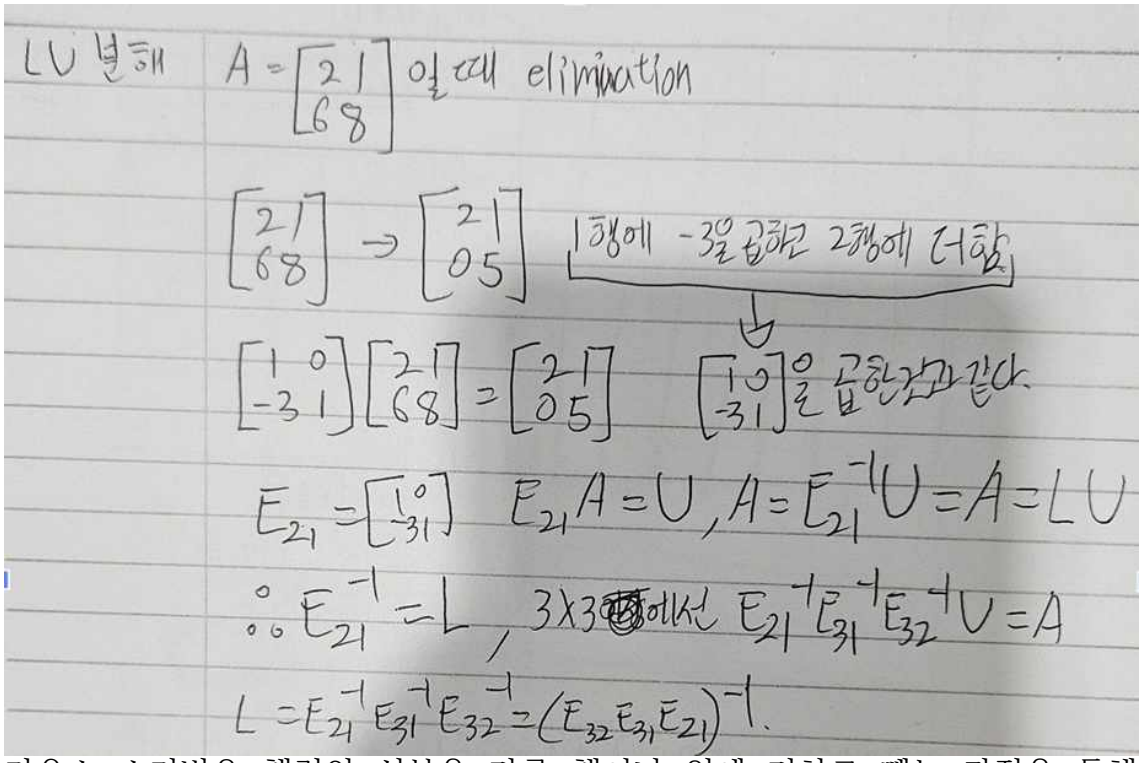
linear algebra assignment2

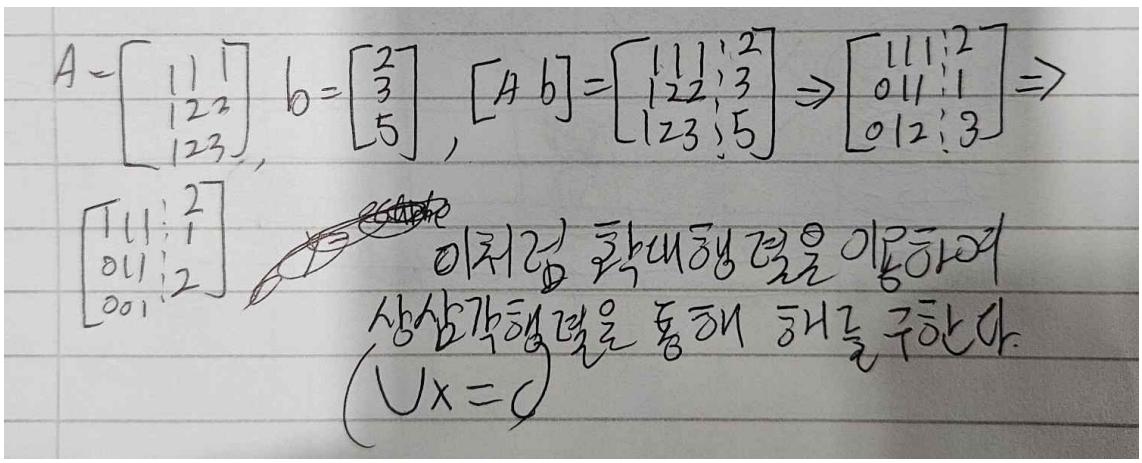
과목명: 선형대수학
교수님: 박철수 교수님
student ID : 2023202032
Name : 남호준

1. 소개: LU, 가우스 소거법 설명, 예시 작성

LU factorization은 LU 분해라고 해석할 수 있으며, 한 행렬 A를 삼각행렬 L, U로 분해하는 방법이다. 이때, L과 U는 각각 하삼각행렬, 상삼각행렬이다.



가우스 소거법은 행렬의 성분을 다른 행이나 열에 더하고 빼는 과정을 통해 상삼각행렬을 만들어 해를 찾는 것이다.



위처럼 행렬의 피벗과 mtp를 대입한 행렬을 만들어 A를 분해하면 LU 분해법, 피벗을 찾아 스칼라 곱을 한 후, 나머지 행에 빼거나 더하여 상삼각행렬로 만들면 가우스 소거법이다.

2. 코드: 코드 작성, 주석

test.m

```
%계수 행렬 A와 벡터 b를 생성
size=7;
A = rand(size,size);%임의의 정사각 행렬로 만듬
b = rand(size,1);%임의의 열 벡터로 만듬

%LU Factorization의 연산 시간 측정
disp('LU Factorization 연산 시간:')
tic;
[L,U]=slu(A);%L,U 값
x_LUF=slv(A,b);% LU facto 의 해 구함
toc;

%가우스 소거법 연산 시간 측정
disp('가우스 소거법 연산 시간:')
tic;
x_gau=gauss_elim(A,b);
toc;
x_gau=x_gau';
disp('행렬 A:');
disp(A);
disp('벡터 b:');
disp(b);
disp('행렬 L:');
disp(L);
disp('행렬 U:');
disp(U);
disp('LU Factorization의 결과값:');
disp(x_LUF);
disp('가우스 소거법의 결과값:');
disp(x_gau);
```

gauss_elim.m

```
% Gauss elimination
function x = gauss_elim(A, b)
tol = 1.e-6;%오차 정밀도 설정(10의 -6제곱의 오차)
Ab = [A, b];% Augmented matrix 만듬

[rows,cols]=size(A);%A의 행과 열 크기 가져옴
    % Forward elimination
    for i = 1:rows-1
        [~, maxRow] = max(abs(Ab(i:cols, i)));% 현재 열에서 최대 절대값을 가진
행 찾음
        maxRow = maxRow + i - 1;
        if abs(Ab(maxRow, i)) < tol % 최대값이 허용 오차보다 작은 경우
            error('피벗의 값이 너무 작음');
        end
        if maxRow~=i % 최대값이 현재 행이 아닌 경우
        Ab([i maxRow], :) = Ab([maxRow i], :); %행 교환
        end
        for j = i+1:cols
            factor = Ab(j, i) / Ab(i, i); % 소거 요소 연산
            Ab(j, :) = Ab(j, :) - factor * Ab(i, :); %행 소거
        end
    end

    % Backward substitution
    x = zeros(cols, 1); % 결과 벡터 초기화
    x(cols) = Ab(cols, cols+1) / Ab(cols, cols); % 마지막 열에 대한 값을 계산
    for i = cols-1:-1:1 % 역순 계산
        x(i) = (Ab(i, cols+1) - Ab(i, i+1:cols) * x(i+1:cols)) / Ab(i, i); % 현재
행 값 계산
    end
end
```

slu.m

```
%LU factorization
function [L, U] = slu(A)
tol = 1.e-6;%정밀도 설정(10의 -6제곱)
[rows, cols] = size(A); % A의 크기 추출
if(rows~=cols)%행과 열 크기가 다른 경우
    error('this isnt square max_id_rowsatrix')
end
L=eye(cols);%L을 단위 행렬로 만듬
U = zeros(cols);%U를 영행렬로 만듬
for j= 1:cols
  if(abs(A(j,j))) <tol
      [max_id_rows, max_id_cols] = max(abs(A(j+1:cols, j)));
      if max_id_rows == 0 % 행렬 A가 singular할 경우
             error('행렬 A는 비가역적이다');
      end
      [A(j,:), A(max_id_cols+j,:)] = deal(A(max_id_cols+j,:), A(j,:)); % 행 교환을
수행
      [L(j,1:j-1),    L(max_id_cols+j,1:j-1)]    =    deal(L(max_id_cols+j,1:j-1),
L(j,1:j-1)); % L에서도 대응하는 행을 교환
  end %diagonal elements가 0이 있으면 LU factorization이 안됨
    for i = j + 1 : cols
        L(i, j) = A(i, j)/ A(j, j); % j열의 원소들에 대한 j열의 스칼라값을 계산하여
L의 i행 j열 원소에 대입
        for k = j + 1 : cols % i행, j열의 여인수 행렬에 대한 연산
            A(i, k) = A(i, k) - L(i, j) * A(j, k);  % A 행렬 i,j 인덱스 원소를 계
산
        end
    end
    U(j, j:cols) = A(j, j:cols);  % j행 계산 후 결과를 U에 저장
end

end
```

slv.m

```
% forward backward
function x = slv(A, b)
 % Forward substitution
[L, U] = slu(A);
[~,cols]=size(A);
a=zeros(cols,1);%a 벡터를 영 벡터로 초기화
x=zeros(cols,1);%x 벡터를 영 벡터로 초기화
sum=0;
for i=1:cols
    for j=1:i-1
    sum = sum + L(i, j) * a(j); %이전 합계에 a[j]와 L[i,j]의 곱을 더함
    end
    a(i) = (b(i) - sum) / L(i, i);
    sum=0;
end
 % Backward substitution
t=0;
for i = cols : -1 : 1 % x(n)에서부터 x(1)까지 역순으로 진행
    for j = i+1:cols % Baci substitution
        t = t + U(i, j) * x(j); % U와 이후 x(j)를 곱해서 더함
    end
    x(i) = (a(i) - t)/ U(i, i); % 피벗으로 나눔
    t = 0; %다음 i 이전 t reset
end
x = x'; % 열벡터로 transpose함
end
```

3. 결과 화면: 결과, 실행시간 캡처

A=3*3, b=3*1일 때

```
LU Factorization 연산 시간:
경과 시간은 0.000328초입니다.
가우스 소거법 연산 시간:
경과 시간은 0.000117초입니다.
행렬 A:
    0.8649    0.9084    0.8221
    0.5711    0.6636    0.2877
    0.1710    0.5027    0.8810

벡터 b:
    0.5314
    0.0926
    0.4835

행렬 L:
    1.0000         0         0
    0.6604    1.0000         0
    0.1977    5.0646    1.0000

행렬 U:
    0.8649    0.9084    0.8221
         0    0.0638   -0.2552
         0         0    2.0108

LU Factorization의 결과값:
    0.5460   -0.6940    0.8389

가우스 소거법의 결과값:
    0.5460   -0.6940    0.8389
```

A=7*7, b=7*1일 때

```
LU Factorization 연산 시간:
경과 시간은 0.000266초입니다.
가우스 소거법 연산 시간:
경과 시간은 0.000164초입니다.
행렬 A:
    0.2525    0.7913    0.4084    0.1773    0.2969    0.4002    0.1170
    0.0177    0.7274    0.2399    0.5857    0.6830    0.8776    0.7105
    0.4507    0.8231    0.3576    0.7215    0.9247    0.9532    0.0977
    0.9046    0.9363    0.1643    0.7213    0.2917    0.0044    0.5848
    0.6571    0.1912    0.0025    0.5086    0.4760    0.6014    0.8476
    0.1434    0.5269    0.4067    0.3259    0.3915    0.7856    0.2307
    0.2585    0.2806    0.2796    0.4548    0.3189    0.6385    0.1960

벡터 b:
    0.0378
    0.8275
    0.0283
    0.3672
    0.1748
    0.5041
    0.4960

행렬 L:
    1.0000         0         0         0         0         0         0
    0.0702    1.0000         0         0         0         0         0
    1.7850   -0.8771    1.0000         0         0         0         0
    3.5822   -2.8252    3.7703    1.0000         0         0         0
    2.6020   -2.7799    2.5406    0.3877    1.0000         0         0
    0.5680    0.1154   -0.8075   -0.5196   -6.1872    1.0000         0
    1.0237   -0.7879   -0.1503   -0.5017   -9.3099    1.4782    1.0000

행렬 U:
    0.2525    0.7913    0.4084    0.1773    0.2969    0.4002    0.1170
         0    0.6719    0.2112    0.5732    0.6622    0.8495    0.7023
         0         0   -0.1862    0.9078    0.9756    0.9839    0.5049
         0         0         0   -1.7171   -2.5792   -2.7388    0.2463
         0         0         0         0    0.0656    0.4837    1.1173
         0         0         0         0         0    2.8243    7.5318
         0         0         0         0         0         0    0.0972

LU Factorization의 결과값:
   -0.4301    2.5206   -4.9602    1.4552   -4.9090    3.6521   -0.7218

가우스 소거법의 결과값:
   -0.4301    2.5206   -4.9602    1.4552   -4.9090    3.6521   -0.7218
```

4. 고려 사항: 속도 차이 분석

위의 캡처 사진에서도 보이듯이, 3, 7 등의 작은 수에선 LU factorization보다 가우스 소거법이 더 빠른 것을 볼 수 있다.
n==100인 경우에도 가우스 소거법이 빠른 것을 볼 수 있다.

```
LU Factorization 연산 시간:
경과 시간은 0.001670초입니다.
가우스 소거법 연산 시간:
경과 시간은 0.001201초입니다.
```

하지만, 두 알고리즘의 시간복잡도는
LU factorization: (4/3)*n^3 => O(n^3)
가우스 소거법: (1/3)*n^4 => O(n^4)
이다.

따라서, n이 커질수록 LU 방법의 시간이 더욱 적게 소요될 것이다.
가우스 소거법은 특정 행렬 구조 등에서 더 빠른 연산을 행할 수도 있지만, 일반적인 상황에서 n이 충분히 클수록 LU Factorization이 가우스 소거법보다 빠르다.
가우스 소거법은 간단하여 구현이 쉽고 작은 행렬에서 쉽게 연산할 수 있는 효과가 있다. 반대로 LU Factorization은 일반적인 상황에서 대규모 연산을 행하거나 일반화된 식, 경우 등을 처리할 때 효과적이다.