고급소프트웨어실습I

4주차 보고서

20171646 박태윤

- 구동 방법 및 간략한 소개

빌드 후 실행을 하면 입력을 받습니다. 실습1, 실습2, 실습4, 과제1, 과제2 순서대로 입력을 받습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위의 실행 예시를 참고하여 입력을 받으시면 될 거 같습니다. 현재 메인에서 bisection method에 대한 함수는 f1, 즉 실습 1에 해당하는 함수로 지정이 되어있습니다. Secant Method로 입력을 받는 경우에는 한 번에 2개의 수를 입력 받으면 됩니다.

-실습1

(결과)

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

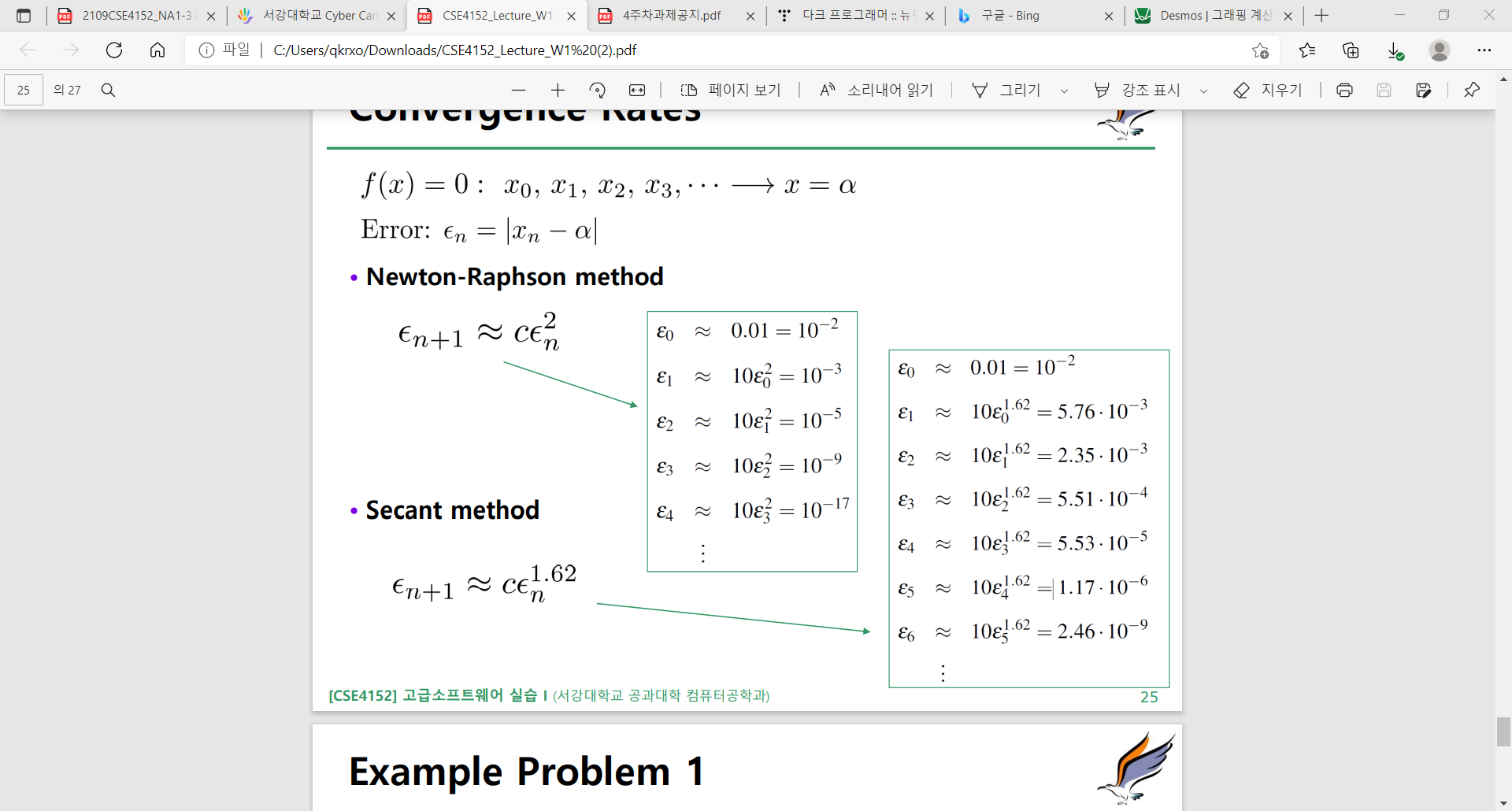
(1) 두 방법에 대한 초기값을 각각 x0 = 3.0과 x0 = 2.0, x1 = 4.0으로 설정하여 자신이 작성한 프로그램을 수행시켜 자신의 프로그램이 원하는 근을 정확 히 찾고 있는지 분석하라. 과연 근이 맞는지 어떻게 확인할 수 있을까? 논˙ 리˙ 적˙ 으˙ 로˙ 타˙ 당˙ 한˙ 방˙ 법˙ 으˙ 로˙ 분˙ 석˙ 한˙ 내˙ 용˙ 을˙ 보고서에 기술하라.

테이블이(가) 표시된 사진

자동 생성된 설명

위의 그래프와 실제로 실습에서 뉴턴-랩슨, 시컨트 방법으로 구한 근이 매우 유사한 값을 나타내는 것을 확인할 수 있다. 뉴턴-랩슨 방법에서는 근과 어느 정도 근접한 x0=3.0이라는 값을 입력하여 현재 구한 함수 값이 델타(0.000001)보다 작은 경우 또는 반복 횟수가 50이 넘어가는 경우 또는 이전에 구한 x값과 현재 구한 x값의 차가 엡실론(0.00001)보다 작아 의미가 있는 전진을 하지 않는 경우 계산을 종료하였다. 시컨트 방법에서도 마찬가지로 x0=2.0, x1=4.0이라는 값을 입력하여 앞의 조건과 동일한 조건 안에서 근을 구하는 계산을 진행하였다. |f(xn1)|의 값을 보면 이 값이 매우 작은 값, 즉 0과 매우 근접한 값으로 수렴하고 있는 것을 확인할 수 있기에 완전 정확한 근이라고는 할 수 없지만 실제 근과 매우 유사한 값, 근사치를 얻었다고 할 수 있다.

(2) 위에서 산출한 결과를 볼 때, 각 방법의 근의 수렴 속도가 과연 앞˙ 에˙ 서˙ 설˙ 명˙ 한˙ 속˙ 도˙ 로˙ 수˙ 렴˙ 하˙ 는˙ 지˙ 비˙ 교˙ 분˙ 석˙ 한˙ 후˙ 그 결과를 보고서에 기술하라.



자료에서 설명이 되었던 것처럼, 뉴턴-랩슨 방식이 시컨트 방식보다 해에 근접하는 속도가 좀 더 빠른 것을 확인할 수 있다. 뉴턴-랩슨 방식에서 4번 안에 끝나는 경우 시컨트 방식에서는 각각 10번, 8번으로 계산 반복 횟수가 더 많다.

(3) 위에서 제시한 초기값 외에 임의의 초기값들을 사용하여 자신이 작성한 프 로그램을 수행한 후, 이 두 방법이 항˙ 상˙ 임˙ 의˙ 의˙ 초˙ 기˙ 값˙ 에˙ 대˙ 해˙ 빠˙ 르˙ 게˙ 수˙ 렴˙ 하˙ 는˙ 지˙ 보고서에 기술하라.

텍스트이(가) 표시된 사진

자동 생성된 설명

앞선 경우보다 훨씬 큰 값을 입력하였는데, 유사한 근을 잘 구한 것을 확인할 수 있다.

테이블이(가) 표시된 사진

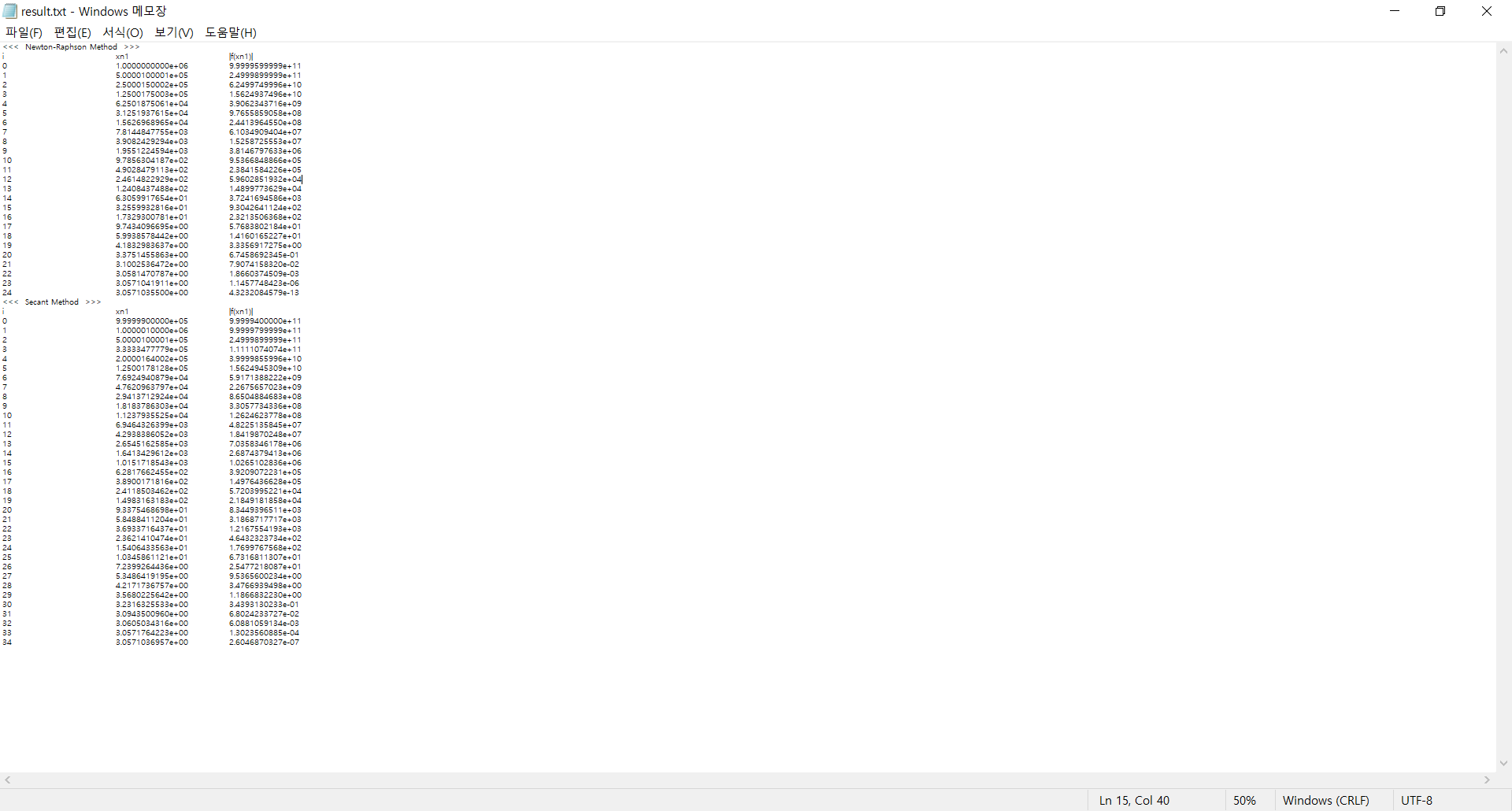
자동 생성된 설명

그에 따라 계산 반복 횟수가 좀 더 늘어난 것을 확인할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명

훨씬 큰 값을 입력했을 때도 근의 근사치를 구하는 것을 확인할 수 있다.



매우 큰 값임에도 뉴턴-랩슨은 24번, 시컨트 방식은 34번으로 꽤나 근을 빨리 구한다는 것을 확인할 수 있다.

-실습2

(결과)

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

-실습4

(결과)

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

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

double type을 사용하여 구한 결과는 2.7182815563이고 float type을 사용하여 구한 결과는 2.7182815075로 소수점 8번째 자리부터 차이를 나타내는 것을 확인할 수 있고, double로 구한 결과가 좀 더 실제 e값에 근접한 것을 확인할 수 있다. float는 4byte로 7자리, double은 8byte로 15자리 까지 표현이 가능하기에, double로 계산을 하는 것이 좀 더 정밀한 결과를 얻을 수 있다.

-과제1

(결과)

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

(1) 프로그램이 완성되면, 실습 시간에 사용한 세 개의 함수 f1(x), f2(x), 그리 고 f3(x)에 대해 적절한 초기 구간을 사용하여 올˙ 바˙ 르˙ 게˙ 근˙ 에˙ 수˙ 렴˙ 하˙ 는˙ 지˙ 에˙ 대˙ 한˙ 분˙ 석˙ 내˙ 용˙ 을 보고서에 기술하라.

(f1(x)를 사용)

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

(f2(x)를 사용)

텍스트이(가) 표시된 사진

자동 생성된 설명

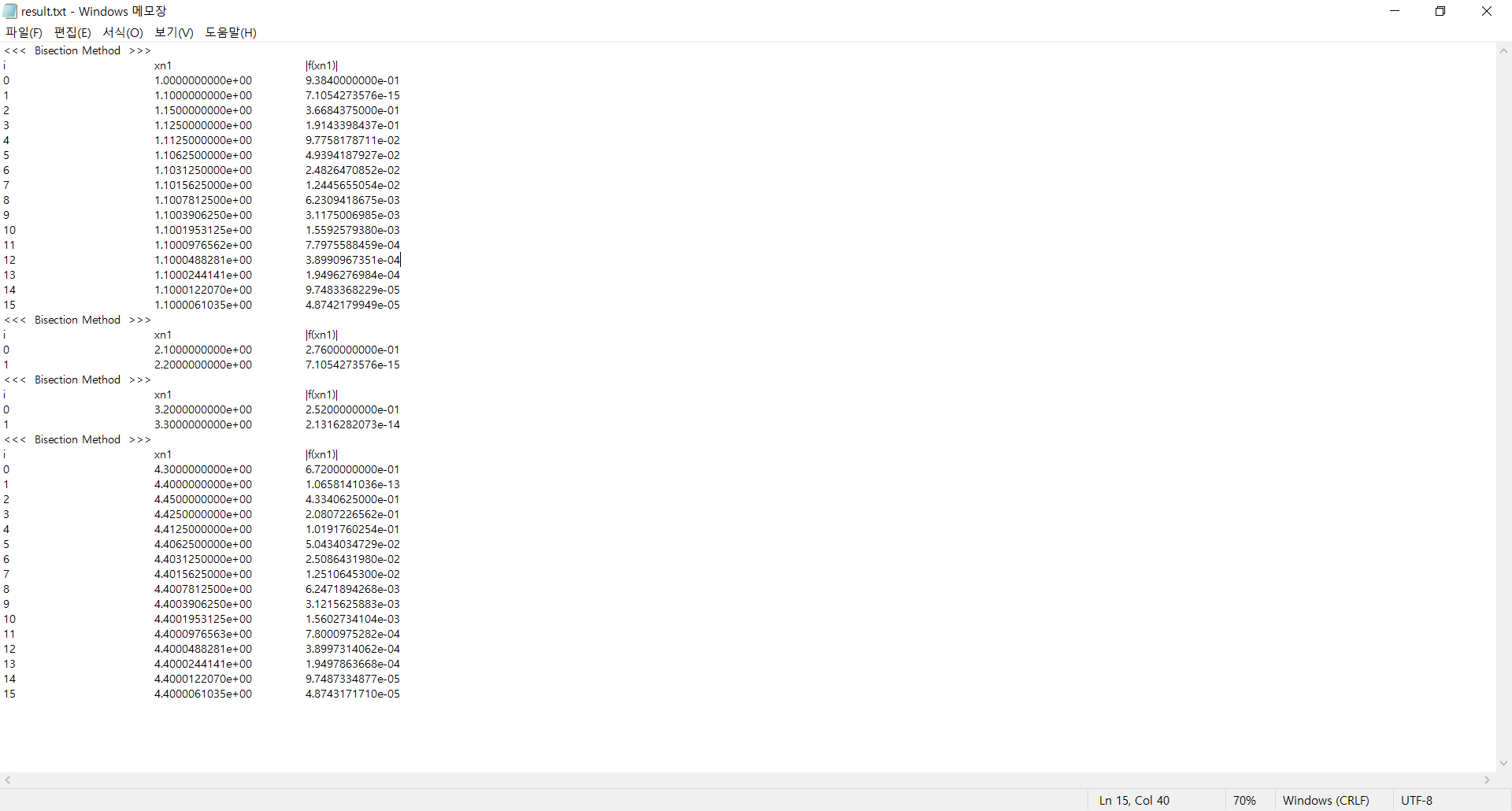
텍스트이(가) 표시된 사진

자동 생성된 설명

(f3(x)를 사용)

텍스트이(가) 표시된 사진

자동 생성된 설명



어느 정도 올바르게 근에 수렴하는 것을 확인할 수 있다.

(2) Bisection 방법의 수렴 속도는 선형적인, 즉 εn+1 ≈ 1 2 εn 형태를 보인다. 위 의 세 함수에 대한 방정식을 대상으로 Newtown-Raphson 방법, Secant 방법, 그리고 Bisection 방법을 적용해보고, 과˙ 연˙ 각˙ 방˙ 법˙ 이˙ 이˙ 론˙ 적˙ 인˙ 수˙ 렴˙ 속˙ 도˙ 를˙ 보˙ 이˙ 는˙ 지˙ 를 분석하고 그 내용을 보고서에 기술하라.

(1번 함수)

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

(2번 함수)

텍스트이(가) 표시된 사진

자동 생성된 설명

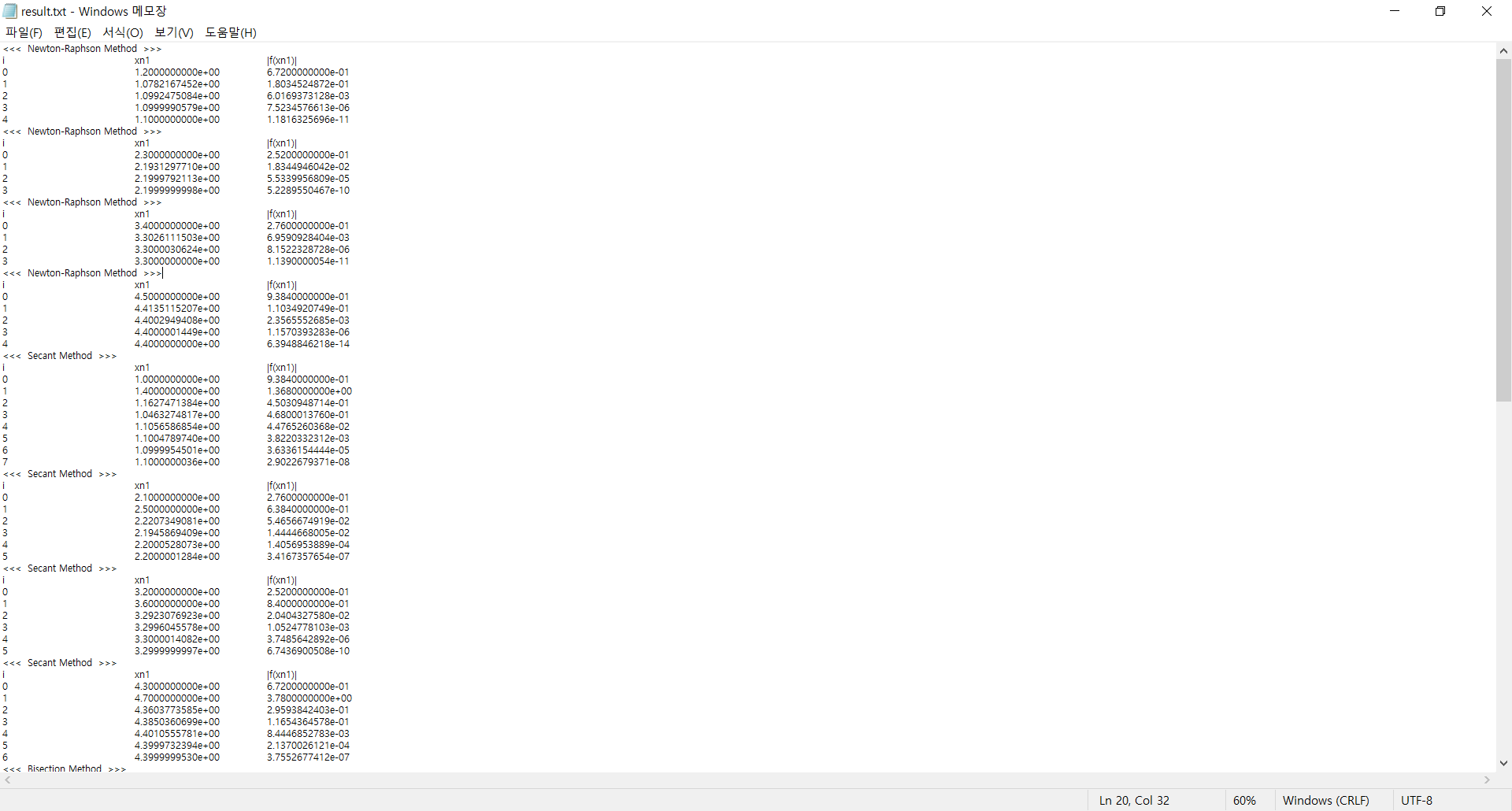
테이블이(가) 표시된 사진

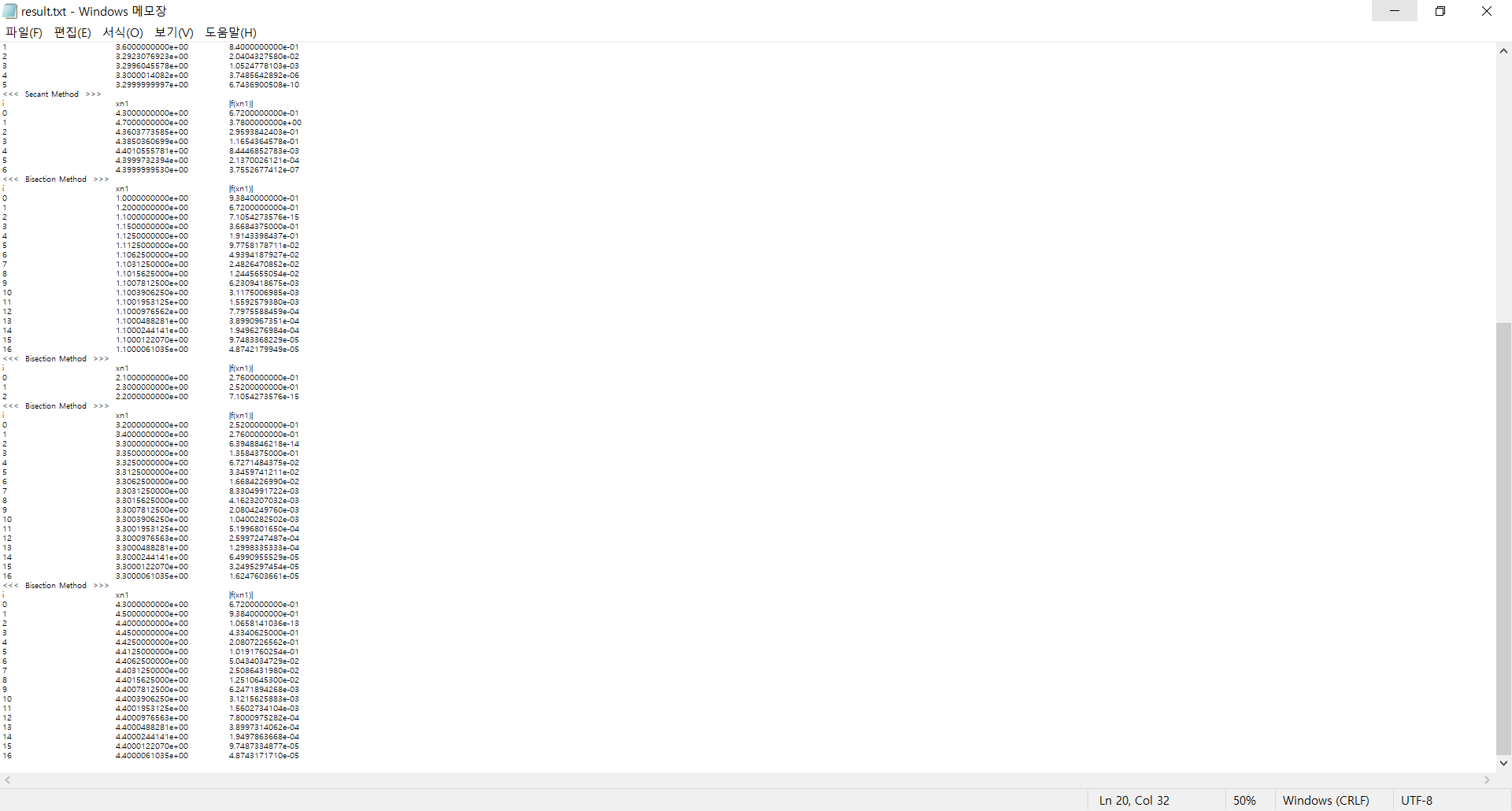
자동 생성된 설명

(3번 함수)

텍스트이(가) 표시된 사진

자동 생성된 설명





3개의 방식 모두 어느 정도 적절한 근을 구하는 것을 확인할 수 있고, 수렴 속도에 따라 실제 결과 또한 뉴튼-랩슨 방식보다 시컨트 방식이, 시컨트 방식보다 바이섹션 방식에서 계산 반복 횟수가 더 많이 나타나는 것을 확인할 수 있다.

-과제2

(결과)

텍스트이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

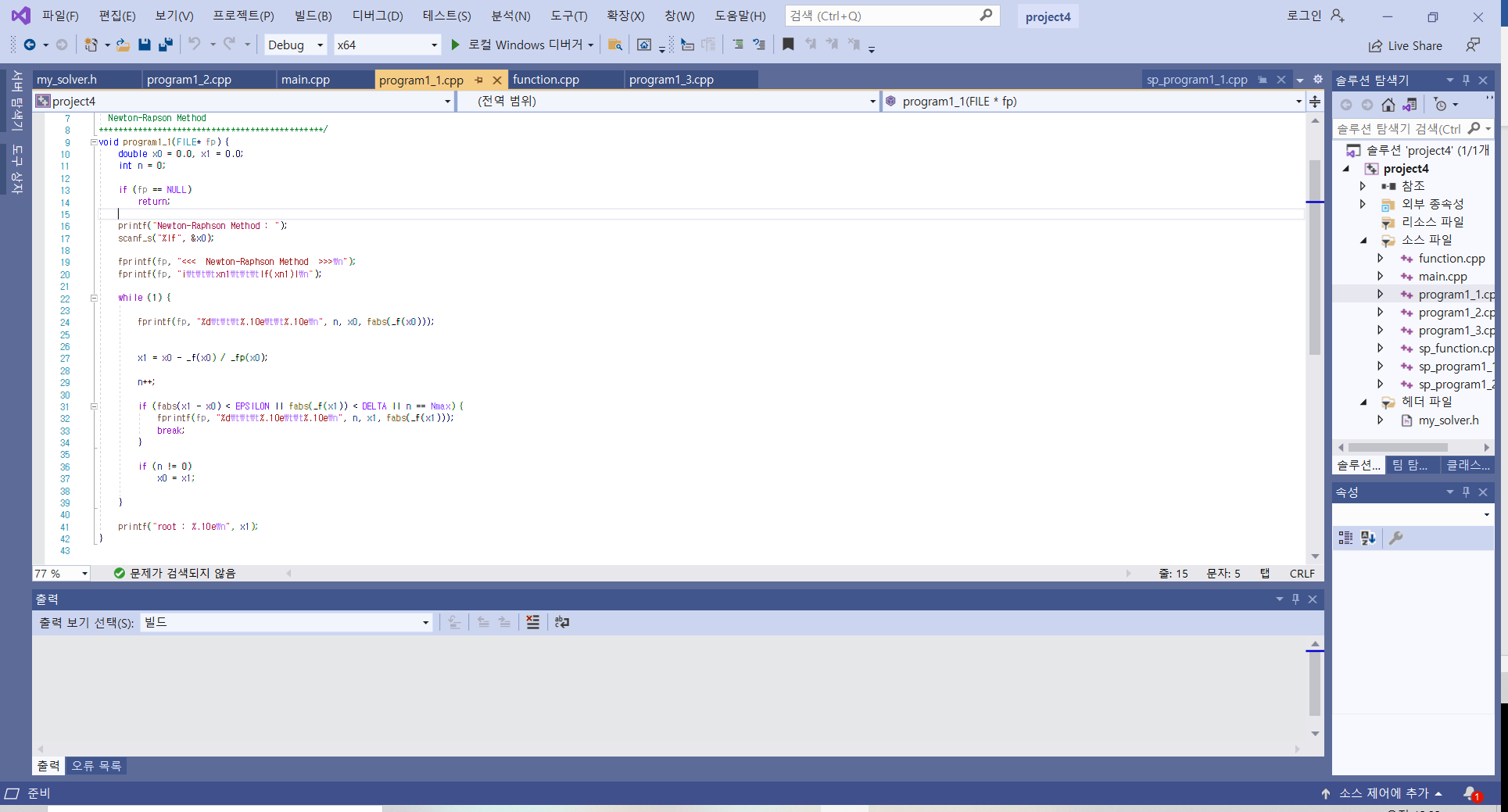
자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

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

뉴턴-랩슨 방법을 이용하여 근을 구하였다.



while(1)로 무한 반복문을 돌면서 x1, x0의 차가 엡실론 보다 작거나 현재 구한 함수 값이 델타보다 작거나 반복 횟수가 Nmax랑 같아지는 경우 반복문을 종료하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

계산을 반복할 때 마다 값이 작아지는 것을 확인할 수 있으며 이 경우에선 입력 값으로 33을 줬을 때 50번 끝까지 반복을 하고 종료가 되는 것을 확인할 수 있다.

텍스트, 스크린샷, 노트북, 실내이(가) 표시된 사진

자동 생성된 설명

과제2에서 쓰이는 수식은 다음과 같이 정의하였다.