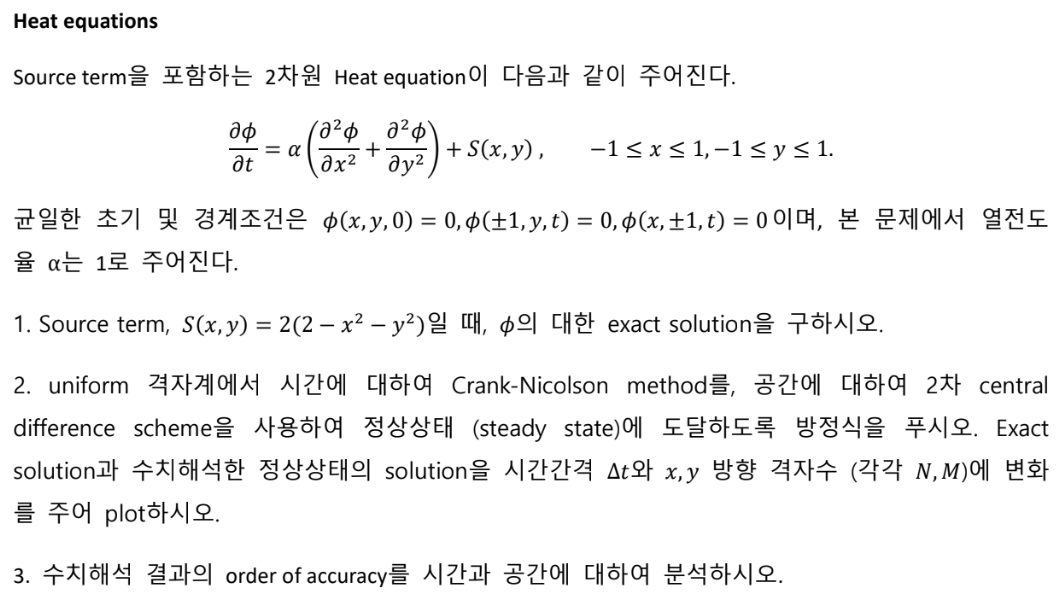
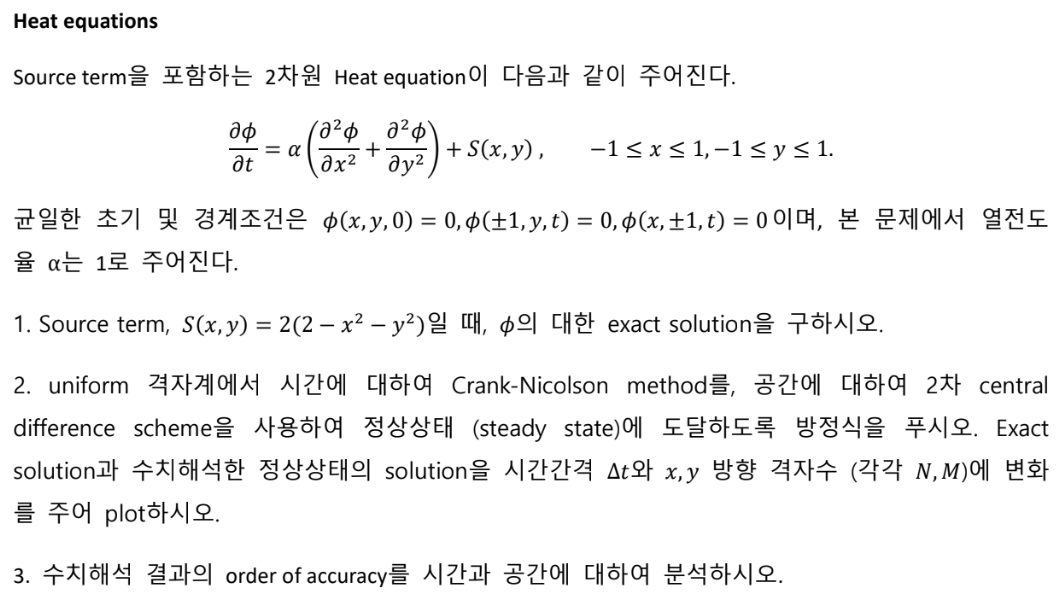
4주차 과제 #6

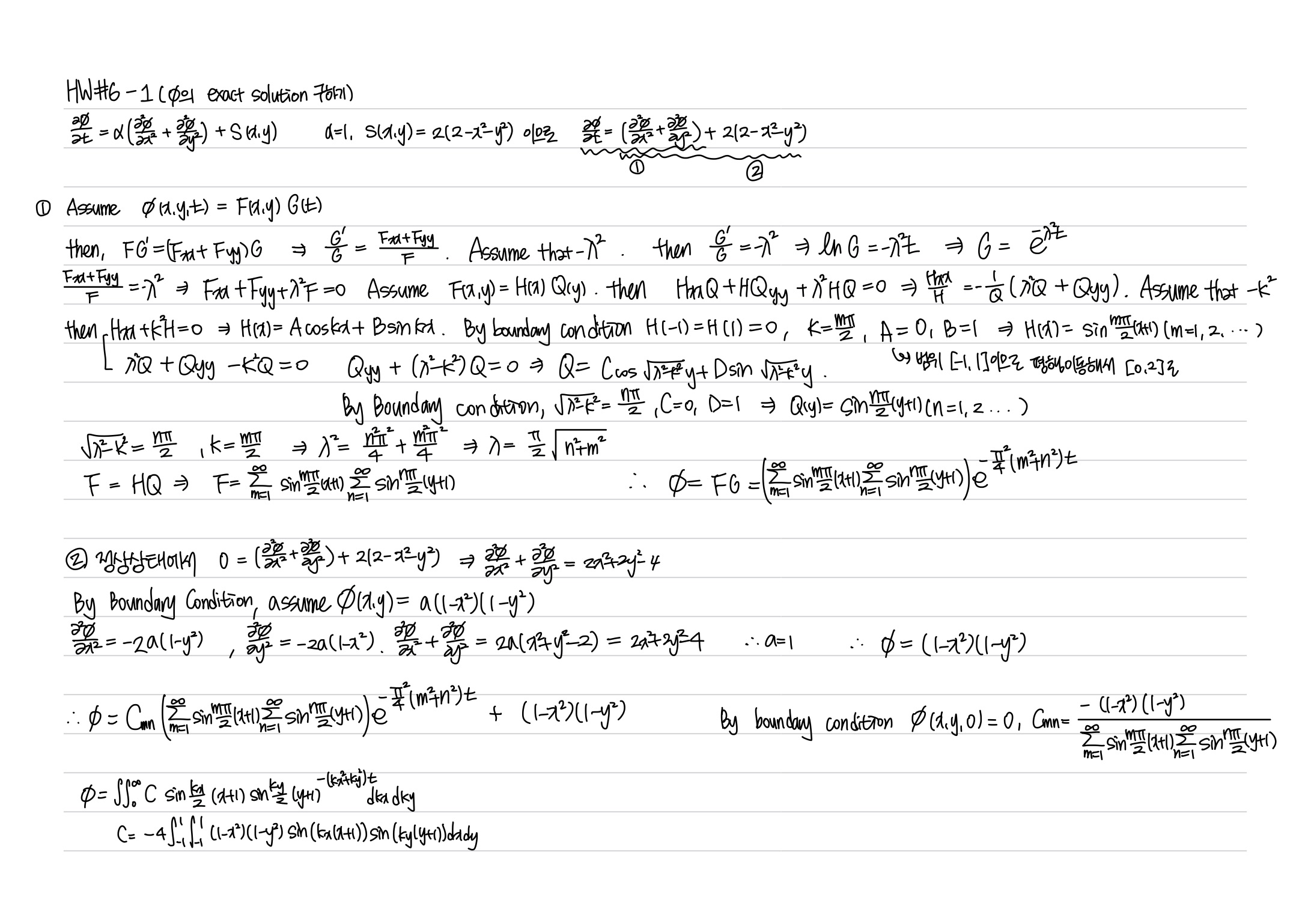
2022145079 임혜린

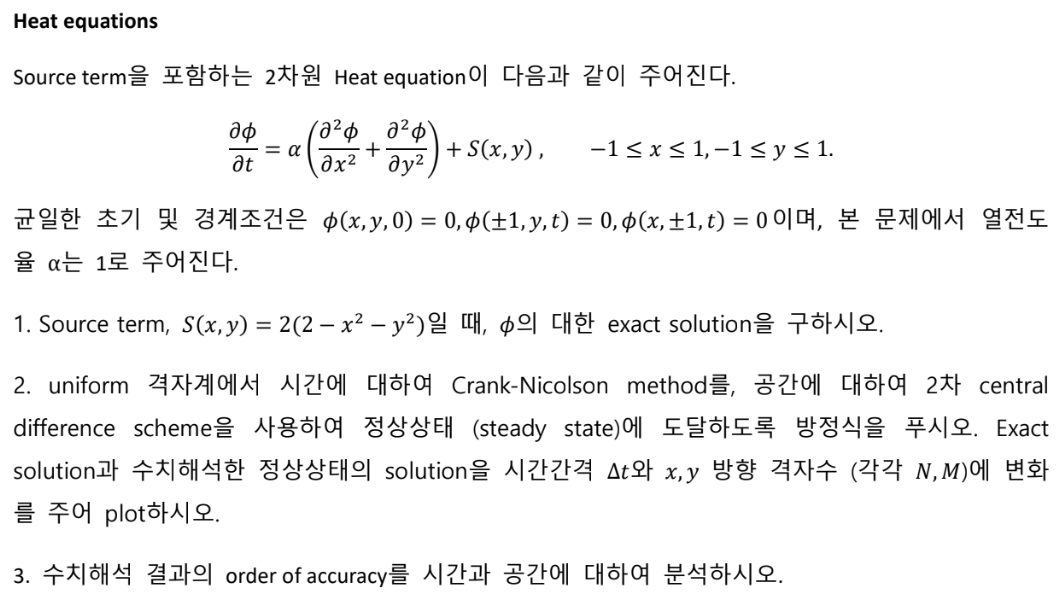
본 과제는 Python, VS Code를 사용하였음을 밝힙니다.

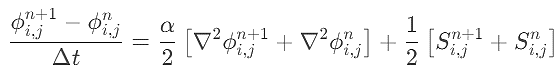
공통 조건 (Heat equations)



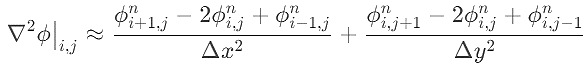






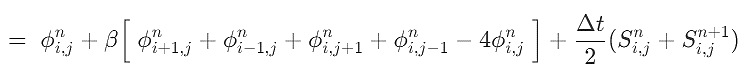
Crank-Nicolson scheme은 시간 미분을 전 시간 단계와 다음 시간 단계의 중앙 평균으로 근사한다. Heat equation의 지배방정식에 이를 적용하면 아래와 같다.

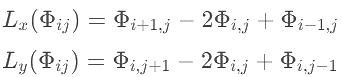
n은 time step, i,j는 spatial index이며 이때



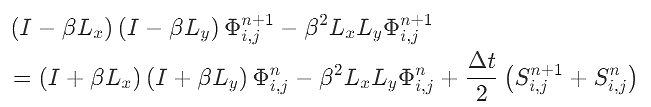
이다. Δx=Δy이므로 Δx=Δy=h로, β=αΔt/2h2으로 놓고 좌변을 unknown, 우변을 known으로 정리하면 아래와 같다.





이때 operator Lx, Ly를 적용하면, 식을 아래와 같이 변형할 수 있다.





β2항은 n에 따라 크게 변화하지 않으므로 양 변의 β2항이 상쇄 가능하다 가정하면 결과적으로 아래와 같은 식을 얻을 수 있다.

또한 상쇄한 β2항이 Δt2에 비례하므로 dt에 대한 L2norm error가 2차로 나올 것을 예상할 수 있다.

마지막으로 얻었던 식의 좌변의 I-βL2를 제외한 부분을 ψ, 우변을 R이라고 하면 식은 (I-βLx)ψ=R이 된다. 이 형태에서 한 번 풀어서 ψ를 알아낸 후, (I-βLy)ф=ψ를 이용하여 ф를 알아내는 방식으로 코드를 작성하였다. 즉,

이와 같이 변형할 수 있으므로 이것을 np.linalg.solve로 한 번 푼 후, 이렇게 찾은 ψ를 이용하여

이를 진행하여 ф를 구한 후, 이를 이용하여 다시 다음 t에서의 ф를 구해 나가는 방식이다.

텍스트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

텍스트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

h=x, y 방향 격자수, h=Δx=Δy=2/n, I\_\_bx=2차원 넘파이 배열로 나타낸 I-βLx, I\_bx=2차원 넘파이 배열로 나타낸 I+βLx, pi\_list=각 t에 대한 ф를 담은 2차원 넘파이 배열을 담은 리스트이다.

n=41, t=0.1로 설정하였다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

이중 초기 t에 대한 등고선을 그려보았다.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

내부 온도가 진동하고 이 진동이 시간이 지날수록 느려진다는 것을 확인하였다. 다음으로 t=10.0일 때의 등고선과 exact solution의 등고선을 비교함으로써 두 등고선이 거의 일치함을 확인하였다.

|  |  |
| --- | --- |
|  |  |

다음으로 Δt와 h를 바꾸어가며 등고선을 비교하였다. Δt=0.1, 0.01, h=0.2, 0.1, 0.05이며 t가 0.5일 때와 10.0일 때, 총 12개의 등고선을 비교하였다.

그 결과 h가 감소, 즉 n이 증가할수록 등고선이 더욱 스무스하게 그려짐을 확인하였으며, Δt에 대해서는 육안으로 뚜렷한 차이를 찾지 못하였으나 Δt가 작을수록 작은 t에 대해 더 스무스한 결과를 얻게 되는 것으로 예상된다.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

텍스트, 스크린샷, 폰트, 대수학이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* h의 변화

Δt=0.1, t=10.0(steady state)를 고정해 놓은 뒤, ф와 (range(99)한 후 pi\_list[-1]) steady state에서의 exact solution (1-x2)(1-y2)의 L2norm을 구하여 error을 구하였다. n=4, 6, 8, 10으로 격자 수의 변화를 주며 error을 구하며 구한 각 격자 수에서의 error을 error\_list에 담아 error\_list를 완성하였다.

log10(h)와 log10(error\_list)의 그래프를 그려 그래프의 order of accuracy를 구하였다. 또한 기울기가 2인 점선 그래프를 그려 한 눈에 비교할 수 있도록 하였다.

텍스트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

텍스트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

텍스트, 라인, 도표, 그래프이(가) 표시된 사진

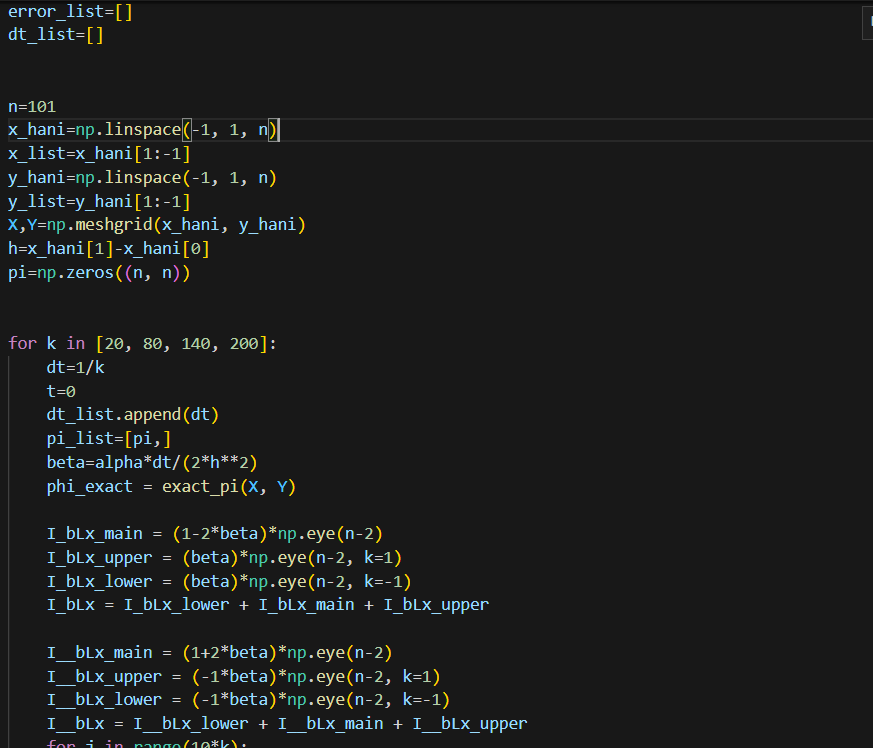
AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

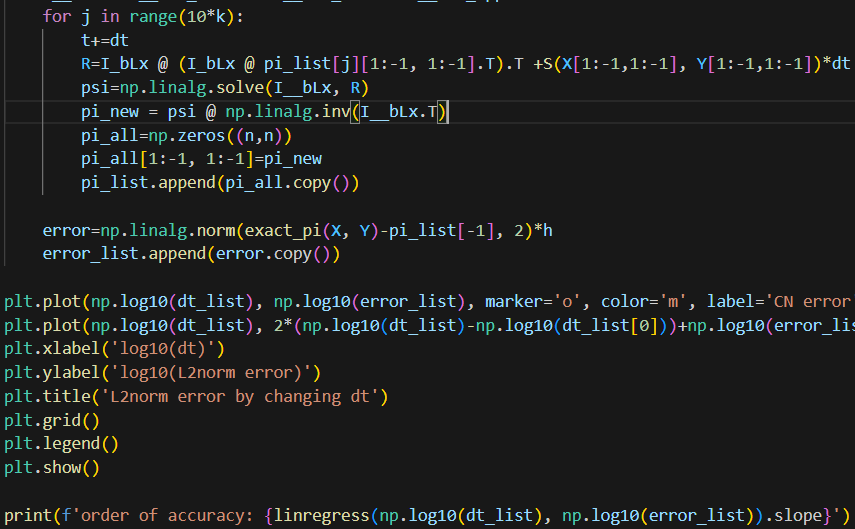


그 결과 order of accuracy가 2에 가깝게 나오는 것을 확인하였다.

* Δt의 변화

비슷한 방법으로 Δt를 변화시키며 진행하였다. 사용된 Δt는 1/20, 1/80, 1/140, 1/200이다.





텍스트, 라인, 그래프, 도표이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.



이 또한 기울기가 2에 가깝게 나오는 것을 확인할 수 있었다.

다만, 이 선형성은 다양한 dt와 h에 대해서 항상 성립한다고 보기 어렵다. error 값이 균일하게 나오지 못하는 것에는 초기 t부터 exact solution을 steady state로 잡은 것이 작용했을 것으로 보인다. 따라서 문제 1에서의 답을 exact solution으로 적용한다면 더욱 정확한 error 그래프를 얻을 수 있을 것으로 보인다.