Coding Report 2

May 10, 2023

1 Problem Description

This report explores the use of different scripts to compute the solution to a linear equation Ax=b, where A is a square matrix and b is a (n $\times$ 1) vector. In this report, we mainly use four functions: Gaussian Elimination without Pivoting, Back Substitution, LU Factorization, and a solver for LU Factorization using forward and backward solves. These four functions are defined in the code as "GaussElimWithoutPivot", "BackwardSubstitution", "luFactorization" and "luSolve". These functions will be applied to a square matrix A we studied in problem 3 of the second problem set with $b = [3, 2, 1]^T$ to confirm their correctness. To further evaluate the effectiveness of these methods, we propose an experiment to solve the system Ax=b using a defined square matrix $A = (5\sqrt{n})I + R$, where R is a matrix with random entries from the normal distribution centered at 0 with a standard deviation of 1 and I is the identity matrix, for varying values of n (50, 100, 250, and 500). We measure the residual errors of Gaussian Elimination and LU Factorization and plot the time needed by each method against n. Lastly, we explore a few experiments with a matrix family $\hat{A}$ = R and discuss the observations and possible reasons behind them. Overall, this report provides insights into the efficiency of each method for solving linear equations and their practical applications.

2 Results

2.1 Factored Matrices and Solution x Obtained Using GE and LU

In this section, we present the factored matrices and solutions (x) that we obtained using Gaussian elimination without pivoting and LU factorization. We applied these scripts to the (n×n) matrix A and and the (n × 1) vector b:

$$A = \begin{bmatrix} 2 & 4 & 5 \\ 7 & 6 & 5 \\ 9 & 11 & 3 \end{bmatrix}, b = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

from problem 3 in the second problem set to confirm their correctness. For Gaussian elimination without pivoting, we obtain the upper-triangular matrix of matrix A (U) and then use backwards substitution to calculate the solution (x):

$$U = \begin{bmatrix} 2 & 4 & 5 \\ 0 & -8 & -12.5 \\ 0 & 0 & -8.5625 \end{bmatrix}, y = \begin{bmatrix} 3 \\ -8.5 \\ -5.0625 \end{bmatrix}, x = \begin{bmatrix} -0.2555 \\ 0.1387 \\ 0.5912 \end{bmatrix}$$

For LU factorization, we obtained the factored matrices of the lower-triangular matrix (L) and the upper-triangular matrix (U) and the solution (x):

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 3.5 & 1 & 0 \\ 4.5 & 0.8750 & 1 \end{bmatrix}, U = \begin{bmatrix} 2 & 4 & 5 \\ 0 & -8 & -12.5 \\ 0 & 0 & -8.5625 \end{bmatrix}, x = \begin{bmatrix} -0.2555 \\ 0.1387 \\ 0.5912 \end{bmatrix}$$

Furthermore, based on the results above, we have confirmed the correctness of the factored matrices and solutions (x) obtained from our programmed implementations by comparing them with the hand calculations performed in the second problem set. We have demonstrated that the solutions (x) obtained using both Gaussian elimination without pivoting and LU factorization are the same.

2.2 Residual Errors Table

	GE	LU
50	4.15×10^{-15}	3.75×10^{-15}
100	6.36×10^{-15}	5.79×10^{-15}
250	1.56×10^{-14}	1.41×10^{-14}
500	3.93×10^{-14}	3.44×10^{-14}

Table 1: The residual error for $\|Ax - b\|_2$ for different sized matrices GE, and LU factorization

In this section, we computed the residual error $||Ax - b|\,|_2$ to measure the accuracy of solutions obtained using both the Gaussian elimination without pivoting (GE) and LU factorization (LU) methods. We conducted an experiment to solve the system Ax = b, where A is defined as a (n×n) matrix $A = (5\sqrt{n})I + R$. R is an n × n matrix with random entries from the normal distribution with a mean of 0 and a standard deviation 1, and I is an n × n identity matrix. The vector b is a (n × 1) vector with entries drawn from the same random distribution. We used GE and LU methods to solve the system Ax = b for n values of 50, 100, 250, and 500.

Our results in Table 1 above shows that all the residual error of LU is smaller than that of GE for the same size n. Moreover, as n increases, the residual errors in both methods increase slightly, indicating that larger matrices are more challenging to solve accurately. Both methods produce relatively small residual errors close to zero, indicating accurate results for solving this system.

2.3 Time vs. Size Plot of System Solution

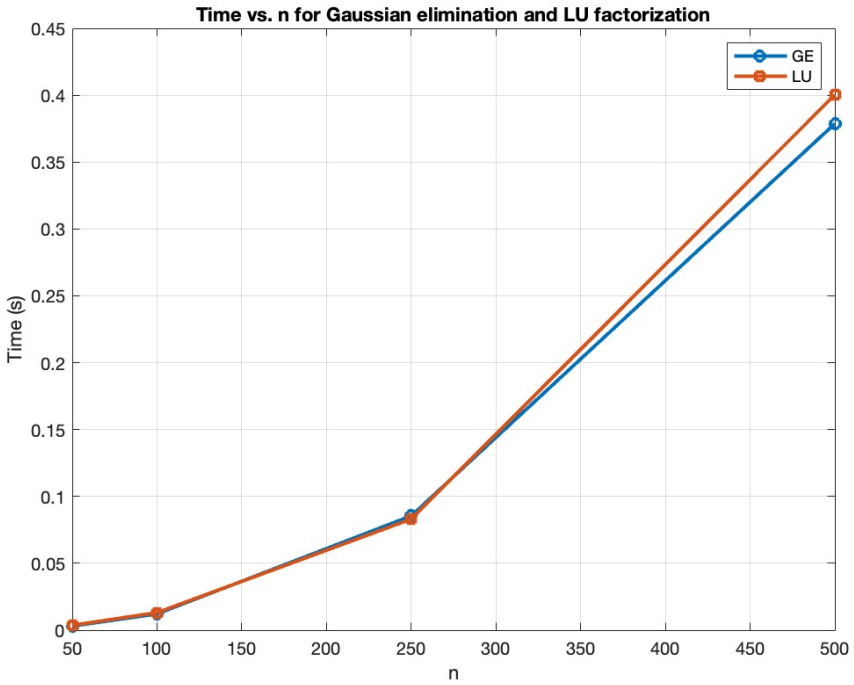

Figure 1: Plot of the time needed to solve the system (y-axis) vs. n (x-axis) for Gaussian elimination without pivoting (GE) and LU factorization (LU).

In this section of the experiment, we compared the performance of Gaussian Elimination without pivoting and LU Factorization in solving a system of linear equations with different sizes (n = 50, 100, 250, 500). We plotted the data and observed how the time needed by each method changed as the size of n increased, identifying any significant differences in their performance.

As shown in Figure 1 above, for smaller values of n, such as 50, 100, and 250, the time taken by both methods to solve the system is similar, and the lines for GE and LU are almost aligned. However, as n increases from 250 to 500, the additional computations required by LU factorization become more significant, resulting in slightly longer times compared to GE. This is because the LU factorization method involves decomposing the matrix A into an upper and lower triangular matrix using backward and forward substitutions, which requires more computations than the simple row operations used in Gaussian elimination without pivoting.

When we changed the matrix $A = (5\sqrt{n})I + R$ to the random matrix family $\hat{A} = \mathrm{R}$, we repeated the same experiment and compared the performance of GE and LU methods to solve the system Ax=b for n values of 50, 100, 250, and 500. We observed that the time needed by GE and LU takes slightly longer than the previous experiment. In this case, the time needed by GE takes slightly longer than LU. Additionally, the residual errors of both methods increased compared to the previous experiment, since the random matrix family is less structured and not guaranteed to be diagonally dominant.

3 Collaboration

Yirong Xu and Sebrina Zhu

4 Academic Integrity

On my personal integrity as a student and member of the UCD community, I have not given nor received any unauthorized assistance on this assignment.

5 Appendix

```
clc;clear; close all;
%---Gaussian elimination without pivoting-----
% matrix from PS2 problem 3
A = [2 4 5; 7 6 5; 9 11 3];
b = [3; 2; 1];
% Perform Gaussian elimination without pivoting
[U, b_transformed] = GaussElimWithoutPivot(A, b)
% Perform back substitution
x = BackwardSubstitution(U, b_transformed);
% Print solution vector x
disp(x);

%---------------LU solver--------------
% matrix from PS2 problem 3
A2 = [2 4 5; 7 6 5; 9 11 3];
b2 = [3; 2; 1];
% Perform LU factorization
[L, U] = luFactorization(A2)
% Solve system using LU factorization
x2 = luSolve(L, U, b2);
% Print solution vector
disp(x2);

% -----Define matrix A and vector b for n=50, 100, 250,
   500------
n_values = [50, 100, 250, 500];
p = length(n_values);
err_ge = zeros(p, 1);
err_lu = zeros(p, 1);
time_ge = zeros(p, 1);
time_lu = zeros(p, 1);

for i = 1:p
    n = n_values(i);
    R= randn(n,n);
    A = (5*sqrt(n))*eye(n) + R;
    %A = R;
    b = randn(n, 1);

    % Solve Ax=b by using Gaussian elimination without
       pivoting
    tic
    [uuuu,bbbb] =GaussElimWithoutPivot(A, b);
    x_gauss =BackwardSubstitution(uuuu, bbbb);
    time_ge(i) = toc;
    err_ge(i) = norm(A*x_gauss-b);
```

```
    % Solve Ax=b by using LU factorization
    tic
    [L, U] = luFactorization(A);
    y = ForwardSubstitution(L, b);
    x_lu = BackwardSubstitution(U, y);
    time_lu(i) = toc;
    err_lu(i) = norm(A*x_lu-b);
end

% ---display a table with residual errors for GE and LU------
% Print results
fprintf('n\t||Ax-b|| (GE)\t||Ax-b|| (LU)\tTime (GE)\tTime (LU
   )\n')
for i = 1:p
    fprintf('%d\t%.2e\t\t%.2e\t\t%.4f\t\t%.4f\n', n_values(i)
       , err_ge(i), err_lu(i), time_ge(i), time_lu(i))
end

% Plot time vs. n for GE and LU
figure
plot(n_values, time_ge, '-o', 'LineWidth', 2)
hold on
plot(n_values, time_lu, '-s', 'LineWidth', 2)
xlabel('n')
ylabel('Time (s)')
legend('GE', 'LU')
title('Time vs. n for Gaussian elimination and LU
   factorization')
grid on


%-----Gaussian elimination without pivoting------
% this function is on the augmented matrix A | b
% and returns the upper-triangular matrix U
% and returns the transformed right-hand side vector b
% A: the matrix of coefficients
% b: the right-hand side vector
% U: the upper triangular matrix

function [U, b] = GaussElimWithoutPivot(A, b)
    n = size(A, 1);
    U = A;
    % Gaussian elimination without pivoting
    for k = 1:n-1
        for p = k:n
            if U(p, k) ~= 0
                break;
```

```
            end
            fprintf("No unique solution exists.");
            return;
        end
        for i = k+1:n
            factor = U(i,k) / U(k,k);
            U(i,k:n) = U(i,k:n) - factor * U(k,k:n);
            b(i) = b(i) - b(k) * factor;
        end
    end
end
```

```
%--------------- Back Substitution--------------
% this function is to solve the system Ux = b for x, where
% U: the upper triangular matrix
% b: the right-hand side vector
% x: the solution vector
function x = BackwardSubstitution(U,b)

    n = size(U,1);
    if size(b, 1) ~= n
        error('Matrix dimensions are inconsistent.')
    end
    x = zeros(n,1);

    for i = n:-1:1
        x(i) = b(i);
        for j = i+1:n
            x(i) = x(i) - U(i,j)*x(j);
        end
        x(i) = x(i)/U(i,i);
    end

end
```

```
%--------------- Forward Substitution--------------
% this function is to solve the system Ly = b for y, where
% L: the lower-triangular matrix
% b: the right-hand side vector
% y: the solution vector

function x = ForwardSubstitution(L, b)
    n = size(L,1);
    x = zeros(n,1);
```

```
    x(1,1) = b(1,1)/L(1,1);
    for i = 2:n
        sum = 0;
        for j = 1:i-1
            sum = sum + L(i,j)*x(j,1);
        end
        x(i,1) = (b(i,1)-sum)/L(i,i);
    end
end

%--------LU factorization--------------------
function [L, U] = luFactorization(A)
% this function is to solve the LU factorization of a nxn
   matrix A
% A: the nxn matrix
% L: the lower-triangular matrix
% U: the upper-triangular matrix
    n = size(A,1);
    % Initialize L and U matrices
    L = eye(n);
    U = A;
    % Gaussian elimination without pivoting
    for k = 1:n-1
        if U(k,k) == 0
           disp('Factorization impossible');
        end
        for i = k+1:n
            factor = U(i,k) / U(k,k);
            L(i,k) = factor;
            U(i,k:n) = U(i,k:n) - factor * U(k,k:n);
        end
    end
end

%----------------luSolver----------------
function x2 = luSolve(L, U, b)
% this function is to solve the system LUx = b for x, where
% L: the lower-triangular matrix
% U: the upper-triangular matrix
% b: the right-hand side vector
% x: the solution vector
    y = ForwardSubstitution(L, b);
    x2 = BackwardSubstitution(U, y);
end
```