『GPS수신기위치계산문제』

수치 컴퓨팅 실험3: 공개 소프트웨어를 사용한 문제 풀이

고급소프트웨어실습 3분반

20151591 이지현

**숙제 3-1**

## (iii) 위 두 방법을 사용하여 문제를 풀면서 알게 된 내용을 보고서 형식의 파일에 기술하여 제출하라.

각각의 파일 0~2에 각각 초기값을 (0, 0, 0, 0)을 넣었을 때의 결과값은 다음과 같았다.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | hybrj 방식 | hybrd 방식 |
| 0 | x | 2278.123943 -8235.053883 -5979.929995 46.428118 | 2278.123943 -8235.053883 -5979.929995 46.428118 |
| y | -0.000048 -0.000048 -0.000048 -0.000048 | 0.000016 0.000016 0.000016 0.000016 |
| 1 | x | -249.485423 -6262.816152 -8776.418070 49.042321 | -249.485423 -6262.816152 -8776.418070 49.042321 |
| y | 0.000002 0.000002 0.000002 0.000002 | -0.000001 -0.000001 -0.000001 -0.000001 |
| 2 | x | 7709.763830 -6588.540954 1367.364138 51.149464 | 7709.763830 -6588.540954 1367.364138 51.149464 |
| y | 0.000005 0.000005 0.000005 0.000005 | 0.000003 0.000003 0.000002 0.000003 |

원래 우리가 실습을 진행할 때 수식에 넣었던 tol의 값은 0.0000001이었으나, 이번 수식을 처리함에 있어 동일 tol값을 넣을 시 수식의 값이 제대로 나오지 않는 문제가 있었다. 이를 해결하고자 tol의 값을 0.0000001^2의 값을 대입하였더니 각 수식에 대한 y의 값이 0에 매우 수렴함을 알 수 있었다. 따라서 정확도가 중요하다면 tol 위치에 대입하는 값을 매우 작게 설정해주면 된다는 것을 알 수 있었다. 식의 정확도는 우리가 구한 x값에 대해 식의 결과가 0이 나오는지 확인하였고, 그 결과가 매우 0에 유사함을 볼 수 있었다. 또한 두 방식에 대해 나온 결과인 x값이 매우 유사하므로 이 두 방식은 상당히 정확한 위치를 내보냄을 알 수 있었다.

또한 자코비 행렬을 사용하는 hybrj 방식의 경우가 정확도가 더 높다고 생각했었으나, 생각외로 같은 tol과 같은 초기값을 가지는 경우 hybrd 방식의 정확도가 높은 결과를 얻을 수 있었다. 이는 어떤 것이 원인이 되는지는 정확하게 추측할 수 없지만, 추측하건데 같은 tol 값을 가지는 두 방식을 수행하면 수렴 속도가 hybrd가 더 빠를 것으로 예측된다.

**숙제 3-2**

x[0] = x, x[1] = y, x[2] =z, x[3] = w 로 두고 rf 함수를 작성하였다.

결과 값(roots\_found 3-2.txt에 저장)은 다음과 같다.

fvec의 값은 근의 정확도를 보인다.

|  |
| --- |
| roots\_found 3-2.txt |
| x : 0.455790, y : -0.945579, z : 1.041577, w : -0.958423  fvec[0] : 0.000000, fvec[1] : 0.000000, fvec[2] : 0.000000, fvec[3] : 0.000000 |

**숙제 3-3**

이 프로그램을 만들기 위하여 PORTRAN 함수 중 lmdif1 함수를 사용했다. 이 함수의 목적은 Levenberg-Marquardt 알고리즘을 수정하여 n개의 변수에서 m개의 비선형 함수 제곱의 합을 최소화하는 것이다. 주어진 함수로부터 Jacobian 방법으로 전방 차이 근사값으로 계산한다. 이 함수의 사용법은 아래와 같다[[1]](#footnote-1)

|  |
| --- |
| my\_solver.h |
| void **lmdif1\_**(void(\*fcn)(int \*m, int \*n, double \*x, double \*fvec, int \*iflag),  int \*m, int \*n, double \*x, double \*fvec,  double \*tol, int \*info, int \*iwa, double \*wa, int \*lwa); |

|  |
| --- |
| roots\_found 3-3.txt |
| x = 14.408460399767620  y = -0.120928506313017  f0(x, y) = 0.000000000000003  f1(x, y) = 0.000000000000000 |

결과를 보았을 때, 나온 결과값 x, y가 원래 기존 함수 f0, f1에 대입하였을 때 정상적으로 0에 근사한 값이 나오는 것으로 보아 근에 대한 정확도가 신뢰할 수 있는 정도임을 알 수 있다.

1. [↑](#footnote-ref-1)