KUSHAL AGARWAL
2201mc22_kushal@iitp.ac.in
6287173664
21-01-24

LAB 1 (TASK 1)

```
% Task 1

clc, clearvars;

n = 500:50:1000; % For the matrix size as mentioned in ques.

j = 1;

time1 = zeros(10, 1);

time2 = zeros(10, 1);

val1 = zeros(10, 1);

filename = 'L1Q1.xlsx';

% Starting loop (10 Times)

for i = 1:10

   %creating a random-sized matrix A & B using rand fn

   A = rand(n(1, j), n(1, j));

   b = rand(n(1, j), 1);

   % To measure the time in the operation using function

   tic

   x = A\b;

   y1 = toc; % Stopwatch stops, and the value is noted

   % Creating an array that will store all the data

   time1(i) = y1;

   val1(i) = n(1, j);

   % Write val1 and time1 to columns A and B

   writematrix([val1, time1], filename, 'Sheet', 'Sheet1', 'Range', 'A1');
```

```
    tic
    x1 = inv(A)*b;
    y2 = toc;
    time2(i) = y2;
    % Write time2 to column C
    writematrix(time2, filename, 'Sheet', 'Sheet1', 'Range', 'C1');
    j = j + 1;
end
% Plotting the graph
temp=readmatrix(filename);
x_1 = temp(:, 1);
y_1 = temp(:, 2);
z_1 = temp(:, 3);
plot(x_1,y_1);
hold on
plot(x_1,z_1);
hold off
legend('A\B','inv(A)*B')
```

LAB 1 (TASK 2)

```
% Task 2

clc , clearvars;

% To solve this question we are creating a function FdSubs which is taking

% taking two input L and b and returning an ouput matrix x

% Manual Input to test the working of code

L=[1,0,0;2,1,0;3,2,1];

b=[1;4;7];

disp('L:');

disp(L);

disp('b:');

disp(b);

A = FdSub(L,b);

disp('Solution:');

disp(A); % Display answer

function [x] = FdSub(L, b)

[m, n] = size(L);

if m ~= n

   error('L must be a square matrix');  % Error

end

x = zeros(n,1); % If L is square matrix, we create a column matrix with [0]

for i = 1:n

   x(i) = (b(i) - L(i, 1:n) * x) / L(i, i); % Main equation

end

end
```

LAB 1 (TASK 3)

```
% Task 3

clc , clearvars;

% To solve this question, we are creating a function BdSubs, which is

% taking two inputs, U and b, and returning an output matrix x

% Manual Input to test the working of code

U=[1,0,1;0,2,0;0,0,2];

b=[1;2;0];

disp('U:');

disp(U);

disp('b:');

disp(b);

A = BdSub(U,b);

disp('Solution:');

disp(A);  % Display answer

% Dunction for backward subs

function [x] = BdSub(U, b)

[m, n] = size(U);

if m ~= n

   error('U must be a square matrix'); % Error

end

x = zeros(n,1);

for i = 1:n

   j = n+1-i;

   % We can observe back subs when we run the for loop and
```

```
    % put the required values in it
    % Here we are creating a variable j and first we are calculating the value
    % nth row of x...first we calculate the nth element of x then using
    % that n-1th element of x and so on...
    x(j) = (b(j) - U(j, 1:n) * x) / U(j, j); % Main Equation
end
end
```

LAB 1 (TASK 4)

```
% Task 4

clc, clearvars;

% Creating variables as mentioned in the ques

p=8;

x = [2 -1 zeros(1, p-2)];

A = toeplitz(x); % Creating a toeplitz matrix

[m ,n]=size(A);

B=ones(1,8);

% Creating another row matrix B in which all elements are 1

% Algorithm to calculate L and U

L=zeros(m,m);

U=zeros(m,m);

for i=1:m

   for k=1:i-1

       L(i,k)=A(i,k);

       for j=1:k-1

           L(i,k)= L(i,k)-L(i,j)*U(j,k);

       end

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

   end

   for k=i:m

       U(i,k) = A(i,k);

       for j=1:i-1

           U(i,k)= U(i,k)-L(i,j)*U(j,k);
```

```
        end
    end
end
for i=1:m
    L(i,i)=1;
end
%Now we are using FdSubs and BdSubs functions
Y=FdSub(L, B);
X=BdSub(U, Y);
disp('x=');
disp(x);
disp('A=');
disp(A);
disp('L=');
disp(L);
disp('U=');
disp(U);
disp('X=');
disp(X);
% Fdsubs function
function [x] = FdSub(L, b)
[m, n] = size(L);
if m ~= n
    error('L must be a square matrix');
end
x = zeros(n,1);
for i = 1:n
```

```
   x(i) = (b(i) - L(i, 1:n) * x) / L(i, i);
end
end
% Bdsubs function
function [x] = BdSub(U, b)
[m, n] = size(U);
if m ~= n
   error('U must be a square matrix');
end
x = zeros(n,1);
for i = 1:n
   j = n+1-i;
   x(j) = (b(j) - U(j, 1:n) * x) / U(j, j);
end
end
```