 형태여야 한다. 세 번째 함수에서 Newton-Raphson 방법이 Secant 방법보다 연산 횟수가 두 번 더 많은 것으로 측정되었지만 이는 감수할 만한 오차이다. 그러므로 대체로 이론적인 수렴 속도를 따르고 있다.

1. **숙제 2**

문제에 상수값을 대입하게 되면, 아래와 같은 식을 얻을 수 있다.

위에서 구한 , 를 Newton-Raphson method를 이용하여 근을 구하면, 아래와 같다.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 | 1.000000000000000000e+00 | 2.4663266581e+01 |
| 1 | 5.466280157685478000e-01 | 1.4773541846e+00 |
| 2 | 5.761259392915085700e-01 | 3.4168393129e-02 |
| 3 | 5.754733105254510300e-01 | 1.5437980224e-05 |
| **4** | **5.754730153875763400e-01** | **3.1725733152e-12** |

Newton-Raphson method로 구한 근의 값 5.754730153875763400e-001을 60분법으로 바꾸면 약 32.97217500537520366도가 나온다. 문제에 언급된 33도와 아주 유사한 값이다

1. [↑](#footnote-ref-2)