MATH 211.3 Winter Term 2024 Assignment

Assignment #03

Name: Kanwar Pannu

Student Number: 11336732

NSID: eza147

**Problem 1**

**A close-up of a paper

Description automatically generated**

%Question 1

% System (a)

A\_1a = [2 -3; 5 -6];

b\_1a = [2; 8];

x\_1a = naiveGaussian(A\_1a, b\_1a);

% System (b)

A\_1b = [1 2; 2 3];

b\_1b = [-1; 1];

x\_1b = naiveGaussian(A\_1b, b\_1b);

% System (c)

A\_1c = [-1 1; 3 4];

b\_1c = [2; 15];

x\_1c = naiveGaussian(A\_1c, b\_1c);

%Question 2

% System (a)

A\_2a = [2 -2 -1; 4 1 -2; -2 1 -1];

b\_2a = [-2; 1; -3];

x\_2a = naiveGaussian(A\_2a, b\_2a);

% System (b)

A\_2b = [1 2 -1; 0 3 1; 2 -1 1];

b\_2b = [2; 4; 2];

x\_2b = naiveGaussian(A\_2b, b\_2b);

% System (c)

A\_2c = [2 1 -4; 1 -1 1; -1 3 -2];

b\_2c = [-7; -2; 6];

x\_2c = naiveGaussian(A\_2c, b\_2c);

% Display the solutions

disp('Solution for system 1(a):');

disp(x\_1a);

disp('Solution for system 1(b):');

disp(x\_1b);

disp('Solution for system 1(c):');

disp(x\_1c);

% Display the solutions

disp('Solution for system 2(a):');

disp(x\_2a);

disp('Solution for system 2(b):');

disp(x\_2b);

disp('Solution for system 2(c):');

disp(x\_2c);

function x = naiveGaussian(A, b)

n = size(A, 1);

% Elimination step

for j = 1:n-1

if abs(A(j,j)) < eps

error('Zero pivot encountered');

end

for i = j+1:n

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

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

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

end

end

% Back-substitution step

x = zeros(n, 1);

for i = n:-1:1

x(i) = (b(i) - A(i,i+1:n)\*x(i+1:n)) / A(i,i);

end

end

**Problem 2**

**A close-up of a paper with mathematical equations

Description automatically generated**

**A close-up of a paper

Description automatically generated**

**A close-up of a paper with writing

Description automatically generated**

clear;

clc;

A1 = [1 2; 3 4];

A2 = [1 3; 2 2];

A3 = [3 -4; -5 2];

% Matrix (a) Exercise 1

[L1, U1] = naiveLU(A1);

disp('L for matrix (a) Exercise 1:');

disp(L1);

disp('U for matrix (a) Exercise 1:');

disp(U1);

% Matrix (b) Exercise 1

[L2, U2] = naiveLU(A2);

disp('L for matrix (b) Exercise 1:');

disp(L2);

disp('U for matrix (b) Exercise 1:');

disp(U2);

% Matrix (c) Exercise 1

[L3, U3] = naiveLU(A3);

disp('L for matrix (c) Exercise 1:');

disp(L3);

disp('U for matrix (c) Exercise 1:');

disp(U3);

A4 = [3 1 2; 6 3 4; 3 1 5];

A5 = [4 2 0; 4 4 2; 2 2 3];

A6 = [1 -1 1 2; 0 2 1 0; 1 3 4 4; 0 2 1 -1];

% Matrix (a) Exercise 2

[L4, U4] = naiveLU(A4);

disp('L for matrix (a) from Exercise 2:');

disp(L4);

disp('U for matrix (a) from Exercise 2:');

disp(U4);

% Matrix (b) Exercise 2

[L5, U5] = naiveLU(A5);

disp('L for matrix (b) Exercise 2:');

disp(L5);

disp('U for matrix (b) Exercise 2:');

disp(U5);

% Matrix (c) Exercise 2

[L6, U6] = naiveLU(A6);

disp('L for matrix (c) Exercise 2:');

disp(L6);

disp('U for matrix (c) Exercise 2:');

disp(U6);

function [L, U] = naiveLU(A)

[m, n] = size(A);

if m ~= n

error('Matrix must be square');

end

L = eye(n);

U = A;

% Perform LU factorization without pivoting

for k = 1:n-1

if abs(U(k,k)) < eps % Check for zero pivot

error('Zero pivot encountered');

end

for i = k+1:n

L(i,k) = U(i,k)/U(k,k);

U(i,k:n) = U(i,k:n) - L(i,k)\*U(k,k:n);

end

end

% Check the factorization by multiplying L and U

if max(max(abs(A - L\*U))) > 1e-10

error('LU factorization check failed');

end

end

**Problem 3**

**A close-up of a piece of paper

Description automatically generated**

**A close-up of a notebook

Description automatically generated**

**1**

clear;

clc;

format long;

% Call the function for n = 6 and n = 10

error\_and\_condition(6);

error\_and\_condition(10);

function error\_and\_condition(n)

A = zeros(n,n);

for i = 1:n

for j = 1:n

A(i,j) = 5 / (i + 2\*j - 1);

end

end

x = ones(n,1);

b = A \* x;

xc = A\b;

% Calculate the infinity norm of the forward error

forward\_error = norm(x - xc, Inf);

% Calculate the error magnification factor

error\_magnification\_factor = forward\_error / norm(x, Inf);

% Calculate the condition number of A

cond\_A = cond(A, Inf);

% Display the results

disp(['For n = ', num2str(n)]);

disp(['Infinity norm of the forward error: ', num2str(forward\_error)]);

disp(['Error magnification factor: ', num2str(error\_magnification\_factor)]);

disp(['Condition number of A: ', num2str(cond\_A)]);

end

**2**

clear;

clc;

format long;

error\_magnification\_and\_condition([100, 200, 300, 400, 500]);

function error\_magnification\_and\_condition(ns)

for n = ns

% Create matrix A with given entries Aij = |i - j| + 1

A = zeros(n,n);

for i = 1:n

for j = 1:n

A(i,j) = abs(i - j) + 1;

end

end

x = ones(n,1);

b = A \* x;

xc = A\b;

% Calculate the infinity norm of the forward error

forward\_error = norm(x - xc, Inf);

% Calculate the error magnification factor

error\_magnification\_factor = forward\_error / norm(x, Inf);

% Calculate the condition number of A

cond\_A = cond(A, Inf);

disp(['For n = ', num2str(n)]);

disp(['Infinity norm of the forward error: ',num2str(forward\_error)]);

disp(['Error magnification factor: ', num2str(error\_magnification\_factor)]);

disp(['Condition number of A: ', num2str(cond\_A)]);

end

end