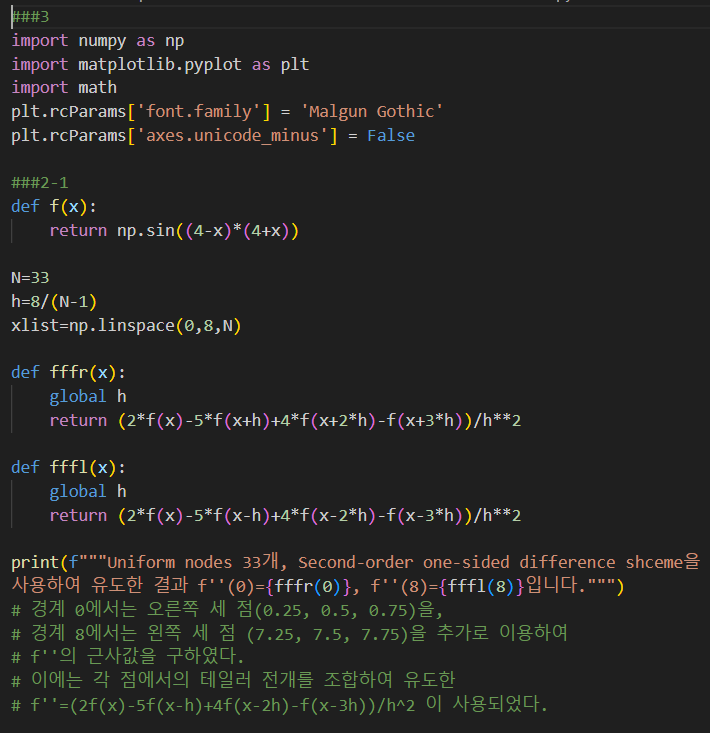
2주차 과제1 #3

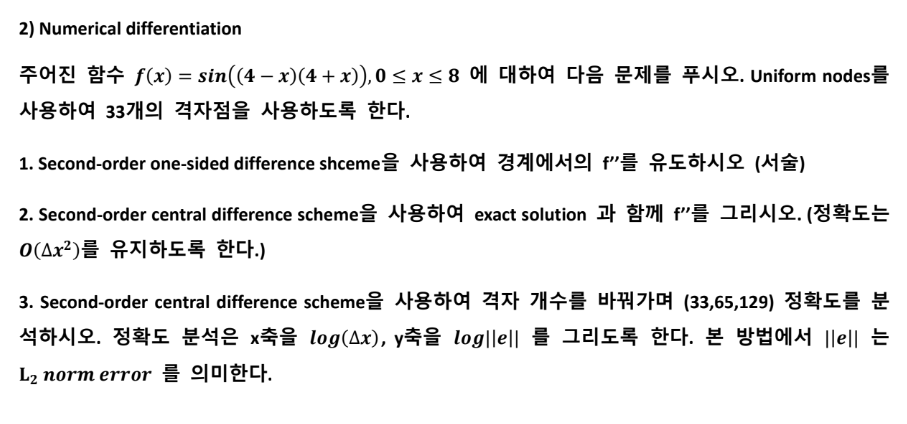
2022145079 임혜린

본 과제는 Python, VS Code를 사용하였음을 밝힙니다.

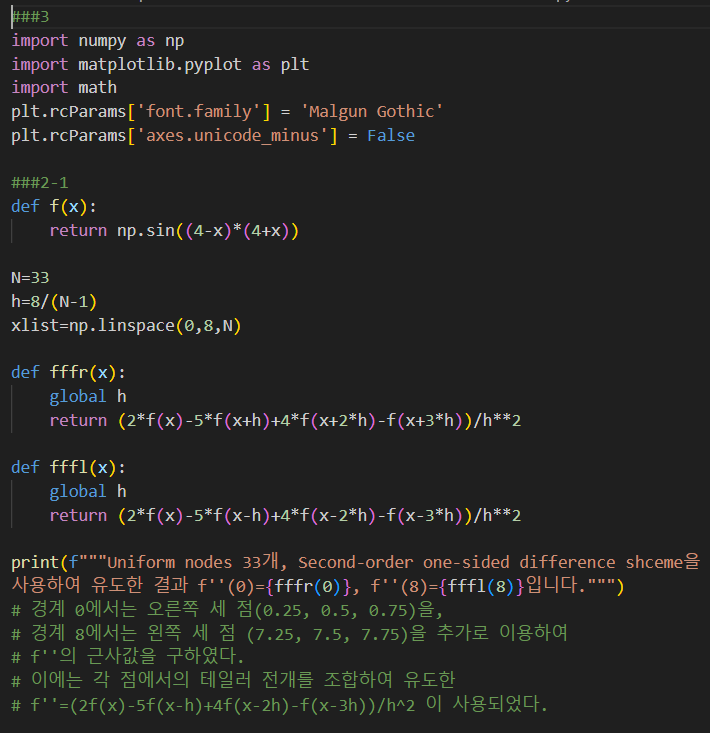
문제를 푸는데 있어 공통적으로 아래의 환경을 사용하였다.



2-1



코드



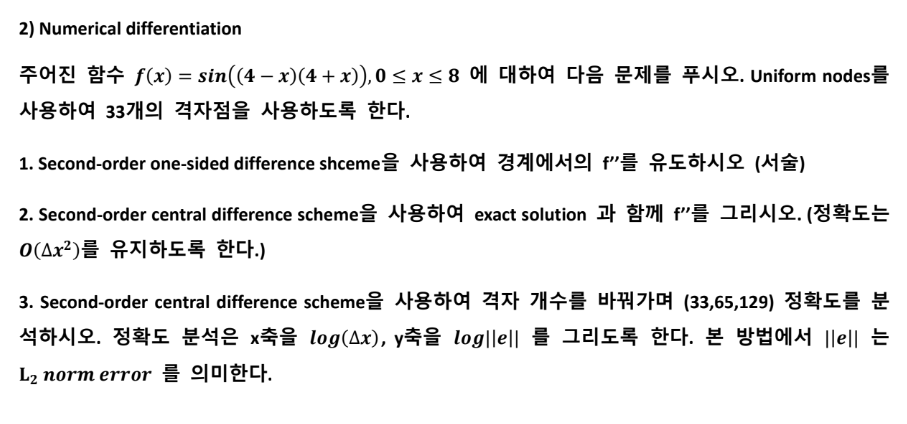
경계 0에서는 오른쪽의 세 점(0.25, 0.5, 0.75)을 이용한 3step 2차 후진 차분, 경계 8에서는 왼쪽 세 점 (7.25, 7.5, 7.75)을 이용한 3step 2차 전진 차분을 이용하여 f’’의 근사값을 구하였다.

이에는 각 점에서의 테일러 전개를 조합하여 유도한 f’’=(2f(x)-5f(x-h)+4f(x-2h)-f(x-3h))/h^2가 사용되었다.

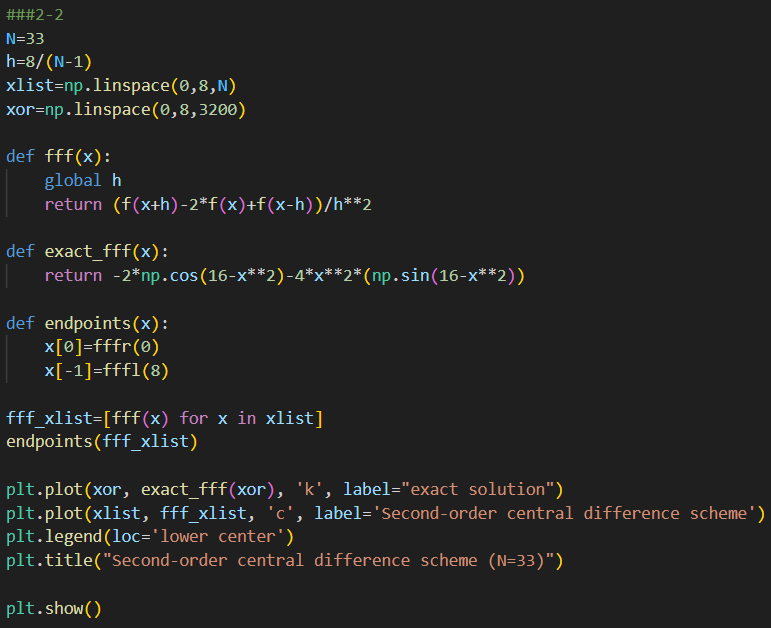
결과



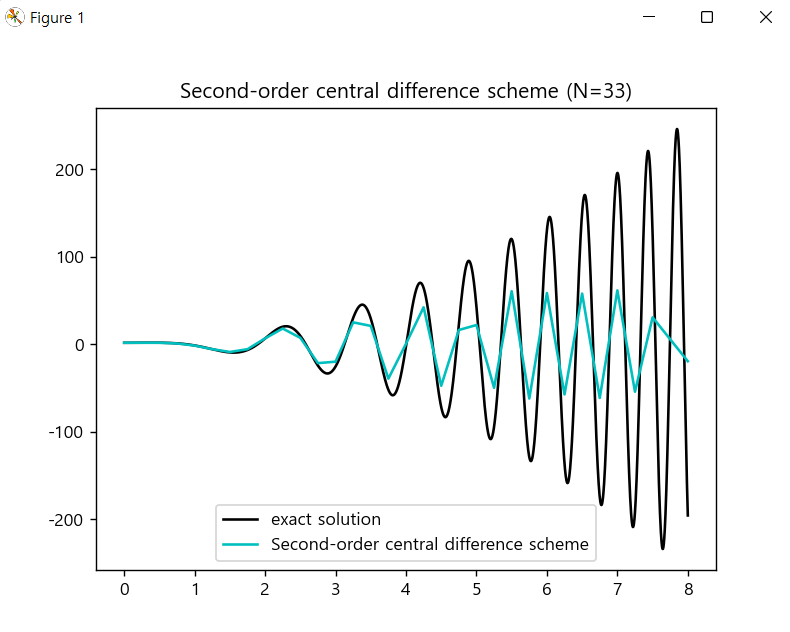
2-2



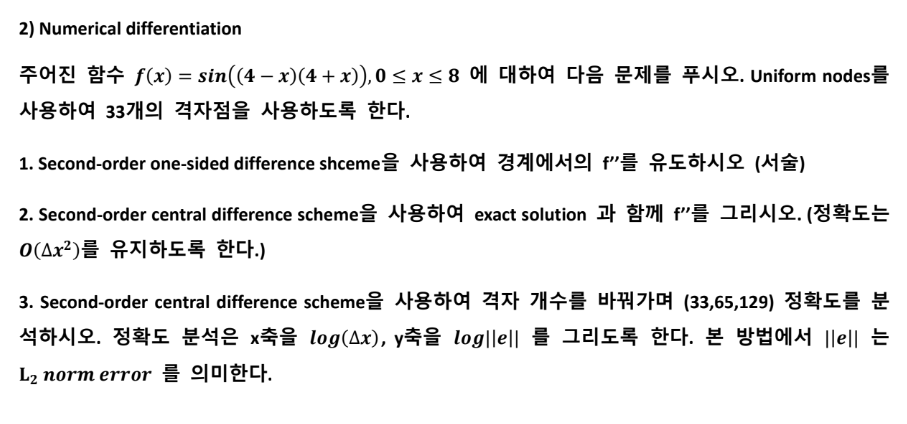
코드

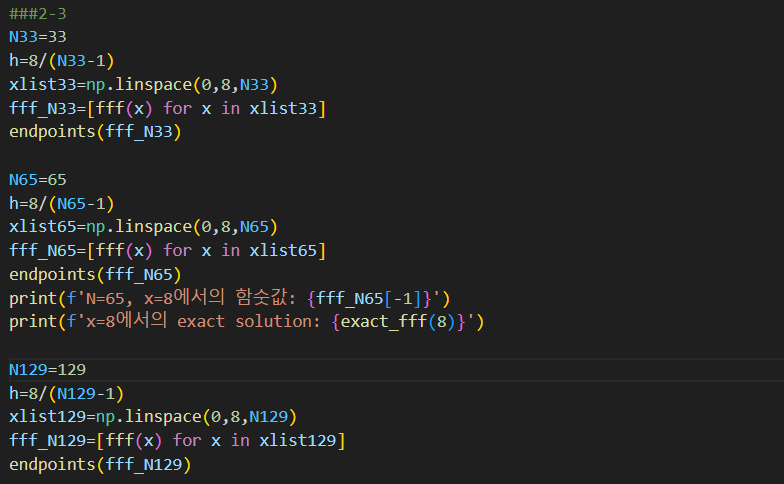


결과

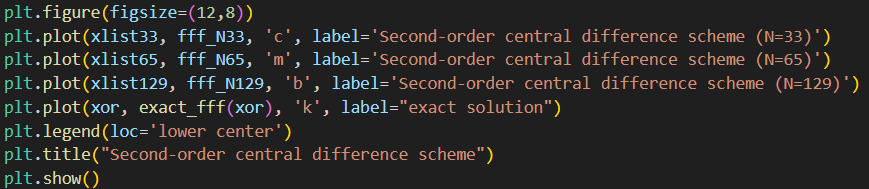


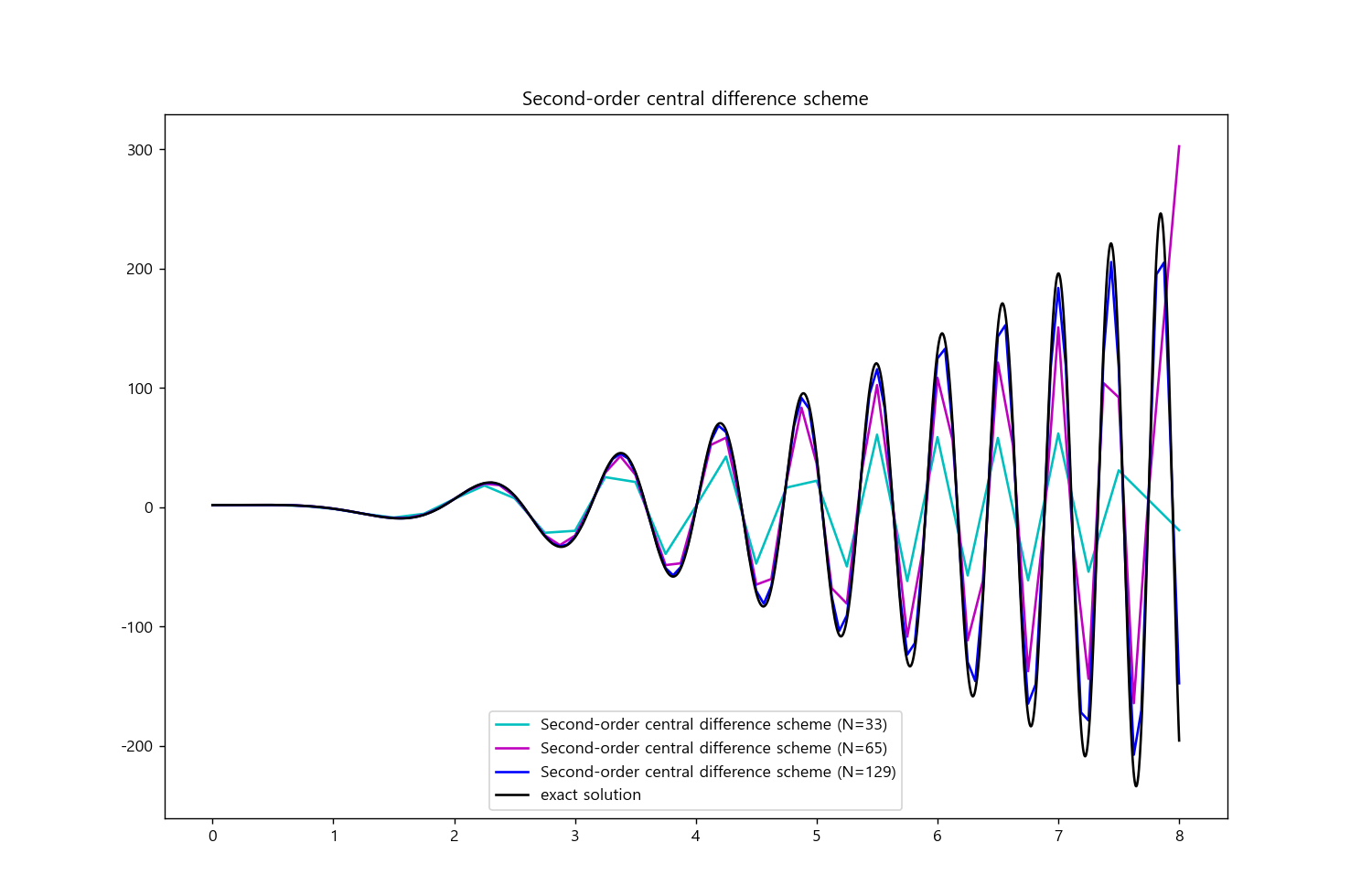
2-3

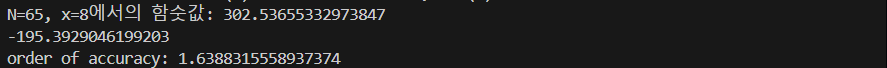


코드

정확도 분석에 앞서 N에 따른 세 f’’ 그래프를 정의하고 한 번에 그려보았다. 각 경계에서는 중앙 차분을 사용할 수 없기 때문에 2-1처럼 후진 차분과 전진 차분을 사용하였다.

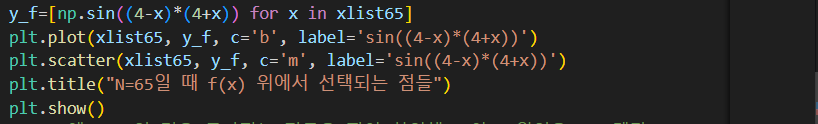


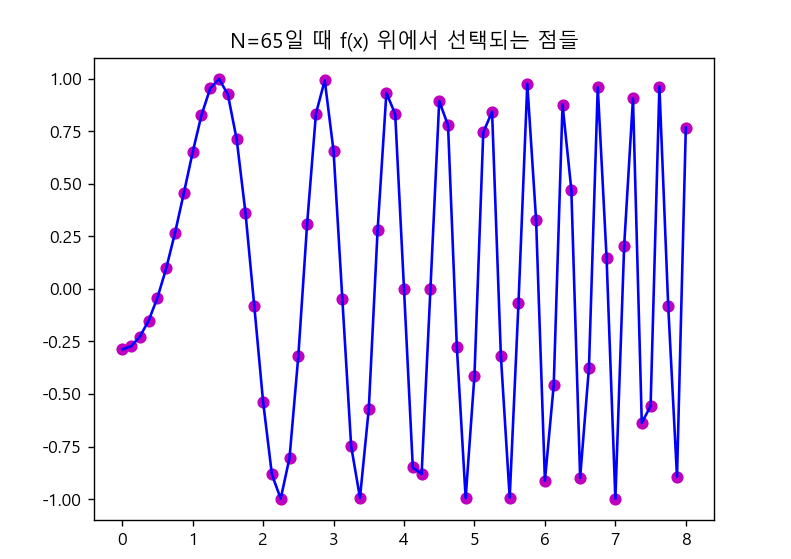




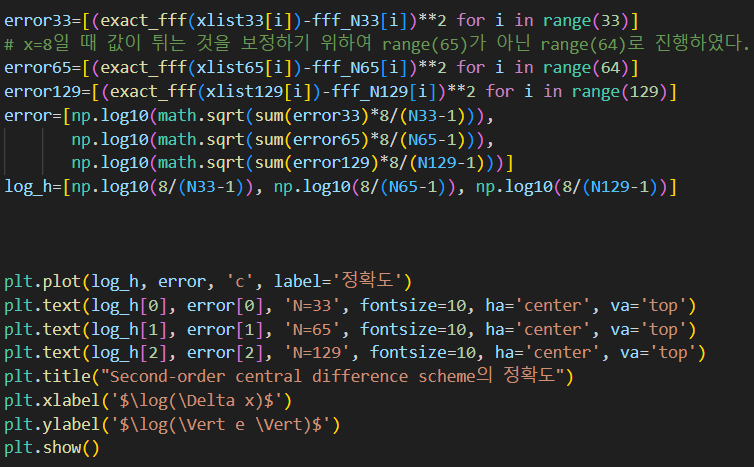
그래프를 통해 N=65인 경우 x=8인 지점에서 함숫값이 302로 튀는 것을 확인하였다. Exact solution의 해는 -195로 매우 큰 차이를 보인다. 이 오차가 아래의 오차 검증에 큰 영향을 미쳐 오차 계산 시 이 점을 제외하고 진행하였다.

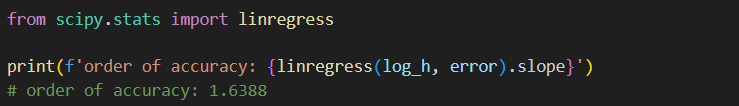
함숫값이 튀는 원인을 확인하기 위해 f(x)에 N=65인 경우 골라지는 점들을 찍어 확인해 보았다



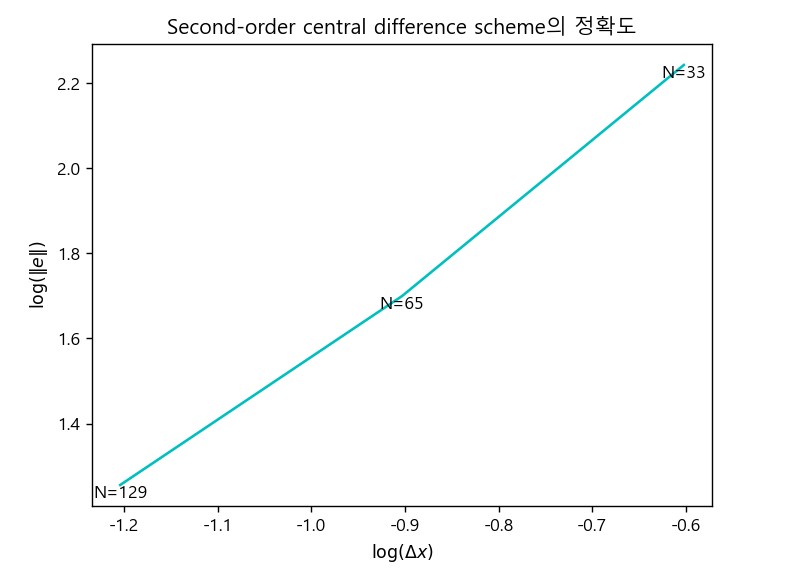


사용한 전진 차분의 식은 (2f(x)-5f(x-h)+4f(x-2h)-f(x-3h))/h^2이다. 가장 가중치가 큰 f(7.75)의 값이 극단적인 것이 주요 원인으로 보이나 직접 값을 확인해 보기 전까지는 이를 예상하기 어려울 것으로 보인다.





결과



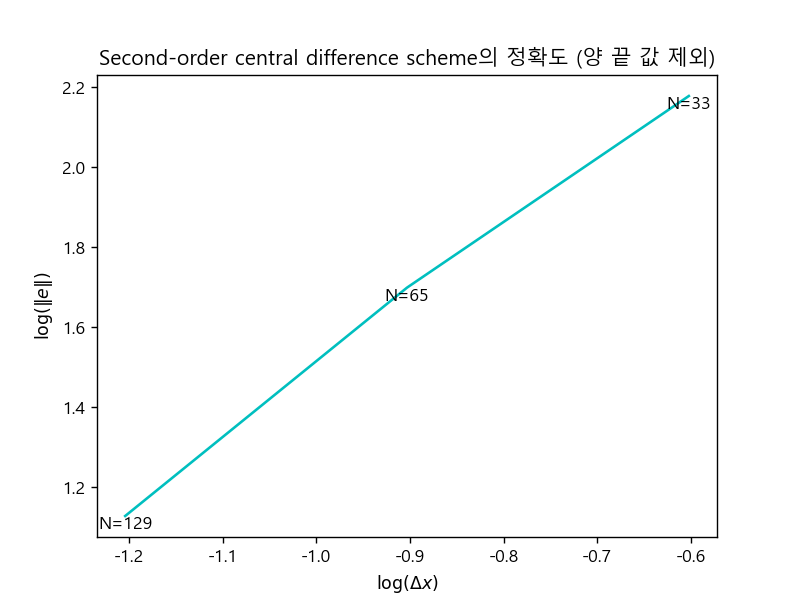
Order of accuracy: 1.6388

함수의 기울기가 유의미하지 않아 전진 차분, 후진 차분으로 만든 부분을 빼고 다시 error을 살펴보았다.





결과



Order of accuracy: 1.6388

함수의 기울기가 2와 유사함을 확인하였다. 이를 통해 오차가 Δx^2에 비례하여 감소함을 알 수 있었다.