수치해석 Linear equaitions 보고서

2024041685 김태규

1. 서론

이번 과제는 Gauss-Jordan, LU, SVD 세 가지 방법을 사용하여 주어진 선형방정식 의 해를 구하고, 각 방법의 장/단점을 논하며 초기 해의 오차를 줄이기 위한 후처리 단계인 해 보정 (iterative improvement)을 사용하여 해를 개선(LU의 분해 결과 사용)한다. 또한 위 문제의 행렬 이 갖는 역행렬과 행렬식을 계산해본다.

1. Method

모든 방법들은 Numerical Recipes in C에 포함된 알고리즘을 사용하였으며, 각 알고리즘의 원리는 다음과 같다.

먼저 Gauss-Jordan 소거법 (gaussj())은 주어진 행렬 A를 단위행렬 형태로 변환하는 방식이다. 이와 동시에 같은 방법으로 b 또한 변환되며, 그 결과가 바로 벡터 x이다. 이 방법은 구현이 간단하고 역행렬의 계산에도 유용하지만, 그만큼 간단한 구현들 때문에 계산량이 많고 그로 인해 roundoff error가 누적되어 안전성이 상대적으로 낮다는 단점이 있다.

LU Decomposition (ludcmp(), lubksb())은 주어진 행렬 A를 상삼각행렬과 하삼각행렬로 분리한 뒤 두 단계의 대입을 통해 해를 구하는 방법이다. Ludcmp()는 주어진 행렬 A를 L과 U로 분리하는 역할을 하며 필요한 경우 행 교환(pivoting)을 시행하여 roundoff error를 최소화한다. 즉 A = LU 형태로 행렬을 변환하는 것이다. 이 단계는 전처리 과정이며, 한 번만 수행해놓으면 동일한 A와 여러 b에 대한 선형방정식의 해를 빠르게 구해낼 수 있다. 그 다음 단계인 lubksb()는 분해된 행렬을 이용하여 실제 해를 계산하는 후처리를 담당한다. 분해 후 대입만으로 빠르게 해를 구할 수 있으며 재사용이 가능하다는 점이 장점이다. 단점으로는 역행렬이 필요할 경우 추가적인 소모값이 발생할 수 있으며, 특이 행렬에 가까울수록 오차가 커진다는 점이 있다

세 번째 방법인 SVD (svdcmp(), svbksb())는 행렬 A를 세 개로 분리하는 방식으로, 계산량이 크지만 수치적 오차에 강하고, 조건수가 크더라도 다 감당하여 정확한 근사해를 찾아낸다. LU와 마찬가지로 svdcmp()는 행렬 A를 U, W, V로 분해하는 역할을 하며, svbksb()는 분해된 행렬들을 통해 x를 찾아낸다.

1. Results

세 가지의 선형방정식 풀이 방법(Gauss-Jordan, LU Decomposition, SVD)을 세가지 데이터에 적용하였다. 먼저 에서의 선형방정식은 SVD 방법을 선제적으로 적용해본 결과 Singular Matrx로 판명되었다. 따라서 SVD를 통해 최소노름(minimum-norm)해를 구하는 것만이 가능했다. 그 결과 x = [1.7333, -1.5333, -0.2000, -0.7333] 이라는 해를 구할 수 있었다. 에서의 선형방정식은 Singular Matrix가 아니었기에 세 가지 방법을 통해 모두 x = [-2.8736, -0.6124, 0.9763, 0.6358, -0.5534] 라는 공통된 해를 구해낼 수 있었다. 에서의 선형방정식 또한 x = [-0.3266, 1.5323, -1.0448, -1.5874, 2.9285, -2.2189] 로 해를 구해내었다.

Gauss-Jordan 소거법은 행 연산으로 구성되어 구현이 단순하고 실험 출력에서 Execution time이 약 0.001ms가 관측되었듯이 작은/중간 크기 문제에서 매우 빠른 수렴속도를 보여준다. (LU 보단 느리고, SVD보단 빠름) 또한 평균 오차가 2.99 x 10^-6정도로 작아 수치적 정확성이 확보되었다. (SVD보다 정확하지만 LU보다는 오차가 약간 큼) 하지만 피벗이 0으로 떨어지면 수치적 불안성이 발생할 수 있으며, 특이행렬에서는 해를 제공할 수 없다. (첫 번째 데이터의 해를 제공하지 못함)

LU 분해법은 분해 후 대입을 통해 풀기에 계산 효율이 좋다. 본 출력에서는 LU시간이 해상도의 한계로 인해 0.000ms로 표기되었으나(셋 중 제일 빠름), 이는 측정불능이라고 보아야 한다. 실제로는 Gauss-Jordan보다 빠르거나 동급의 성능을 갖는다. 또한 오차 평균이 2.01 x 10^-6으로 세 방법 중 가장 작았다. 이를 통해 실무에서 기대하는 높은 수치의 정확성과 빠른 속도를 겸비했음을 보여준다. 단 이 방법도 특이행렬에서 해를 구할 수 없었다. (첫 번째 데이터의 해를 제공하지 못함)

SVD는 특이행렬에서도 해를 구할 수 있었다. (첫 번째 데이터의 최소노름 해를 제공) 또한 두번째와 세번째 방정식에서는 LU, Gauss-Jordan과 동일한 해를 산출하여 정확성을 입증했다. 오차 평균은 5.48 x 10^-6으로 세 방법 중 가장 크지만 그래도 충분히 허용할 정도의 오차를 보여주었다. 분해하는 방법이 고비용이기에 세 방법 중 가장 느리다.

**정확도(잔차)**: 세 방법 모두 수준으로 충분히 작으며, full-rank 문제에서는 **동일한 해**를 산출하여 **수치적 일관성**이 검증되었다. 평균 잔차는 LU Gauss–Jordan SVD 순으로 근소한 차이를 보였으나, 모두 실무 허용 범위 내이다.

**속도**: **SVD**  Gauss–Jordan LU의 경향이 확인되었다. 이번 출력에서는 타이머 해상도(약 10 ms) 한계로 Gauss–Jordan/LU가 0.000–0.001 ms로 표기되었으나, 이는 **“매우 빠름”의 간접 증거**로 해석해야 한다.

또한, Iterative Improvement(mprove())를 LU 분해 해에 3회 적용한 결과, 두 번째 데이터의 경우 잔차(norm)가 약 에서 , , 로 **유의미하게 감소**하였다.  
반면 세 번째 데이터는 에서 으로 크게 줄지는 않았으나, 전반적으로 수치적 안정성이 개선되는 양상을 보였다.  
(반복개선의 효과는 시스템 조건 및 반올림 오차의 영향으로 실행마다 다를 수 있다.)

마지막으로 LU의 분리된 행렬을 통해 주어진 행렬 A의 Det(A)와 역행렬을 구해보았다. (결과는 실행파일 출력을 통해 첨부)

1. 결론

이번 과제를 통해 Gauss-Jordan, LU Decomposition, SVD 세가지 선형방정식 풀이법의 장단점을 비교하고 특성을 알아보았다.

그 결과 LU가 가장 빠르고 정확한 결과를 제공하였다. 또한 Gauss-Jordan 방법은 단순한 구현이라는 장점이 있지만 되려 그 점이 안정성을 떨어뜨렸다. 마지막으로 SVD는 계산량이 많고 오래 걸렸지만 다른 방법들이 구해내지 못하는 특이행렬에서도 안정적인 근사치를 해로 제공해냈다.

LU 분해 결과를 가지고 mprove를 수행한 결과, 일부 데이터에서 오차가 매우 크게 감소하여 후처리의 효능을 직접 확인할 수 있었다. 마지막으로, LU의 분해식을 이용해 행렬식과 역행렬을 계산해보았다.

요약하자면 LU는 속도와 정확성을 고루 갖추었고, Gauss는 복잡한 계산에 부적합한 로직을 가졌으며 SVD는 계산 속도나 비용이 크지만 특이한 이상치 케이스들에서도 계산을 잘 해냈다. 따라서 사용자가 어떠한 상황에서 어떠한 요인들을 고려하고 있는지에 따라 사용할 방법은 달라지며, 이번 과제를 통해 세 방법의 수치해석적 특성과 장단점을 명확히 체험, 비교할 수 있었다.