**LE MINH DONG**

HOME WORK 1: Gaussian Elimination

1. **Main code**

A = [1,1,1 ;1,2,1;1,3,4]

b = [1;0;0]

%[y] = gauss1(A,b);

i = 1; % loop variable

X = [ A b ];

[ nX mX ] = size( X); % determining the size of matrix

while i <= nX % start of loop

if X(i,i) == 0 % checking if the diagonal elements are zero or not

disp('Diagonal element zero') % displaying the result if there exists zero

return

end

X = elimination(X,i,i); % proceeding forward if diagonal elements are non-zero

i = i +1;

end

C = X(:,mX)

1. **function elimination**

function X = elimination(X,i,j)

% Pivoting (i,j) element of matrix X and eliminating other column

% elements to zero

[ nX mX ] = size( X);

a = X(i,j);

X(i,:) = X(i,:)/a;

for k = 1:nX % loop to find triangular form

if k == i

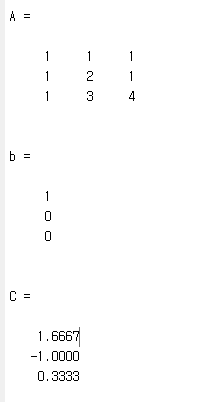
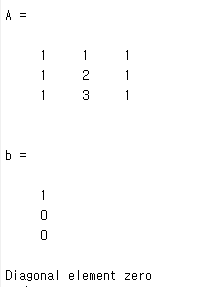
continue

end

X(k,:) = X(k,:) - X(i,:)\*X(k,j); % final result

end

1. **Example results**

Home Work 2: Invert Matrix

1. **Main code**

A1 = [1,1,1 ;1,2,1;1,3,4]

b1 = [1,0,0;0,1,0;0,0,1]

[det1,X1 ] = gauss\_invert( A1, b1 )

1. **Function**

function [det1,X ] = gauss\_invert( A, b )

%GEPP

perm=0;

n = length(b);

%if n~=m

%error('vector has wrong size');

%end

for j = 1:n

p=j;

% choice of main element

for i = j:n

if abs(A(i,j)) >= abs(A(p,j))

p = i;

end

end

if A(p,j) == 0

error('Matrix A is singular');

end

%rows permutation

t = A(p,:);

A(p,:) = A(j,:);

A(j,:) = t;

t = b(p);

b(p) = b(j);

b(j) = t;

if~(p==i)

perm=perm+1;

end

% reduction

for i = j+1:n

t = (A(i,j)/A(j,j));

A(i,:) = A(i,:)-A(j,:)\*t;

b(i) = b(i)-b(j)\*t;

end

end

%determinant

mn=1;

for i=1:n

mn=mn\*A(i,i);

end

det1=mn\*(-1)^perm;

% solution

X = zeros(1,n);

X(n) = b(n)/A(n,n);

if (det1~=0)

for i = 1:n

s = sum( A(i, (i+1):n) .\* X((i+1):n) );

X(i) = (b(i) - s) / A(i,i);

end

end

end

1. **Example Result**

