```matlab
% Q1

A = [1 1 1; 2 1 3; 3 4 -2];
b =[4; 7; 9];

% a.
x = solutionofLinearEquations(A, b);
disp("Solution using Gauss Elemination:");
disp(x);

% b.
x = solutionofLinearEquations(A, b, 2);
disp("Solution using LU Decomposition:");
disp(x);

% c.
x = solutionofLinearEquations(A, b, 3);
disp("Solution using Gauss elimination + partial pivoting:");
disp(x);


% ------FUNCTION DECLARATIONS--------
function fval = solutionofLinearEquations(a, b, choice)
    if ~exist('choice', 'var')
        % third parameter does not exist, so default it to something
        choice = 1;
        % By Default method is Gauss Elemination
    end
    switch choice
        case 1
            fval = GaussElemination(a, b);
        case 2
            fval = LUdecomposition(a,b);
        case 3
            fval = partialpivoting(a, b);
    end
end
function fval = GaussElemination(A, b)
    % get augumented matrix
    Ab = [A, b];
    % row Operation
    % Rj = Rj - k(i, j)*Ri where k(i, j) = A(j, i) / A(i, i)
    n = length(A);
    % A(1, 1) as pivot element
    for i = 2:n
        k = Ab(i, 1) / Ab(1, 1);
        Ab(i, :) = Ab(i, :) - k*Ab(1, :);
    end
    % A(2, 2) as pivot element
    i = n;
    k = Ab(i, 2) / Ab(2, 2);
    Ab(i, :)= Ab(i, :) - k*Ab(2, :);
```

```
    % A(3, 3) as pivot element
    % Back-Subsituation
    fval = zeros(n, 1);
    % x(3) = Ab(3, 4) / Ab(3, 3);
    for i = n : - 1 : 1
        % x(2) = (Ab(2, 4) - Ab(2, 3)*x(3)) / Ab(2, 2);
        fval(i) = (Ab(i, end) - Ab(i, i + 1 : n)*fval(i + 1 : n)) /
 Ab(i, i);
        % x(1) = (Ab(1, 4) - (Ab(1, 3)*x(3) + Ab(1, 2)*x(2))) / Ab(1,
 1);
        % x(1) = (Ab(1, 4) - (Ab(1, 1 + 1 : n)*x(1 + 1 : n)) / Ab(1,
 1);
    end
    end
    function fval = LUdecomposition(A, b)
    % get augumented matrix
    Ab = [A, b];
    n = length(A);
    L = eye(n);
    % Row Operation
    % Rj = Rj - k(i, j)*Ri where k(i, j) = A(j, i) / A(i, i)
    % A(1, 1) as pivot element
    for i = 2 : n
        k = Ab(i, 1) / Ab(1, 1);
        L(i, 1) = k;
        Ab(i, :)= Ab(i, :) - k*Ab(1, :);
    end
    % A(2, 2) as pivot element
    i = n;
    k = Ab(i, 2) / Ab(2, 2);
    L(i, 2) = k;
    Ab(i, :) = Ab(i, :) - k*Ab(2, :);
    % A(3, 3) as pivot element
    U = Ab(1 : n, 1 : n);
    y = inv(L)*b;
    fval = inv(U)*y ;
end
function fval = partialpivoting(A, b)
    % get augumented matrix
    Ab = [A, b];
    n = length(A);
    % A(1, 1) as pivot element
    % Ensure A(1, 1) is largest element in column-1
    col1 = Ab(:, 1);
    [dummy, idx] = max(col1);
    dummy = Ab(1, :);
    Ab(1, :) = Ab(idx, :);
    Ab(idx, :) = dummy;
    for i = 2 : n
        k = Ab(i, 1) / Ab(1, 1);
        Ab(i, :) = Ab(i, :) - k*Ab(1, :);
    end
    % A(2, 2) as pivot element
    % Ensure A(2, 2) is largest element in column-2
```

```
    col2 = Ab(2 : end, 2);
    [dummy, idx] = max(col2);
    dummy = Ab(2, :);
    Ab(2, :) = Ab(idx, :);
    Ab(idx, :) = dummy;
    i = 3;
    k = Ab(i, 2) / Ab(2, 2);
    Ab(i, :) = Ab(i, :) - k*Ab(2, :);
    % A(3, 3) as pivot element
    % Back-Subsituation
    fval = zeros(n, 1);
    % x(3) = Ab(3, 4) / Ab(3, 3);
    for i = n : -1 : 1
        fval(i)= (Ab(i, end) - Ab(i, i + 1 : n)*fval(i + 1 : n)) /
 Ab(i, i);
    end
end
```

```
Solution using Gauss Elemination:
     1
     2
     1

Solution using LU Decomposition:
     1
     2
     1

Solution using Gauss elimination + partial pivoting:
    1.0000
    2.0000
    1.0000
```

Published with MATLAB® R2020b