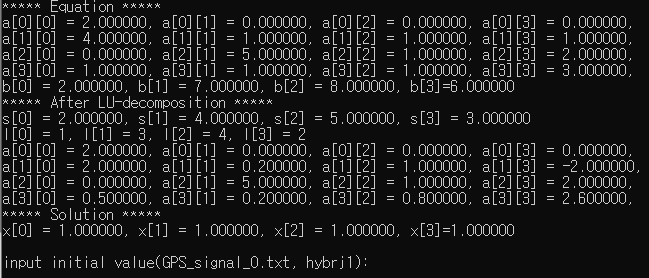
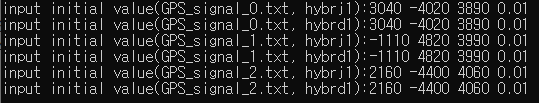
[고소실\_6주차과제]2반\_20161595\_배성현

**실습/과제 프로그램의 구동 방법**

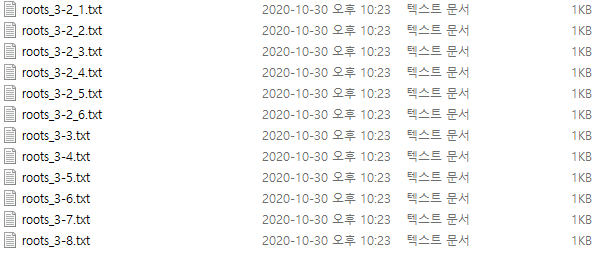


처음 프로그램을 실행하면, 먼저 실습에 관한 내용들이 진행되고 위의 그림과 같이 켜지게 되는데 여기서부터 과제 3-1의 초기값(x10, x20, x30, x40)을 입력하여 주면 된다. 총 6번의 initial value값을 입력하라고 뜨는데 아래와 같이 순서대로 GPS\_signal\_0.txt의 hybrj1방식에 대한 초기값, hybrd1방식에 대한 초기값 그리고, GPS\_signal\_1.txt의 hybrj1방식에 대한 초기값, hybrd1방식에 대한 초기값, GPS\_signal\_2.txt의 hybrj1방식에 대한 초기값, hybrd1방식에 대한 초기값을 입력하여 주면 된다.

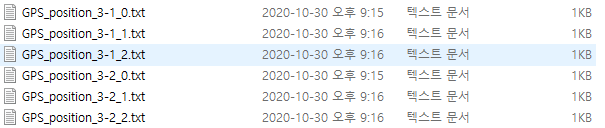


(과제 3-1에 대한 초기값 입력)

이후 별다른 조작없이 과제 3-2, 3-3, 3-4에 대한 코드가 수행되게 되며 프로그램이 완료되게 된다. 그리고 각각의 실습 및 과제에 대한 결과는 아래의 그림과 같이 프로젝트의 폴더에 파일로써 저장되므로 해당 파일을 통하여 결과를 확인 할 수 있다. (단 과제 3-4의 경우에는 프로그램 실행 전 linear\_system\_3-4.txt파일이 미리 프로젝트 폴더에 존재하여야 한다.)



(실습에 대한 결과 파일)



(과제 3-1에 대한 결과 파일[3-1의 경우 hybrj1방식, 3-2의 경우 hybrd1방식])

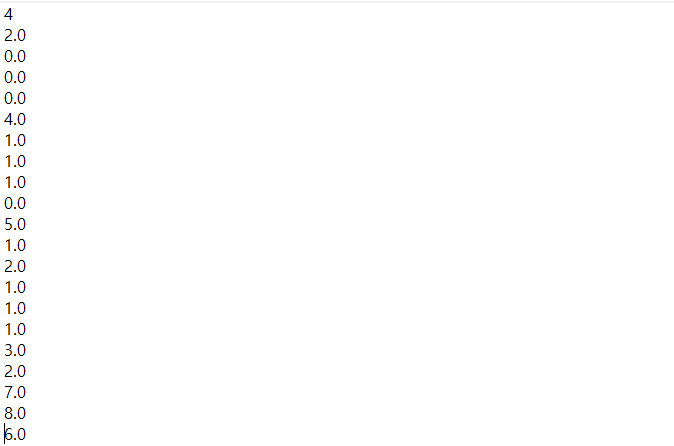


(과제 3-2에 대한 결과 파일)



(과제 3-3에 대한 결과 파일)



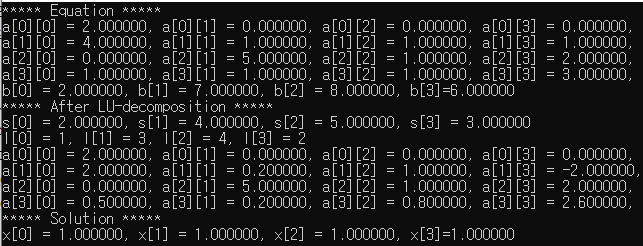


(과제 3-4에 대한 input 파일)



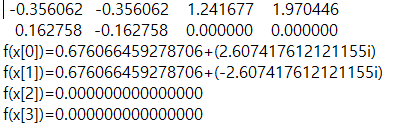
(과제 3-4에 대한 결과 파일)

**실습 3-1**



2x1 =2, 4x1+x2+x3+x4=7, 5x2+x3+2x4=8, x1+x2+x3+3x4=6에 대하여 실험을 진행하였고 이 때 해는 1.0, 1.0, 1.0, 1.0이 나와야 하는데 위의 Solution 부분을 보게 되면 해가 1.0, 1.0, 1.0, 1.0으로 해를 올바르게 구하였음을 확인할 수 있다.

**실습 3-2**



polynomial\_3-2\_3.txt에 대한 결과 파일

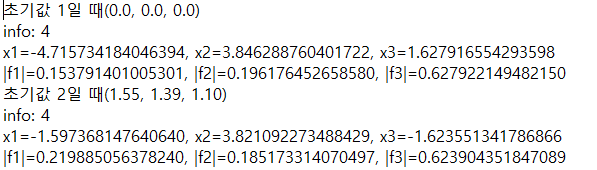
polynomial\_3-2\_3.txt를 받아 rpoly를 이용하여 모든 근을 구하고, 복소수 연산에 유의하여 해당 근을 fx에 대입하였을 때, 위와 같이 결과를 얻을 수 있었고 f(x[2])와 f(x[3])를 보았을 때 0 값을 가지는 것을 보아 근을 잘 구했다는 것을 확인할 수 있다.



polynomial\_3-2\_5.txt에 대한 결과 파일

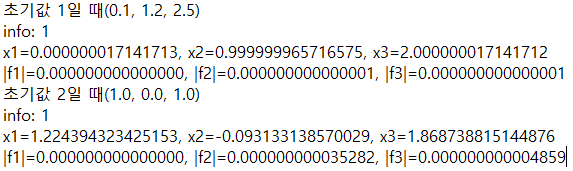
또한 polynomial\_3-2\_5.txt를 받아 rpoly를 구하였을 때 근을 구할 수 없으므로 wrong이 나와야 하는데 위의 결과 파일을 보았을 때, wrong이 잘 나온 것을 확인 할 수 있다.

**실습 3-3**

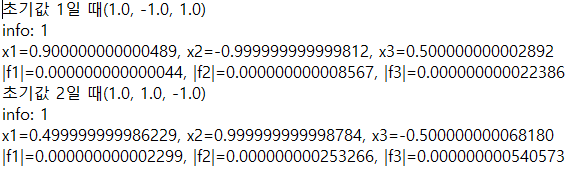


실습 3-3에서는 위와 같이 info값이 4가 나와 근을 제대로 구하지 못하는 것을 확인 할 수 있었다.

**실습 3-4, 3-5**



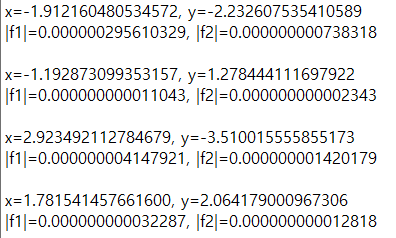
(3-4 결과 파일)



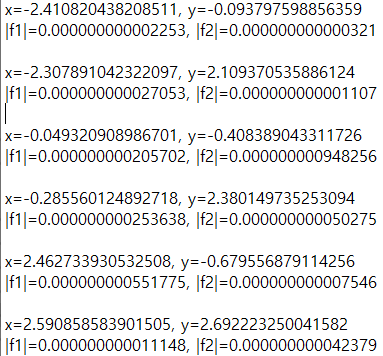
(3-5 결과 파일)

실습 3-4와 3-5에서는 위와 같이 info값 1을 통하여 근을 잘 구하였음을 알 수 있고, 또 이렇게 구한 근을 f1, f2, f3에 대입하여 절대값을 보았을 때 0에 매우 근사하는 것을 확인할 수 있으므로, 구한 근이 정확하다는 것을 알 수 있다.

**실습 3-6, 3-7**



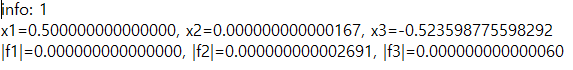
(3-6 결과 파일)



(3-7 결과 파일)

실습 3-6와 3-7에서는 [-4,4]x[-5,5]사이의 근을 구하여야 하는데 실습 3-6의 경우에는 근이 4개, 실습 3-7의 경우에는 근이 6개가 나와야 한다. 위의 결과 파일을 보면 각각 4개와 6개의 근을 잘 구한 것을 확인할 수 있고 또 모든 근이 [-4,4]x[-5,5]의 범위에 잘 들어와 있는 것을 확인할 수 있다. 또 이렇게 구한 근을 f1, f2에 대입하여 절대값을 보았을 때 0에 매우 근사하는 것을 확인할 수 있으므로, 구한 근이 정확하다는 것을 알 수 있다.

**실습 3-8**



실습 3-8에서는 위와 같이 info값 1을 통하여 근을 잘 구하였음을 알 수 있고, 또 이렇게 구한 근이 실제 근인 (0.5, 0.0, −0.52359877)에 수렴하는 것을 확인 할 수 있다. 또한 근을 f1, f2, f3에 대입하여 절대값을 보았을 때 0에 매우 근사하는 것을 확인할 수 있으므로, 구한 근이 정확하다는 것을 알 수 있다.

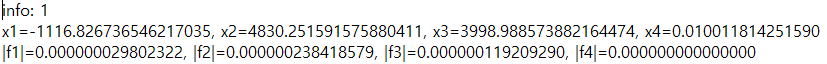
**과제 3-1**



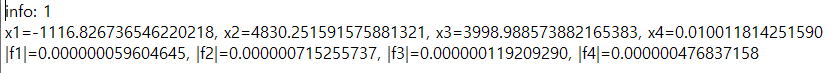
(GPS\_signal\_0.txt의 결과파일[hybrj1])



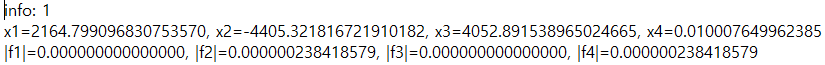
(GPS\_signal\_0.txt의 결과파일[hybrd1])



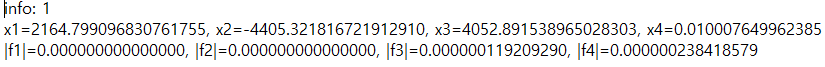
(GPS\_signal\_1.txt의 결과파일[hybrj1])



(GPS\_signal\_1.txt의 결과파일[hybrd1])



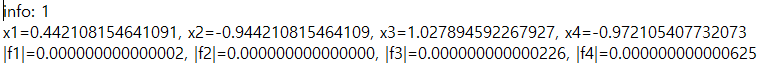
(GPS\_signal\_2.txt의 결과파일[hybrj1])



(GPS\_signal\_2.txt의 결과파일[hybrd1])

위의 그림과 같이 각각의 결과파일의 x1, x2, x3, x4에 따른 |f1|, |f2|, |f3|, |f4|들을 보았을 때, 모두 0에 매우 근사한 값을 가지는 것을 확인 할 수 있다. 따라서 실제로 정확한 위치를 잘 찾아냈음을 확인할 수 있다. 위의 문제를 두 방법을 사용하였을 때, 두 방법은 이미 검증되어 있는 포트란 공개 소프트웨어이기 때문에 굉장히 쉽고 편하게 또 정확하게 근의 값을 얻을 수 있었다. 두 방법을 사용하면서 hybrj1의 경우에는 함수와, 야코비 행렬(1차 미분 정보)을 이용하는 방법이며, hybrd1의 경우에는 야코비 행렬을 이용하지 않고 다변수 함수 값을 계산해 주는 함수만을 이용하는 방법이라는 것을 알 수 있었다. 또한 코드를 짜는 것에 있어서는 hybrj1에 비하여 hybrd1의 방법이 야코비 행렬을 만들어줄 필요가 없어서, 즉 각각의 변수에 대하여 1차 미분 정보를 계산할 필요가 없어서 더 편하였다. 하지만 메모리 측면에서 보았을 때, hybrd1의 방법이 임시적으로 사용하는 저장소인 wa의 공간을 많이 사용하여 hybrj1방법에 비하여 wa에 더 큰 공간을 할당해 줘야 한다는 단점이 있었고, 야코비 행렬을 구해 주지 않아도 되지만, hybrd1 내부에서 야코비 행렬을 근사적으로 계산하여 사용하기 때문에 hybrj1방법에 비하여 더 느리다는 단점이 있었다. 그리고 프로그램의 수행 결과 최종적으로 hybrj1의 방법과 hybrd1의 방법 모두 인공위성의 위치를 잘 구하였음을 확인 할 수 있었다.

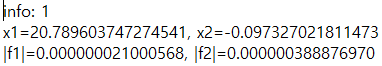
**과제 3-2**



[과제 3-2에 대한 결과파일]

Hybrd1의 방법을 통하여 3-2에 해당하는 비선형 방정식의 우변을 모두 좌변으로 이항하여 f1, f2, f3, f4를 만들고 이에 해당하는 근을 찾아주었는데, 해당 근을 f1, f2, f3, f4에 대입하여 결과를 출력하였을 때, |f1|, |f2|, |f3|, |f4| 모두 0에 매우 근사한 값을 가지는 것을 확인 할 수 있기 때문에 정확한 근을 구했다는 것을 알 수 있다.

**과제 3-3**



[과제 3-3에 대한 결과파일]

Hybrj1의 방법을 통하여 3-3에 해당하는 비선형 방정식의 우변을 모두 좌변으로 이항하여 f1, f2를 만들고 이에 해당하는 근을 찾아주었는데, 해당 근을 f1, f2에 대입하여 결과를 출력하였을 때, |f1|, |f2| 모두 0에 매우 근사한 값을 가지는 것을 확인 할 수 있기 때문에 정확한 근을 구했다는 것을 알 수 있다.

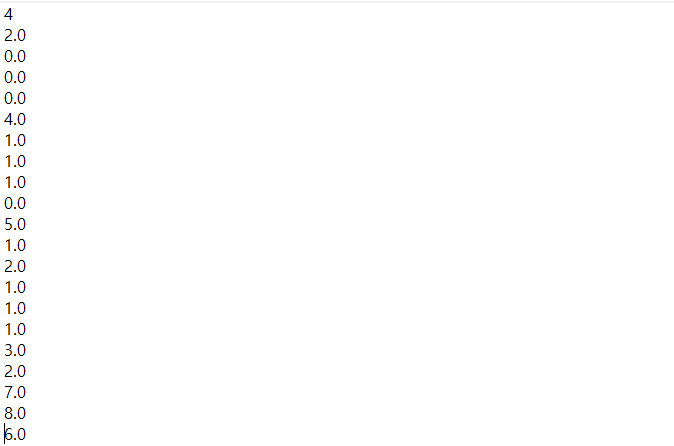
**과제 3-4**

2x1 =2

4x1+x2+x3+x4=7

5x2+x3+2x4=8

x1+x2+x3+3x4=6



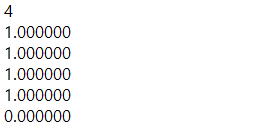
과제 3-4의 Input File

먼저 실습시간에 조교님께서 제공하여 주셨던 선형 방정식을 바탕으로 위와 같이 input 파일을 만들었고, gespp()와 solve()를 아래와 같이 사용하여 근을 구하도록 하였다.



program3\_4.cpp의 일부

해당 선형 방정식의 근은 1.0, 1.0, 1.0, 1.0인데 과제 3-4의 output file을 확인하여 보았을 때 아래와 같이 근의 값들이 1.0, 1.0, 1.0, 1.0이 나오는 것을 확인 할 수 있었다. 또한 오차에 대한 척도를 계산하여 가장 마지막 줄에 출력되도록 하였는데, 이 값을 확인했을 때 0.000000으로, 매우 정확하게 근을 잘 구하였음을 확인 할 수 있다.



과제 3-4의 Output File