Table of Contents


```
clc
clear
```

Variables

4x4 Matrix

```
Y = [3 1 2 0; 1 7 2 4; 2 2 4 0; 0 4 0 4];
j = [6.2 14.6 8.4 8.0].';
```

Calculations

```
[L, U] = crout_factorization(Y);
result_matrix = L * U;  % To compare with Y
eta = forward_substitution(L, j);
v = backward_substitution(U, eta);
proof = result_matrix*v;  % Should be the same as j
proof
j
```

Functions

```
function [L, U] = crout_factorization(A)
    % Preallocate Matrix size
    [N, ~] = size(A);
    L = zeros(size(A));
    U = zeros(size(A));

    L(:,1) = A(:, 1);  % Fill first column of L
    U(1,:) = A(1, :) / L(1, 1);  % Fill first row of U
    for l = 1:N
        U(l,l) = 1;  % Fill diagonal U elements
    end

    for j = 2:N  % Fill non-zero elements per row/column
        for i = j:N  % Fill L elements vertically
            L(i, j) = A(i, j) - dot(L(i, 1:j-1), U(1:j-1, j));
        end
        for i = j+1:N  % Fill U elements horizontally
            U(j, i) = (A(j, i) - dot(L(j, 1:j-1), U(1:j-1, i))) ...
                        / L(j, j);
        end
```

```
    end
end

function eta = forward_substitution(L, j)
    [N, ~] = size(L);
    eta = j;  % replace output with j

    for i = 1:N  % Loop forwards
        eta(i) = eta(i) / L(i, i);
        for k = i+1:N
            eta(k) = eta(k) - L(k, i) * eta(i);
        end
    end
end

function v = backward_substitution(U, eta)
    [N, ~] = size(U);
    v = eta;  % replace output with eta

    for i = N-1:-1:1  % Loop backwards
        for k = i+1:N
            v(i) = v(i) - U(i, k) * v(k);
        end
    end
end
```

```
proof =

    6.2000
   14.6000
    8.4000
    8.0000


j =

    6.2000
   14.6000
    8.4000
    8.0000
```

Published with MATLAB® R2019a