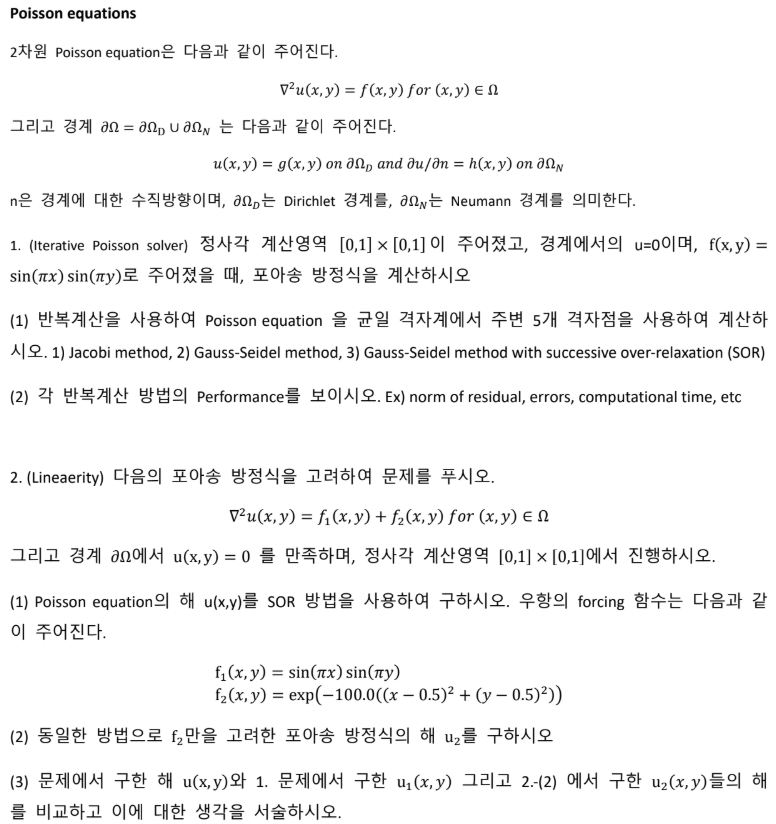
5주차 과제 #7

2022145079 임혜린

본 과제는 Python, VS Code를 사용하였음을 밝힙니다.

공통 조건 (Poisson equations)



푸아송 방정식은 라플라스 방정식을 일반화한 2차 편미분 방정식이다. 유체역학의 영역에서는 압력, 속도 퍼텐셜, 스트림 함수 등에 자주 쓰인다. 압력의 경우에는, 비압축성 유동에서 Navier-Stokes 방정식과 연속방정식을 이용하여 압력에 대한 직접적인 방정식을 만들면 푸아송 방정식 형태가 유도된다.



속도 퍼텐셜의 경우에는, 비회전성 유동에서 source나 sink가 존재한다면 라플라스 방정식이 아닌 푸아송 방정식의 형태로 나타난다.

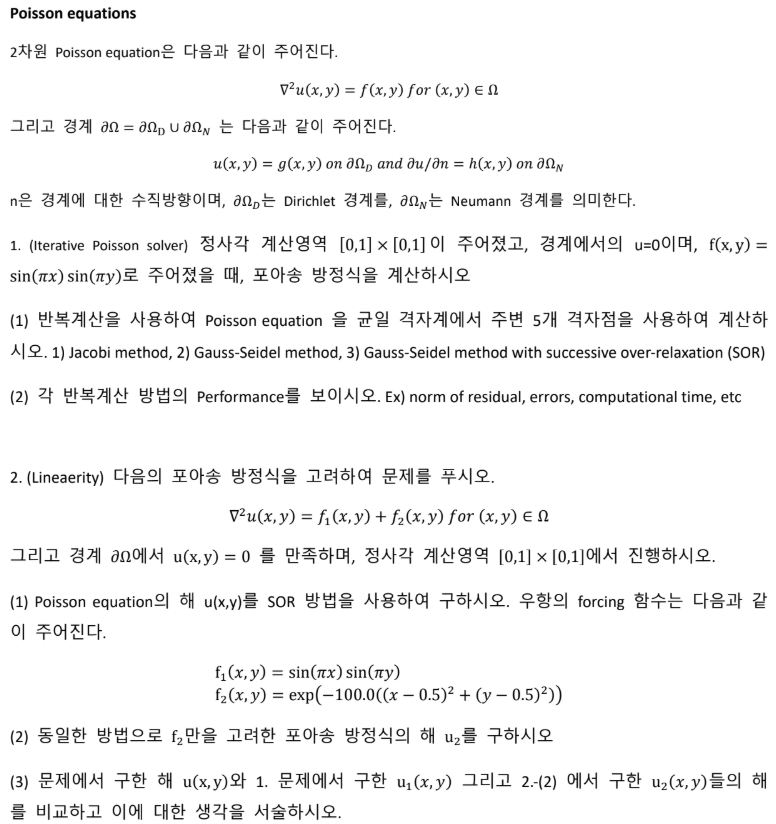


스트림 함수와 와류의 관계 또한 푸아송 방정식이 성립한다.

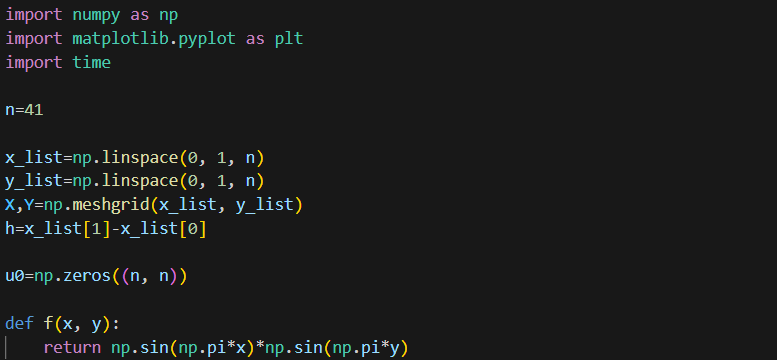


이렇듯 유체역학에 빈번히 적용되는 푸아송 방정식을 CFD에 적용하는 연습을 할 예정이다.

1-1

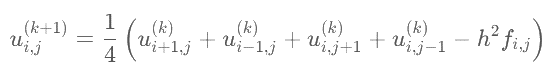


3가지 방법 모두 기본적으로 아래와 같은 환경에서 코딩하였다.



1. Jacobi method

주변 변수를 전부 이전 회차의 값을 이용해 미지수를 계산하는 방식이다. 식은 아래와 같다.



0으로 채워진 (n, n) 2차원 벡터 u\_new를 만든 후, 이전 값들을 모은 u\_list\_J의 가장 최근 값, 즉 직전 값 5개를 이용하여 u\_new의 값을 채워나간다.

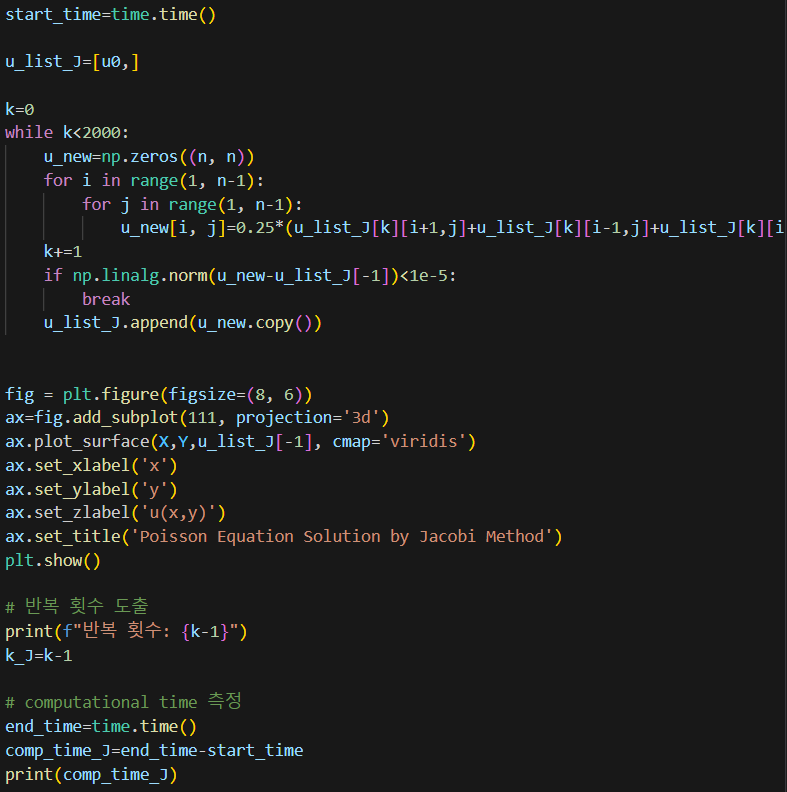
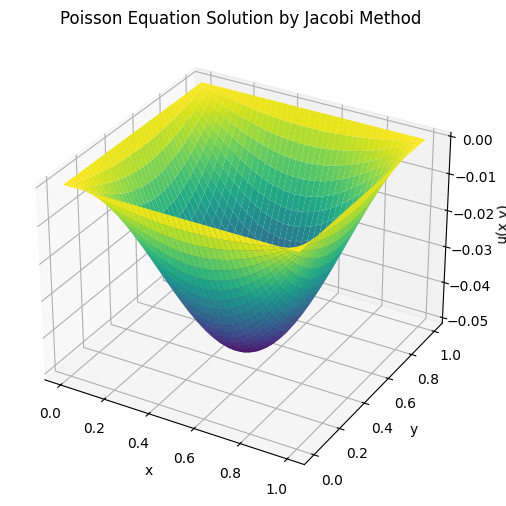
u\_new[i, j]=0.25\*(u\_list\_J[k][i+1,j]+u\_list\_J[k][i-1,j]+u\_list\_J[k][i,j+1]+u\_list\_J[k][i,j-1]-(h\*\*2)\*f(x\_list[i],y\_list[j]))

모두 채워진 후 이 u\_new를 u\_list\_J에 복사해 추가함으로써 모든 반복 횟수 k에서 u\_list\_J[k]가 직전 값이 되도록 한다. 즉, u\_new의 전체가 채워지기 전에는 늘 직전 값만이 이용되는 것이다.

반복은 직전 값과 새로운 값의 L2 값이 10-5보다 작아지면 멈춘다.

1-2번 문제에 사용할 computational time을 측정하기 위해 코드 위 아래에 start\_time과 end\_time을 두어 computational time을 comp\_time\_J에 저장하였다.

반복횟수 k-1은 k\_J로 저장된다.

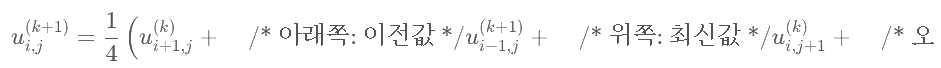


결과



1. Gauss-Seidel method

한 회차에서는 새롭게 갱신된 값이 있어도 모두 이전 회차의 값을 사용하는 Jacobi method와는 달리, Gauss-Seidel method는 새롭게 갱신된 값을 즉시 다른 변수 계산에 적용한다. 따라서 Jacobi method보다 빠르게 수렴하는 경향을 보인다. 식은 아래와 같다.





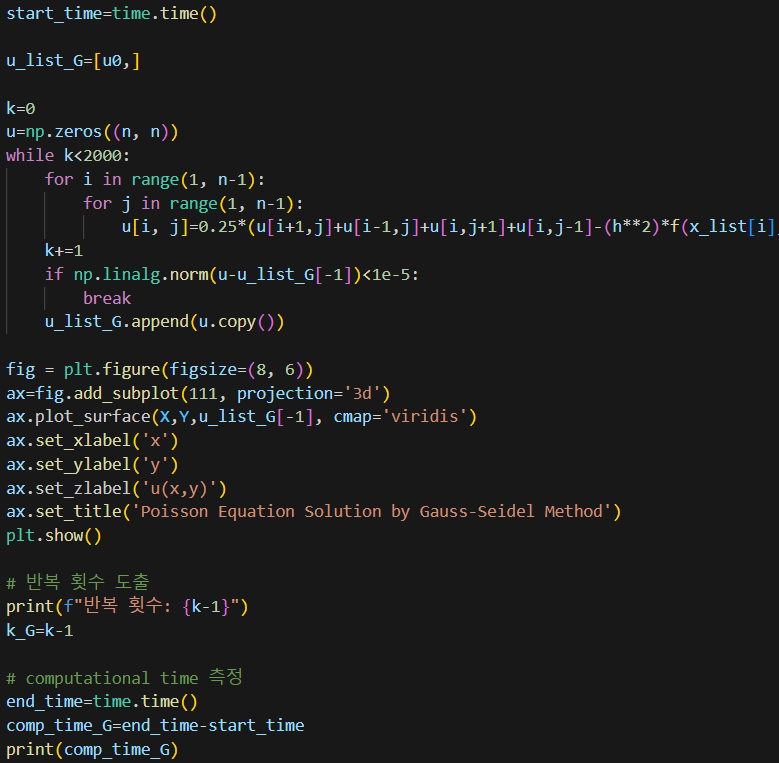
새로운 값이 바로 반영되어야 하기 때문에 좌변과 우변 모두 같은 2차원 벡터를 사용하였다. 새로운 값이 즉시 u에 덮어써지기 때문에 [i-1, j], [i, j-1]이 새로운 값으로 대응된다.

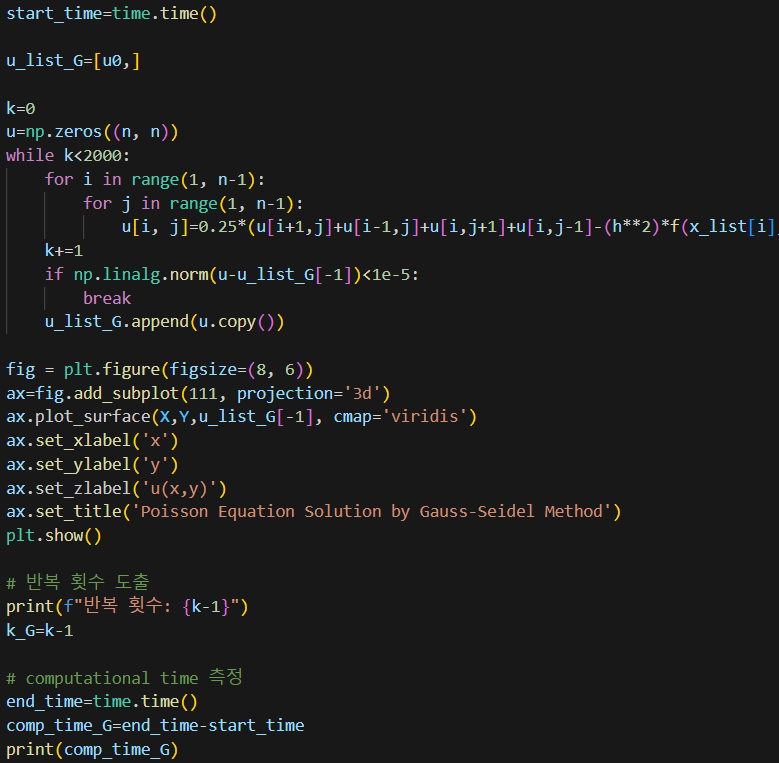
u[i, j]=0.25\*(u[i+1,j]+u[i-1,j]+u[i,j+1]+u[i,j-1]-(h\*\*2)\*f(x\_list[i],y\_list[j]))

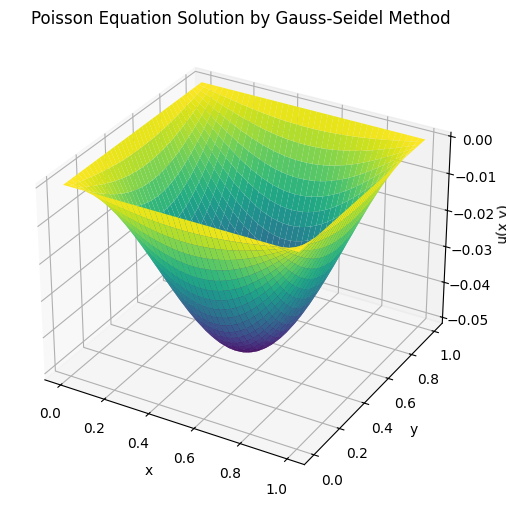
반복은 Jacobi와 마찬가지로 직전 값과 새로운 값의 L2 값이 10-5보다 작아지면 멈춘다.

1-2번 문제에 사용할 computational time을 측정하기 위해 코드 위 아래에 start\_time과 end\_time을 두어 computational time을 comp\_time\_G에 저장하였다.

반복횟수 k-1은 k\_G로 저장된다.



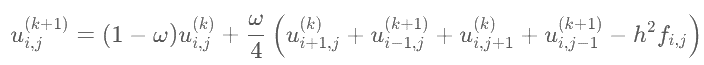


결과



1. Gauss-Seidel + SOR (Successive Over-Relaxation)

기본적으로 Gauss-Seidel method를 사용하되, 수렴 속도 증가를 위해 해의 새 값에 완화계수를 곱하여 적절히 보정하는 방식이다. 완화계수가 너무 크거나 작으면 발산하기 때문에 적절히 조절하는 것이 중요하다.



코드의 틀은 Gauss-Seidel과 같으나 완화계수 ω를 넣어 더욱 빠르게 수렴하도록 하였다.

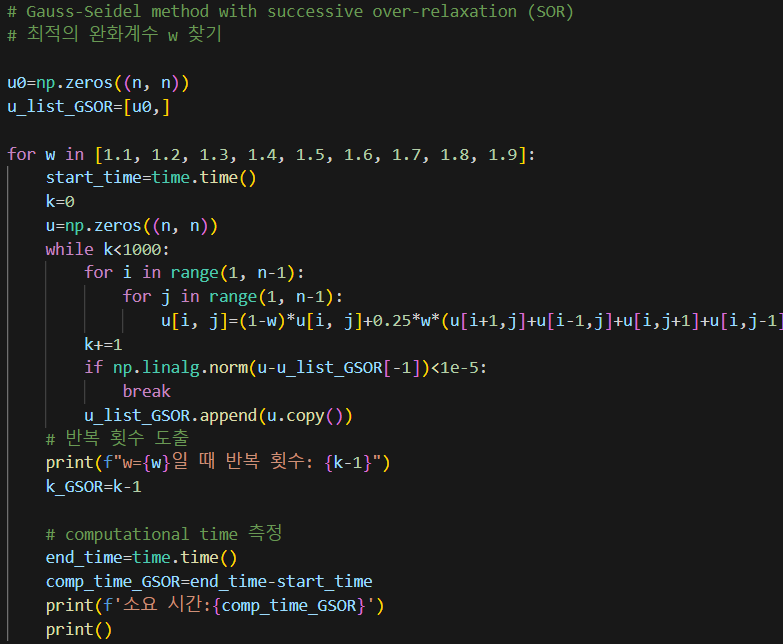
u[i, j]=(1-w)\*u[i, j]+0.25\*w\*(u[i+1,j]+u[i-1,j]+u[i,j+1]+u[i,j-1]-(h\*\*2)\*f(x\_list[i],y\_list[j]))

반복은 마찬가지로 직전 값과 새로운 값의 L2 값이 10-5보다 작아지면 멈춘다.

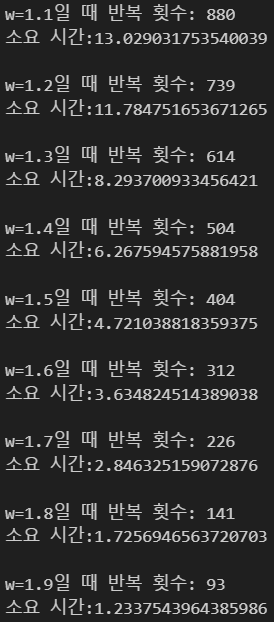
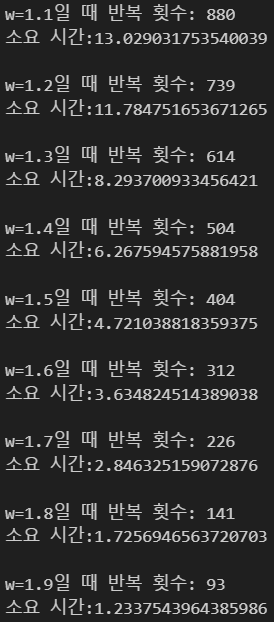
1-2번 문제에 사용할 computational time을 측정하기 위해 코드 위 아래에 start\_time과 end\_time을 두어 computational time을 comp\_time\_GSOR에 저장하였다.

반복횟수 k-1은 k\_GSOR로 저장된다.

수렴을 보장하기 위해서는 |1-ω|<1이어야 한다. 따라서 ω는 0<ω<2이어야 하고, 수렴 속도를 줄이기 위해서는 1<ω<2가 요구된다. 따라서 이 범위에서의 가장 최적의 ω를 찾기 위하여 1.1부터 1.9까지 0.1 간격으로 ω를 적용하였을 때의 반복 횟수와 소요 시간을 살펴보았다.

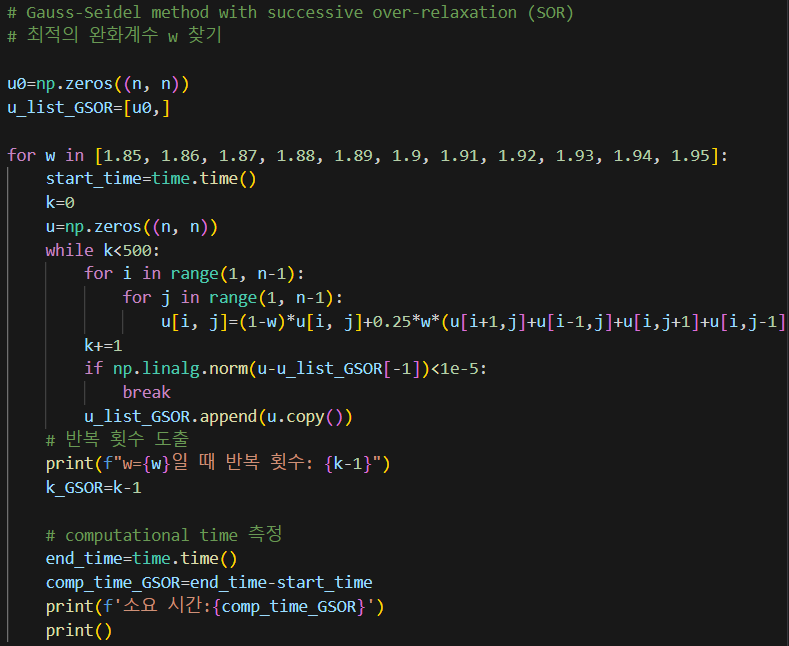


결과

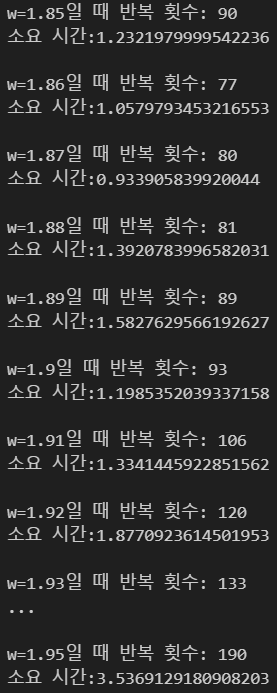
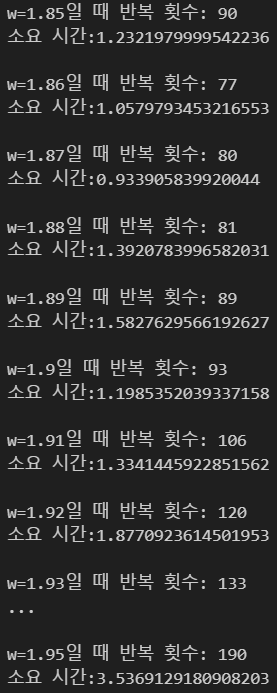


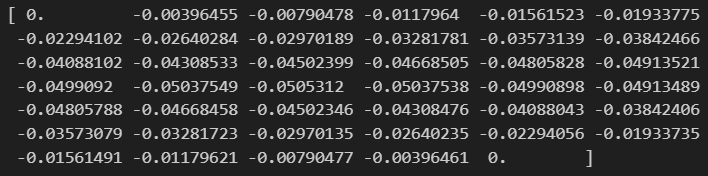
1.9일 때 가장 반복 횟수와 소요 시간이 작았다.

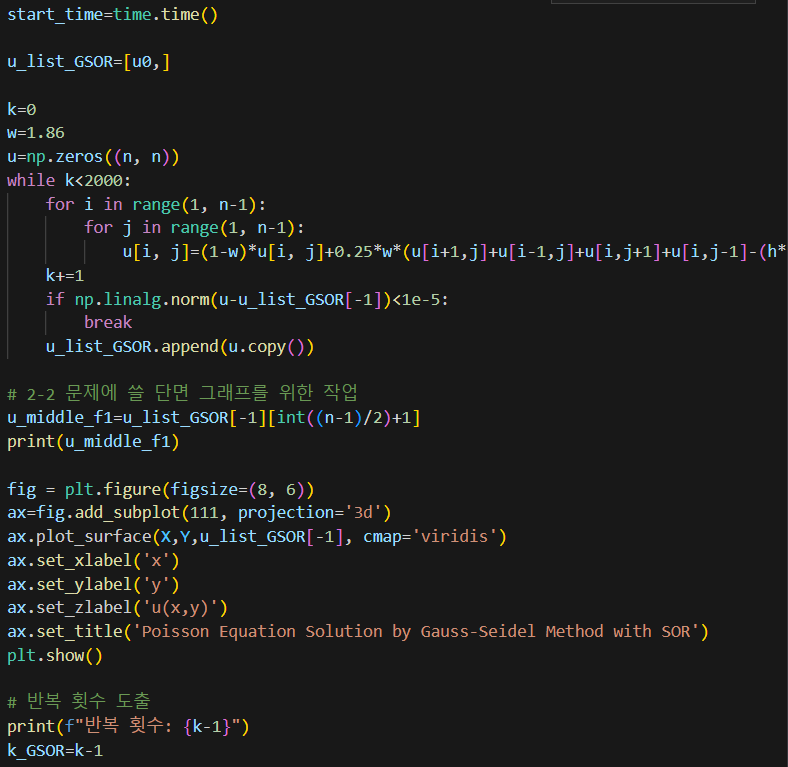
조금 더 자세히 알아보기 위해 1.85부터 1.95까지 0.1 간격으로 다시 측정하였다.

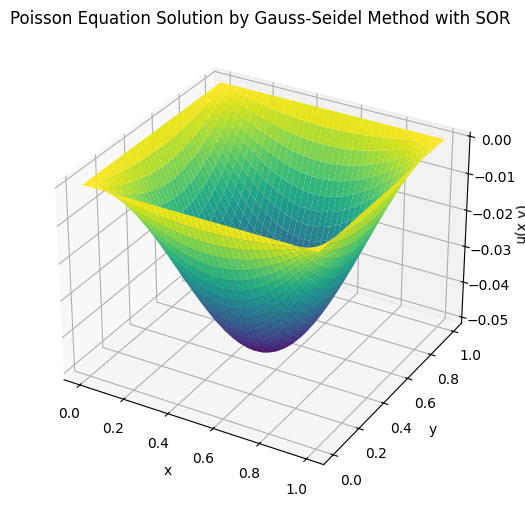


결과

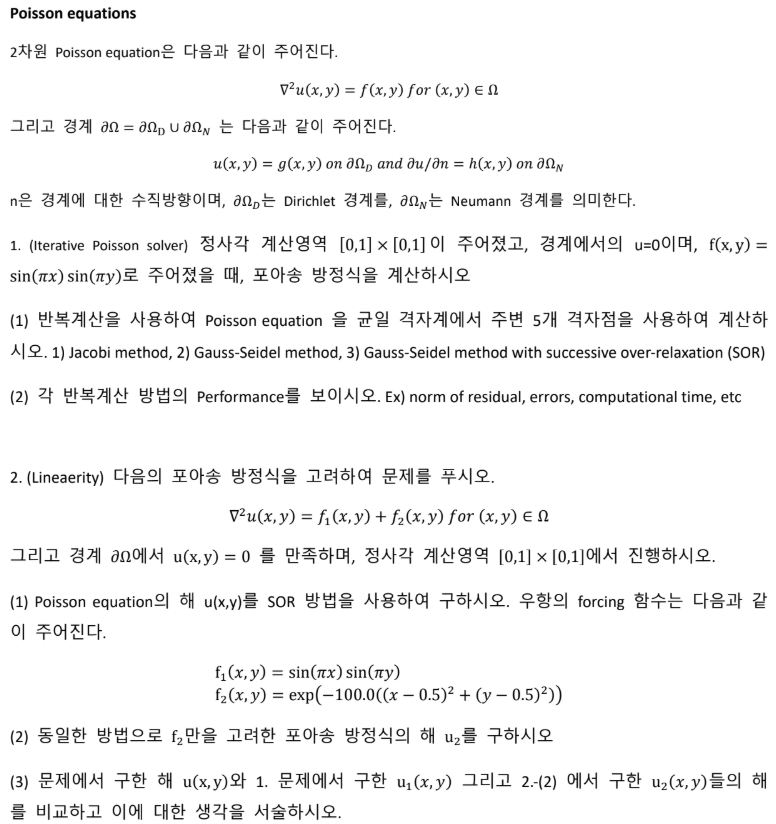


ω=1.86일 때 가장 작은 반복 횟수와 소요 시간을 보였으므로 완화계수 ω는 1.86으로 채택하였다.또한, 2-2번 문제에서 f의 종류에 따른 u의 단면 모양의 차이를 보여주기 위하여 u의 중앙을 지나는 단면의 값 (x=0.5, 즉 u[21]일 때의 u값)을 u\_middle\_f1로 따로 저장해주었다. 저장된 u\_middle\_f1은 아래와 같다.



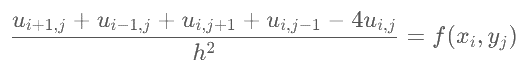
결과





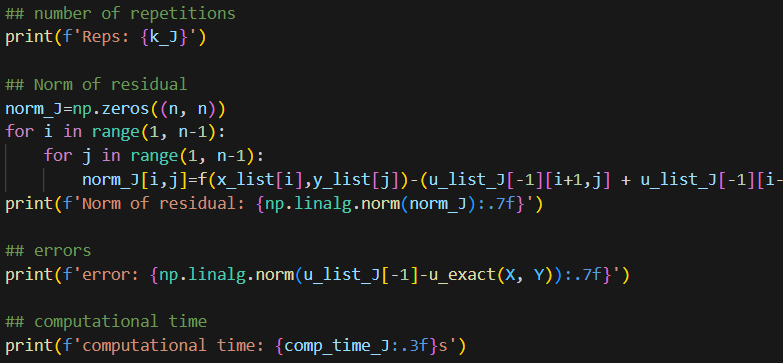
Number of repetitions, Norm of residual, Errors, Computational time을 순서대로 도출하였다.

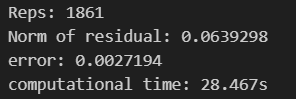
Norm of residual (잔차 노름)은 수치해와 실제 해가 얼마나 다른 지, 즉 현재 식이 PDE를 얼마나 위반하고 있는지를 측정하는 값이다. 다시 말해 아래 공식의 좌변과 우변이 얼마나 다른 지를 나타내는 값이라고 할 수 있다.



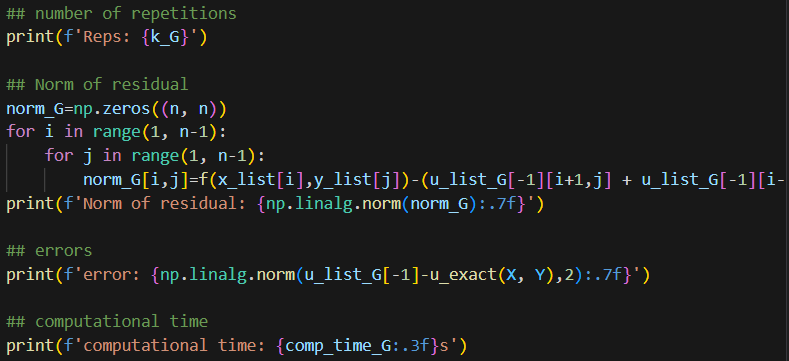
이때 Norm of residual과 Error 모두 L2 norm을 사용하였다.

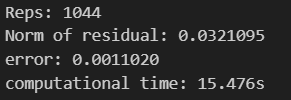
* Jacobi method



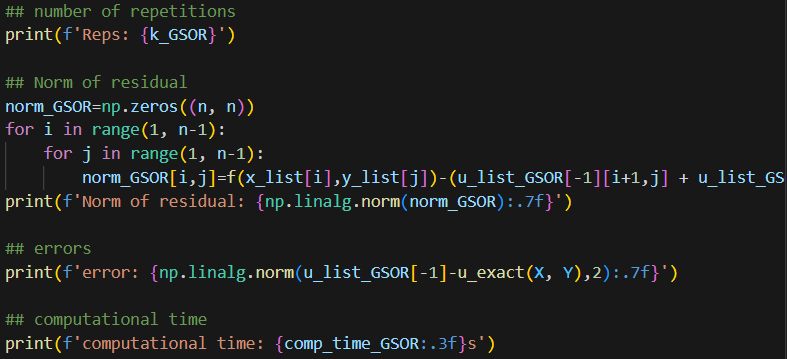
 결과

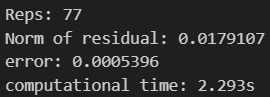
* Gauss-Seidel method



 결과

* Gauss-Seidel + SOR (Successive Over-Relaxation)



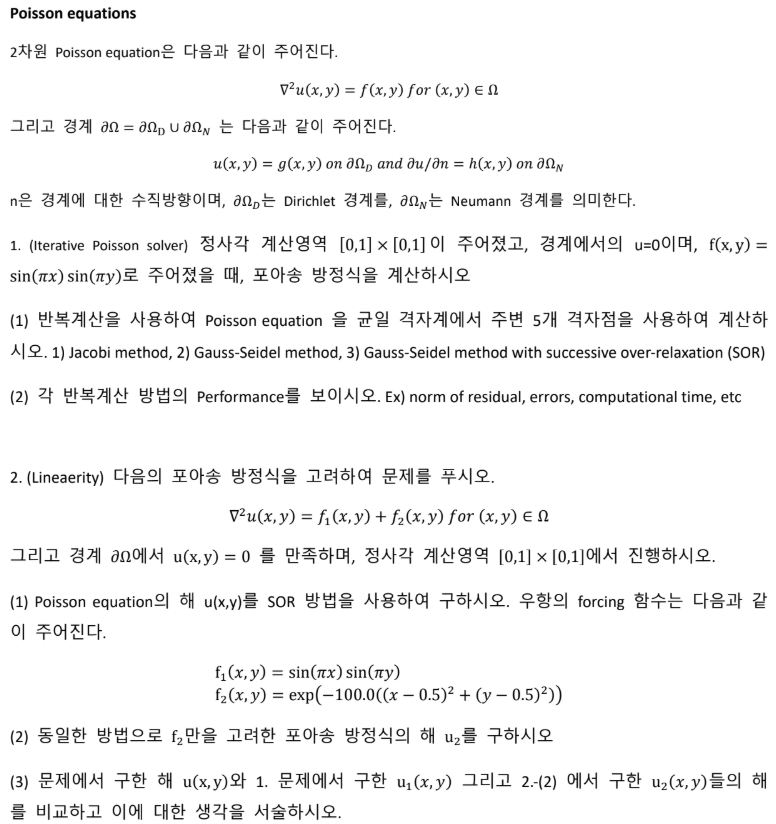
 결과

세 방법의 결과를 표로 나타내면 다음과 같다.

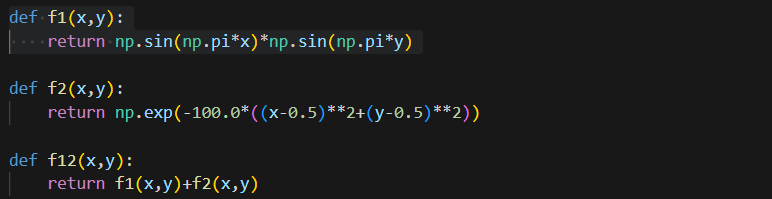
|  |  |  |  |
| --- | --- | --- | --- |
|  | Jacobi | Gauss-Seidel | Gauss-Seidel+SOR |
| Reps | 1861 | 1044 | 77 |
| Norm of residual | 0.0639298 | 0.0321095 | 0.0179107 |
| Error | 0.0027194 | 0.0011020 | 0.0005396 |
| Computational time(s) | 28.467 | 15.476 | 2.293 |

모든 항목의 값이 Jacobi > Gauss-Seidel > Gauss-Seidel+SOR로 나타난 것을 발견할 수 있다.

따라서 Poisson equation을 푸는 데에 Gauss-Seidel+SOR이 가장 적절함을 알 수 있다.



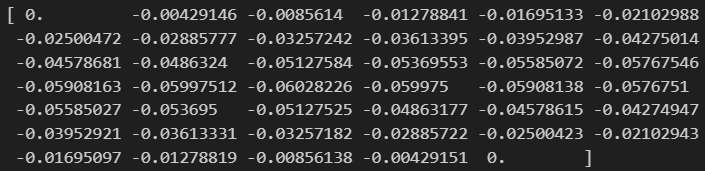
f1, f2, f1+f2를 다음과 같이 정의하였다.

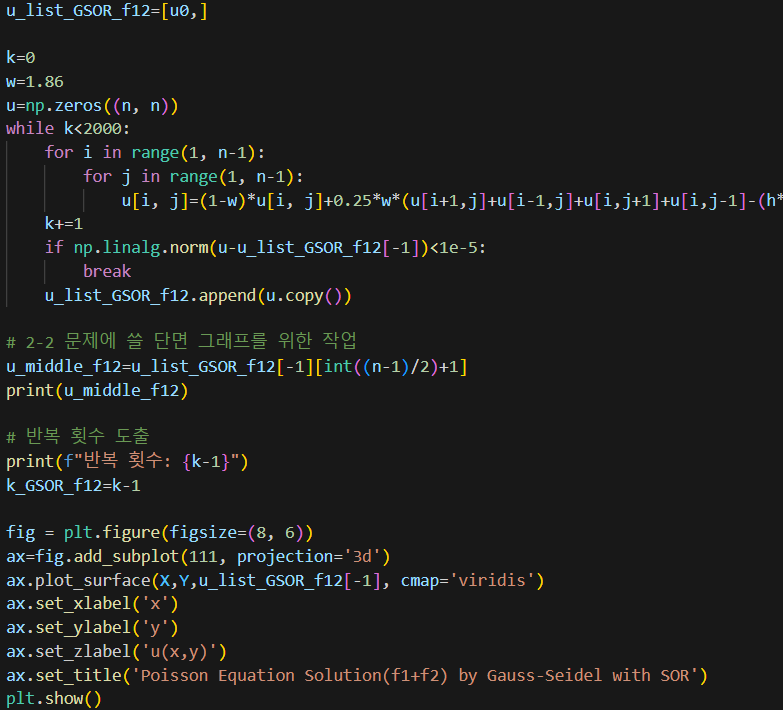


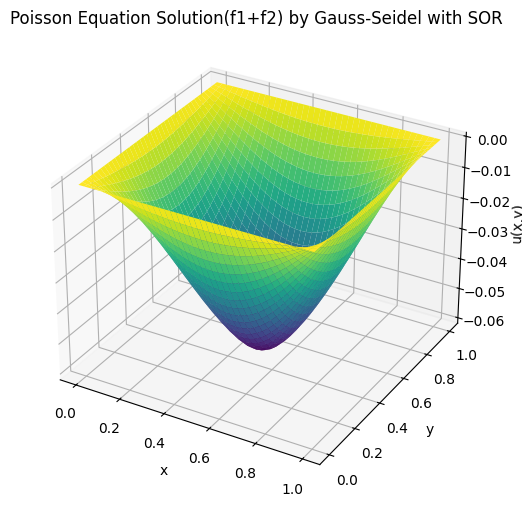
1-1번의 Gauss-Seidel method+SOR과 코드의 틀은 같다. f1만 사용하였던 1-1과는 달리 f1+f2의 형태인 f12가 대신 적용되었다.

f는 완화 계수의 크기에 영향을 주지 않으므로 1-1번에서 도출하였던 ω=1.86을 채택하였다.

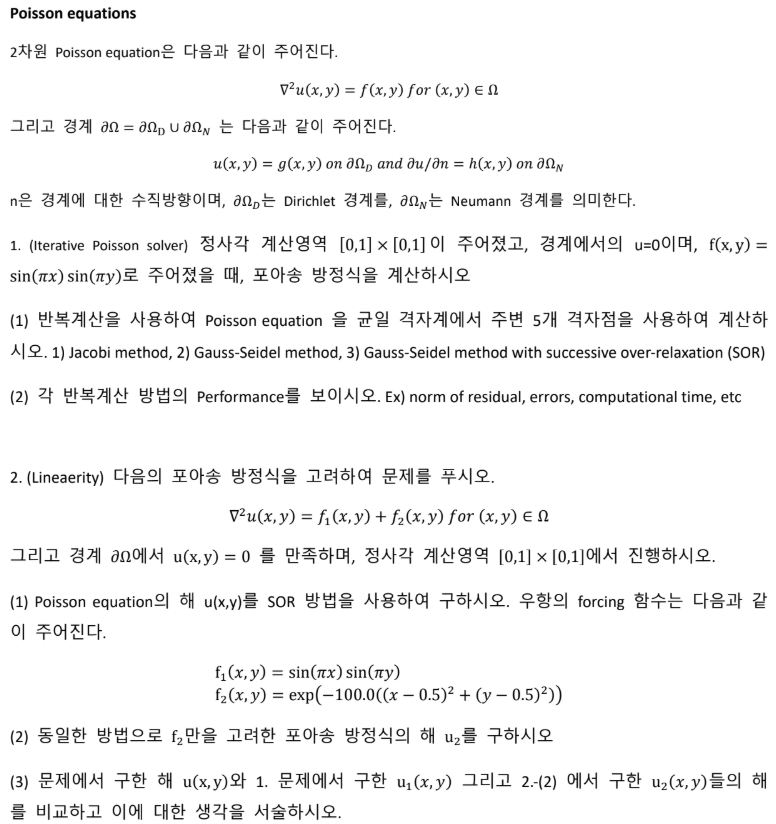
2-2번 문제에서 f의 종류에 따른 u의 단면 모양의 차이를 보여주기 위하여 u의 중앙을 지나는 단면의 값 (x=0.5, 즉 u[21]일 때의 u값)을 u\_middle\_f12로 따로 저장해주었다. 저장된 u\_middle\_f12은 아래와 같다.





결과

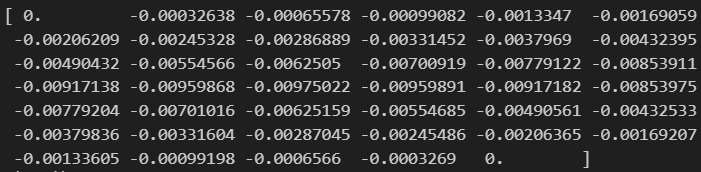


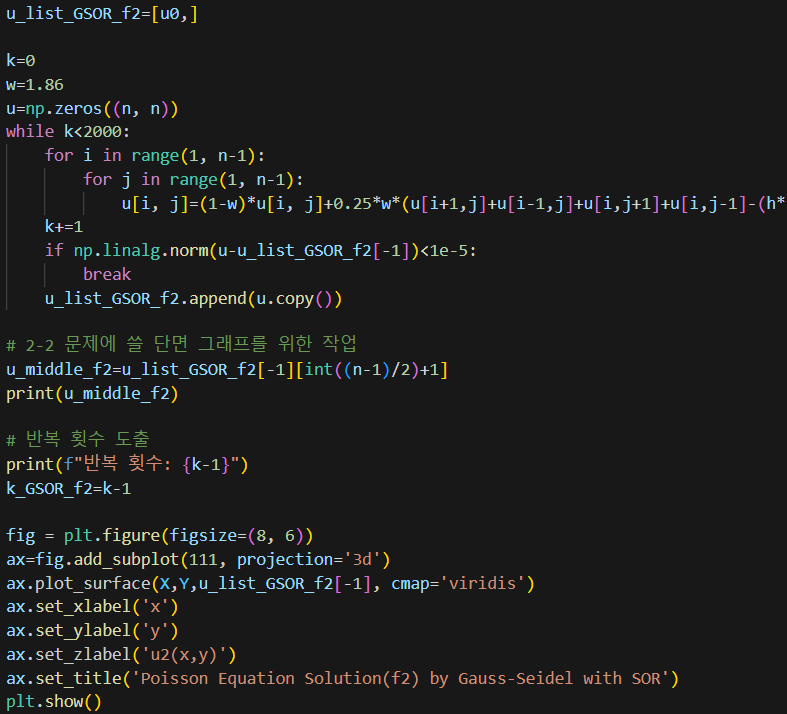


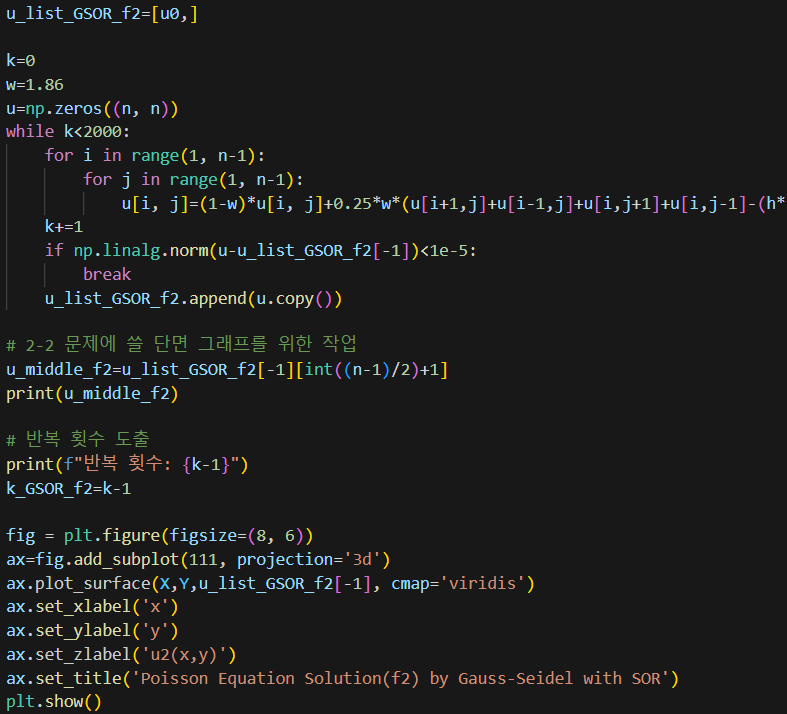
1-1번의 Gauss-Seidel method+SOR과 코드의 틀은 같다. 다만 f2가 대신 적용되었다.

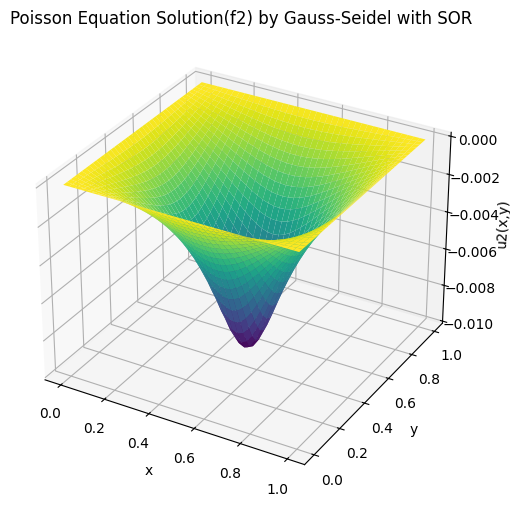
위에서와 마찬가지로 완화 계수 ω=1.86를 적용하였다.

2-2번 문제에서 f의 종류에 따른 u의 단면 모양의 차이를 보여주기 위하여 u의 중앙을 지나는 단면의 값 (x=0.5, 즉 u[21]일 때의 u값)을 u\_middle\_f2로 따로 저장해주었다. 저장된 u\_middle\_f2은 아래와 같다.



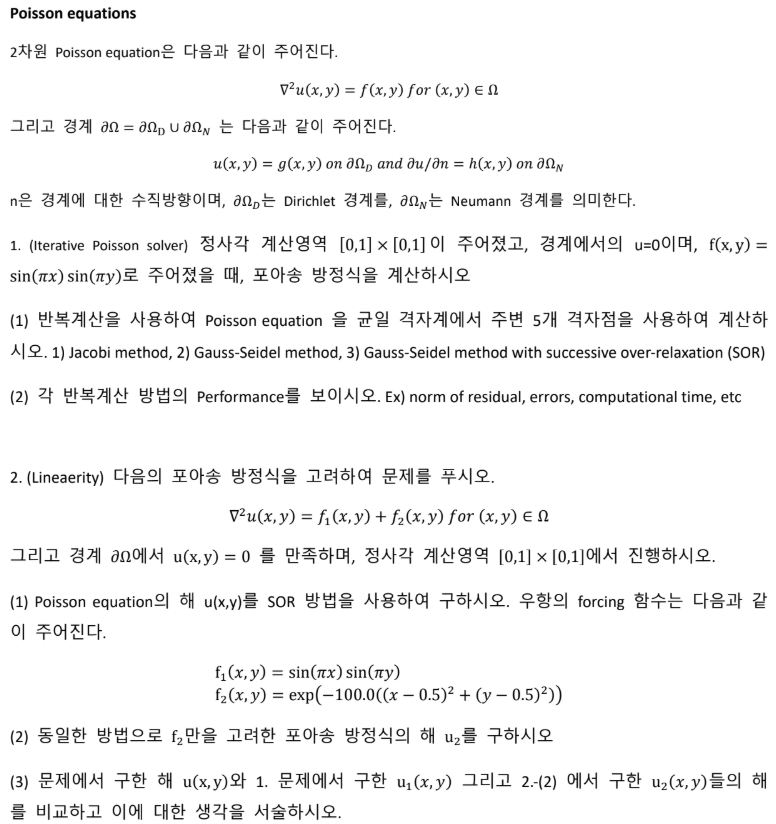






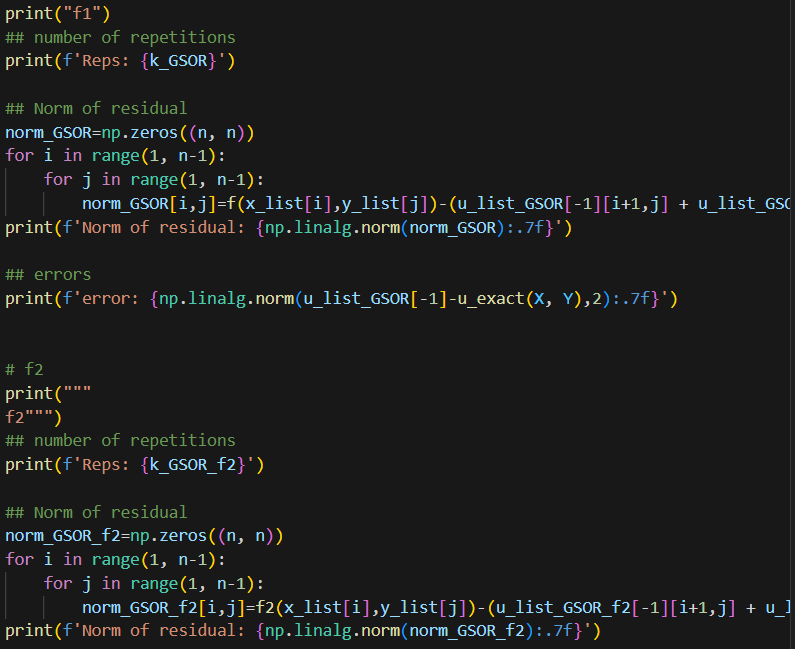
결과



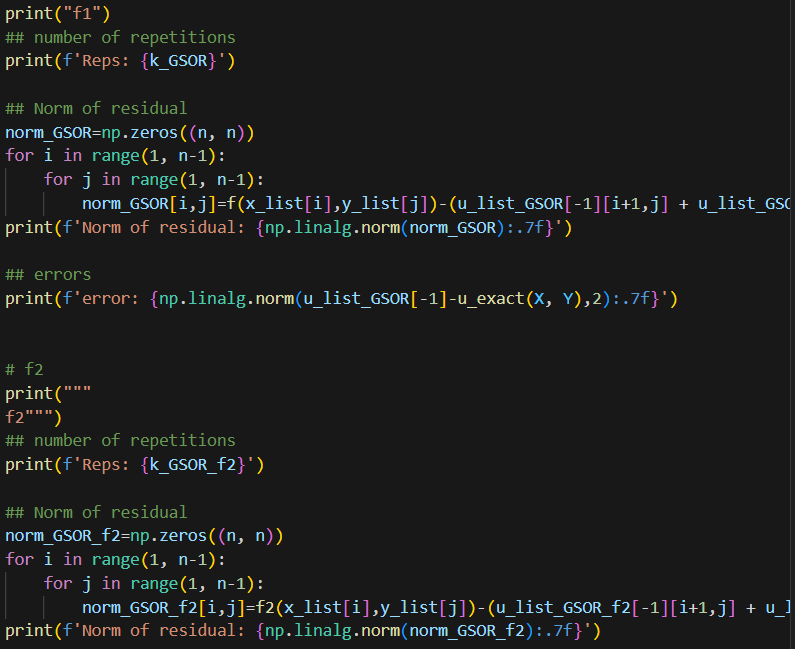


f1, f2, f1+f2의 Reps, Norm of residual을 구하였다. F2는 수학적 닫힌 해가 없으므로 exact solution을 이용하는 Error을 도출할 수 없다. 따라서 Error은 제외하였다. 또한 f가 다른 상황에서 reputational time을 비교하는 것이 유의미하지 않다고 판단하여 이 또한 제외하였다.

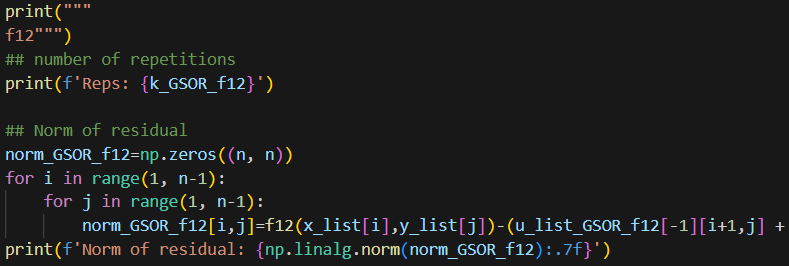
* f1

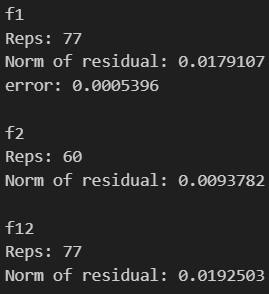


* f2



* f1+f2

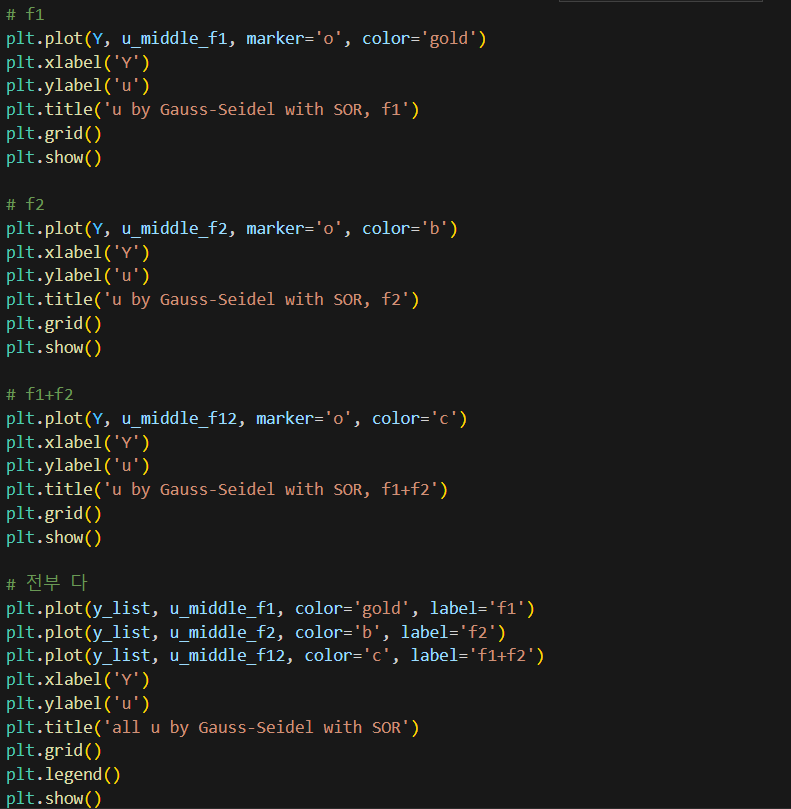


결과

|  |  |  |  |
| --- | --- | --- | --- |
|  | f1 | f2 | f1+f2 |
| Reps | 77 | 60 | 77 |
| Norm of residual | 0.0179107 | 0.0093782 | 0.0192503 |

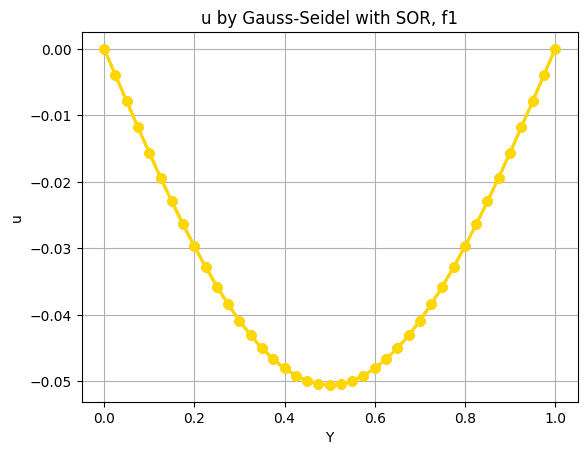
반복 횟수와 Norm of residual 모두 f1의 경우와 f1+f2의 경우가 근사하게 나타났다.

또한 3D 그래프의 단면을 직관적으로 비교해 보기 위해 단면 그래프를 그려보았다. 위의 코드에서 저장했던 u\_middle\_을 사용하였다.

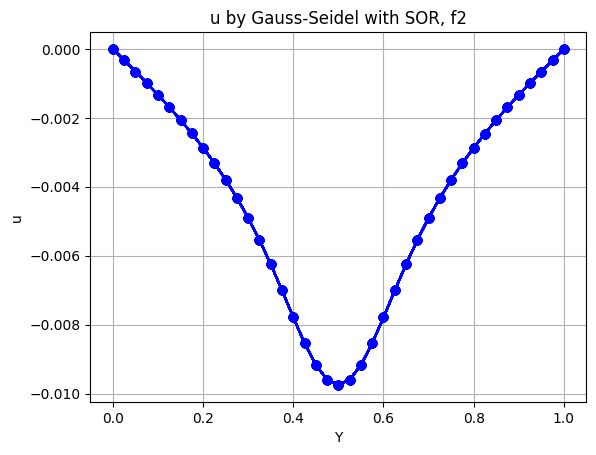


결과

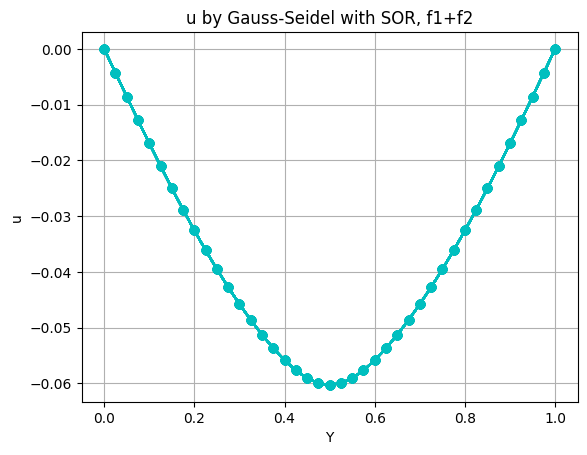
* f1



* f2



* f1+f2

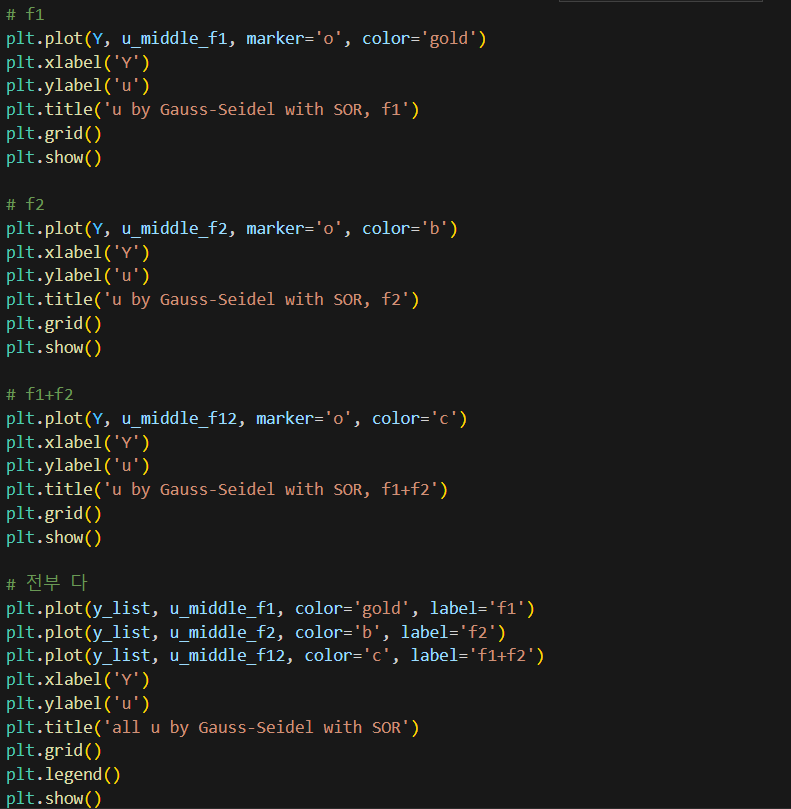


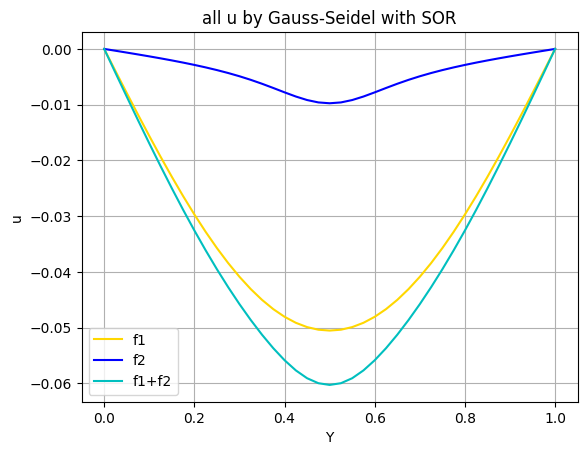
세 u의 3D 그래프와 단면 그래프를 비교하면 다음과 같다.

|  |  |  |
| --- | --- | --- |
| f1 | f2 | f1+f2 |
|  |  |  |
|  |  |  |

f1의 경우 2차 곡선과 비슷한 부드러운 곡선으로 나타난 반면, f2는 중앙에서 값이 급격히 작아져 좁고 우묵한 모양으로 나타났다. 두 f의 선형 조합인 f1+f2의 경우 해도 두 해가 더해진 모양으로 나타났다. 전체적인 모양은 f1와 더 유사하나 f1보다 더 좁고 날카로운 모양이다.

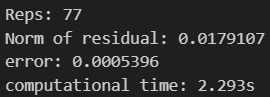
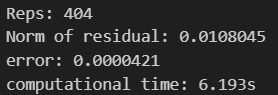
각 그래프의 깊이를 한 눈에 비교하기 위하여 단면 그래프 3가지를 겹쳐 보았다.

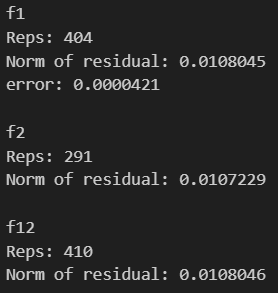
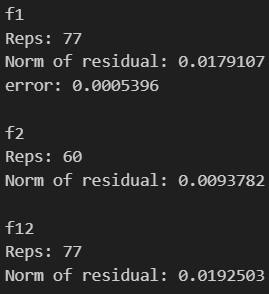


결과

f2에 의한 해가 f1에 비해서 전체적으로 약 5배 정도 작다. 이러한 요인 때문에 f1+f2의 모양이 f1에 가깝게 나온 것으로 보인다. 세 해를 도출하기 위한 반복 횟수와 Norm of residual가 f1와 f1+f2가 매우 비슷하게 나온 것 또한 이러한 해의 크기의 차이가 작용하였을 수도 있다.

+) 처음에는 완화 계수를 일방적으로 1.5를 넣어서 계산했었는데, 적절한 완화 계수 1.86을 찾은 후 이를 덮어쓰는 과정에서 1.5를 사용했을 때 Norm of residual과 error가 더 작게 나타남을 발견하였다. 이는 반복 횟수와 소요 시간을 가장 단축시키는 완화 계수가 정확도를 보장하지는 않는다는 것을 의미한다. 정확도가 중요한 상황에서는 이를 염두에 둘 필요가 있어 보인다.

(G+SOR, f1) <ω=1.5> <ω=1.86>

(G+SOR)