$$\begin{cases} \min\ 2x_1^2 + x_1x_3 + x_2^2 + 2x_2x_3 + 3x_3^2 + x_3x_4 + 2x_4^2 - 5x_1 - 4x_3 + 3x_4 \\ x \in \mathbb{R}^4 \end{cases}$$

```
clear;clc; close all;
```

a) Do global optimal solutions exist? Why?

```
% Form of the objective function is quadratic function
% Calculate the Hessian. If Hessian is positive definite or positive semidefinite

H = hessian_matrix();

f = [-5
    0
    -4
    3];

eigenvalue = eig(H);
display(eigenvalue)
```

```
eigenvalue = 4×1
    1.0608
    3.5933
    4.0000
    7.3460
```

```

% As result we have a positive definite objective function
% Means Global Optimum exist and is unique
```

b) Is it a convex problem? Why?

```
% It is strictly convex

x = quadprog(H,f);
```

```
Minimum found that satisfies the constraints.

Optimization completed because the objective function is non-decreasing in
feasible directions, to within the value of the optimality tolerance,
and constraints are satisfied to within the value of the constraint tolerance.

<stopping criteria details>
```

```
display(x);
```

```
x = 4×1
    1.0000
   -1.0000
    1.0000
```

```
   -1.0000
```

```
x0 = [0
      0
      0
      0];
alpha = 0.1;
gamma = 0.9;
t_bar = 1;
tolerance = 1e-6;


% Gradient Method with line search
x_gradient_method = gradient_method(x0, tolerance,H);
```

```
Gradient method with exact line search

Iteration      f(x)          ||grad f(x)||

 0      0.0000000000000      7.0711e+00
 1      -5.0403225806452      2.4686e+00
 2      -5.5976694221960      1.1368e+00
 3      -5.7883000924345      9.8085e-01
 4      -5.8850831814683      6.0051e-01
 5      -5.9375039953765      5.3213e-01
 6      -5.9660079447169      3.2658e-01
 7      -5.9815113436573      2.8943e-01
 8      -5.9899438100569      1.7763e-01
 9      -5.9945303240821      1.5742e-01
10      -5.9970249811449      9.6616e-02
11      -5.9983818534549      8.5624e-02
12      -5.9991198717155      5.2550e-02
13      -5.9995212882297      4.6572e-02
14      -5.9997396232310      2.8583e-02
15      -5.9998583781180      2.5331e-02
16      -5.9999229702499      1.5546e-02
17      -5.9999581026440      1.3778e-02
18      -5.9999772115522      8.4559e-03
19      -5.9999876051044      7.4939e-03
20      -5.9999932582755      4.5993e-03
21      -5.9999963330995      4.0760e-03
22      -5.9999980055312      2.5016e-03
23      -5.9999989151857      2.2170e-03
24      -5.9999994099572      1.3606e-03
25      -5.9999996790690      1.2058e-03
26      -5.9999998254420      7.4007e-04
27      -5.9999999050559      6.5587e-04
28      -5.9999999483588      4.0253e-04
29      -5.9999999719118      3.5674e-04
30      -5.9999999847225      2.1894e-04
31      -5.9999999916904      1.9403e-04
32      -5.9999999954803      1.1908e-04
33      -5.9999999975417      1.0554e-04
34      -5.9999999986629      6.4772e-05
35      -5.9999999992727      5.7403e-05
36      -5.9999999996044      3.5230e-05
37      -5.9999999997848      3.1222e-05
38      -5.9999999998830      1.9162e-05
39      -5.9999999999363      1.6982e-05
```

```
40      -5.9999999999654      1.0422e-05
41      -5.9999999999812      9.2367e-06
42      -5.9999999999898      5.6689e-06
43      -5.9999999999944      5.0239e-06
44      -5.9999999999970      3.0834e-06
45      -5.9999999999984      2.7326e-06
46      -5.9999999999991      1.6771e-06
47      -5.9999999999995      1.4863e-06
48      -5.9999999999997      9.1218e-07
```

```
% Conjugate Gradient Method
x_conjugate_gradient = conjugate_method(x0, tolerance, H);
```

```
Conjugate Gradient method

Iteration      f(x)           ||grad f(x)||

 0      0.0000000000000      7.0711e+00
 1      -5.0403225806452      2.4686e+00
 2      -5.6669616780246      9.2779e-01
 3      -5.9998774890506      2.9667e-02
 4      -6.0000000000000      2.8436e-15
```

```
% Newton Method with Armijo line search
x_newton_method = newton_method(x0, alpha, gamma, t_bar, tolerance);
```

```
Newton Method

Iterations      f(x)           ||grad f(x)||
 0      0.0000000000000      7.0711e+00
 1      -6.0000000000000      0.0000e+00
```

c) Find the global minimum by using the gradient method with exact line search starting from the point $(0, 0, 0, 0)$ [Use $\|\nabla f(x)\| < 10^{-6}$ as stopping criterion]. How many iterations are needed?

```
function alpha = line_search(g, d, H)
    alpha = (-g'*d)/(d'*H*d);
end

function x_opt = gradient_method(x0, tolerance, H)

    fprintf('Gradient method with exact line search\n\n');
    fprintf('Iteration \t f(x) \t\t ||grad f(x)||\n\n');

    k = 0 ;
    % starting point
    x = x0;
    while true
        [f, g, H] = f_(x);

        fprintf('%2.0f \t %2.13f \t %1.4e\n',k,f,norm(g));
```

```
        % stopping criterion
        if norm(g) < tolerance
            break
        end

        %   search direction
        d = -g;

        %   step size
        alpha = line_search(g, d, H);

        %   new point
        x = x + alpha*d ;
        k = k + 1 ;
    end
    x_opt = x;
end
```

d) Find the global minimum by using the conjugate gradient method starting from the point $(0, 0, 0, 0)$. How many iterations are needed? Write the point found by the method at any iteration.

```
function beta = conjugate_function(g,d,H)
    beta = (g'*H*d)/(d'*H*d);
end
% x0: Initial point
% d: search direction
function x_opt = conjugate_method(x0, tolerance, H)
    fprintf('Conjugate Gradient method \n\n');
    fprintf('Iteration \t f(x) \t\t ||grad f(x)||\n\n');

    x = x0;
    k = 0;
    g = gradient_function(x);
    while true
        f = objective_function(x);
        fprintf('%2.0f \t %2.13f \t %1.4e\n',k,f,norm(g));

        % Stop Criteria
        if norm(g) < tolerance
            break;
        end

        if k == 0
            d = -g;
        else
            % conjugate function
            beta = conjugate_function(g,d,H);
            d = -g + beta * d;
```

```
        end

        % step size
        alpha = line_search(g,d,H);

        % new point
        x = x + alpha .* d;

        g = gradient_function(x);
        k = k+1;

    end
    x_opt = x;
end
```

e) Find the global minimum by using the Newton method starting from the point $(0, 0, 0, 0)$. How many iterations are needed?

```
function x_opt = newton_method(x0,alpha,gamma,t_bar, tolerance)
    fprintf("Newton Method\n\n");
    fprintf("Iterations \t f(x) \t\t ||grad f(x)||\n");
    k = 0;
    x = x0;
    while true
        [f, g, H] = f_(x);

        fprintf("%2.0f \t %2.13f \t %1.4e\n", k,f,norm(g));

        % Stop Criteria
        if norm(g) < tolerance
            break;
        end

        % Search direction H*d = -g
        % Example Ax=b Linear equation  x=A\b
        d = -H\g;

        % Step size with Armijo line search
        t = t_bar;

        while f_(x+t*d)> f + alpha*t*d'*g
            t = gamma*t;
        end
        x = x + t*d;
        k = k+1;
    end
    x_opt = x;
end
```

```
function [f,g,H] = f_(x)
    f = objective_function(x);
    g = gradient_function(x);
    H = hessian_matrix();
end

function H = hessian_matrix()
    H = [4 0 1 0
         0 2 2 0
         1 2 6 1
         0 0 1 4];
end

function f = objective_function(x)
% x: Input vector
% f: output scalar value of the objective function
    x1 = x(1);
    x2 = x(2);
    x3 = x(3);
    x4 = x(4);

    f = 2*x1^2+x1*x3+x2^2+2*x2*x3+3*x3^2+x3*x4+2*x4^2-5*x1-4*x3+3*x4;
end

function g = gradient_function(x)
    x1 = x(1);
    x2 = x(2);
    x3 = x(3);
    x4 = x(4);

    g = [4*x1+x3-5
        2*x2+2*x3
        x1+2*x2+6*x3+x4-4
        x3+4*x4+3];
end
```