```
% Q2

A = [1 2 2 1; 2 2 4 2; 1 3 2 5; 2 6 5 8];
b =[1; 0; 2; 4];

% a.

x = IterativemethodofLS(A, b, 1, 1e-3, 100);
disp("Solution using Gauss Seidel Method :");
disp(x);

% b.

x = IterativemethodofLS(A, b, 2, 1e-3, 100);
disp("Solution using Jacobi Method :");
disp(x);

% ------FUNCTION DECLARATIONS--------

function fval= IterativemethodofLS(a,b,choice,tol,maxItr)
    switch choice
        case 1
            fval = gaussSeidel(a, b, tol, maxItr);
        case 2
            fval = Jacobi(a, b, tol, maxItr);
    end
end
function sol= gaussSeidel(A, b, tol,maxitr)
    % Here co-efficient matrix A must be 'strictly diagonally dominant
 matrix'
    % tol is maximum bearable tolerance in answer
    % maxitr is limit of iterations
    n = length(A);
    Xnext = zeros(n, 1); % assuming initial approximation as zero
 vector
    for loop = 1 : maxitr
        Xcurr = Xnext;
        for i = 1 : n
            temp = 0;
            for j = 1 : n
                if(i ~= j)
                    temp = temp + (A(i, j)*Xnext(j));
                end
            end
            Xnext(i) = (b(i) - temp) / A(i, i);
        end
        error = Xnext - Xcurr;
        err = norm(error);
        if err <= tol
            sol = Xnext;
            break;
        end
```

```
    end
    sol=Xnext;
end
function fval= Jacobi(A,b,tol,maxitr)
    % Here co-efficient matrix A must be 'strictly diagonally dominant
 matrix'
    %tol is maximum bearable tolerance in answer
    % maxitr is limit of iterations
    n = length(A);
    Xcurr = zeros(n, 1); % assuming initial approximation as zero
 vector
    Xnext = zeros(n, 1);
    for loop = 1 : maxitr
        for i = 1 : n
            temp = 0;
            for j = 1 : n
                if(i ~= j)
                    temp = temp + (A(i, j)*Xcurr(j)); % This loop
 calculates #k#j a(k,j)*x(j)
                end
            end
            Xnext(i)=(b(i)-temp)/A(i,i);
        end
        error=Xnext-Xcurr;
        err=norm(error);
        if err<=tol
            fval=Xnext;
            break;
        end
        Xcurr=Xnext;
    end
    fval=Xnext;
end
```

```
Solution using Gauss Seidel Method :
   1.0e+30 *

   -5.0703
   -1.2677
    7.6056
   -2.5352

Solution using Jacobi Method :
   1.0e+54 *

   -1.1841
   -0.9701
   -0.8970
   -0.4485
```

Published with MATLAB® R2020b