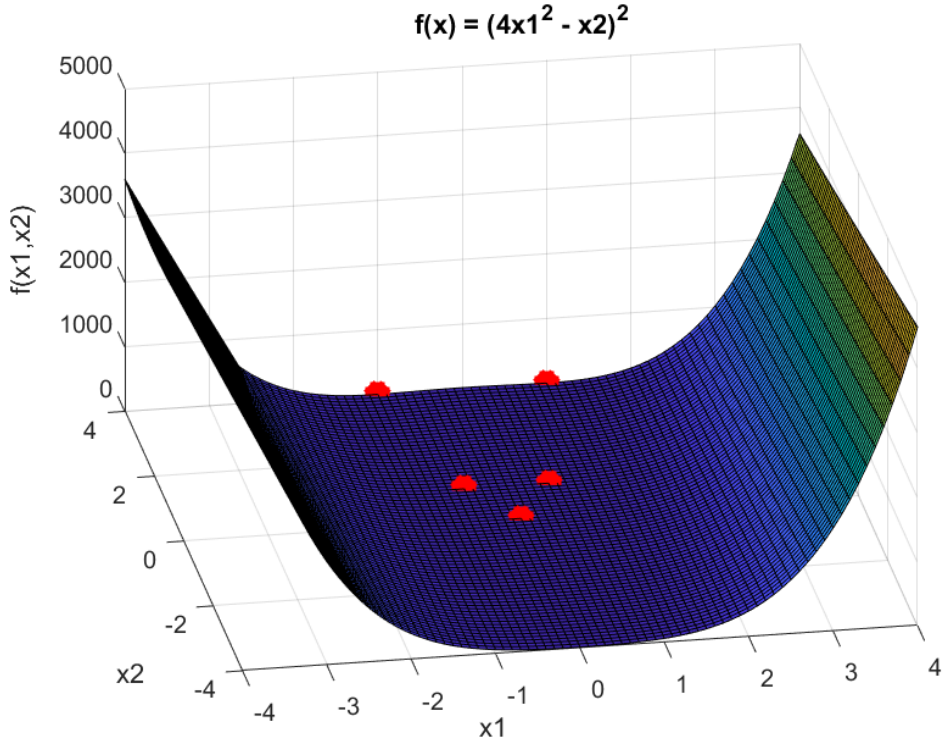
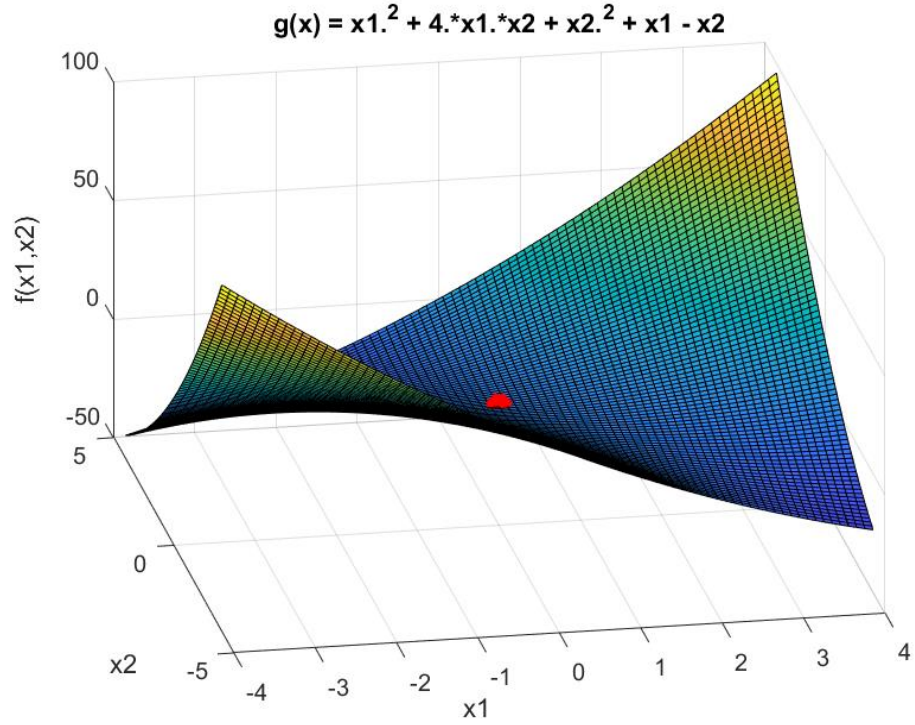
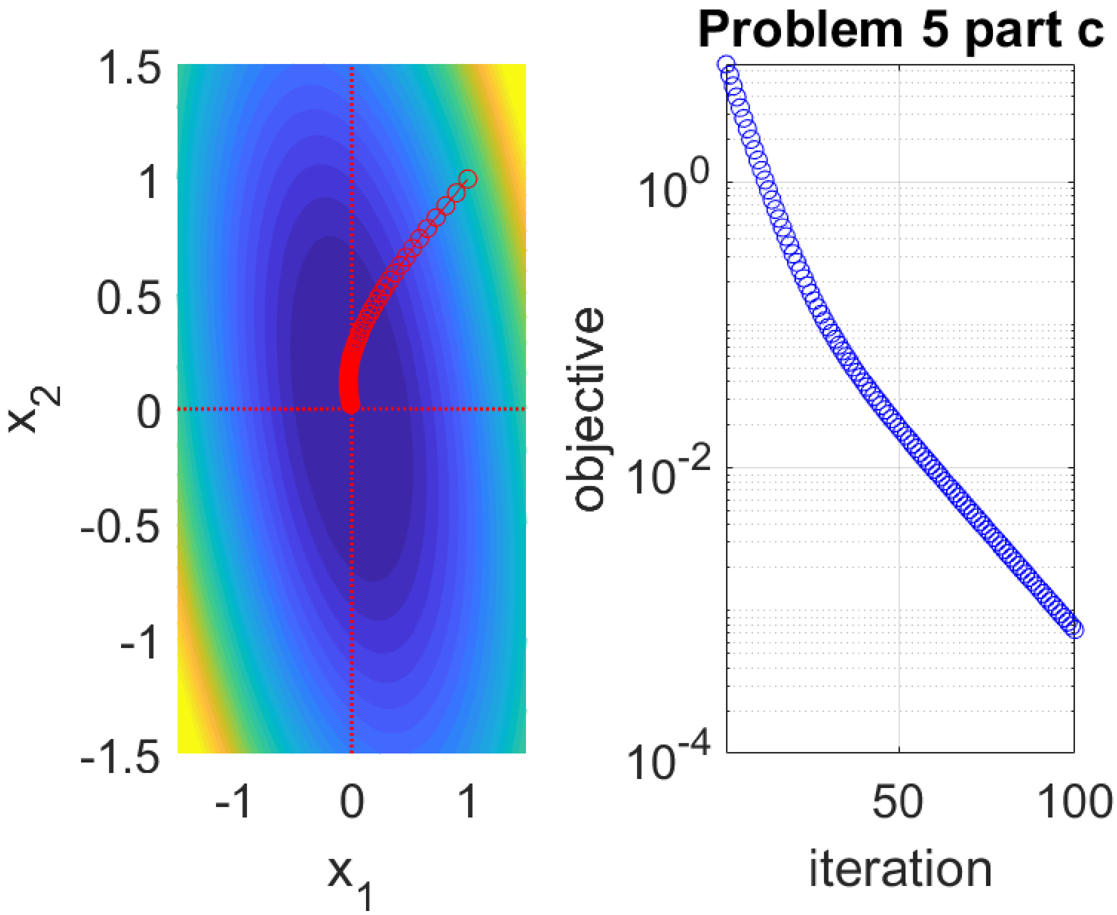
The points on the f(x) graph represent the parabolic function x2 = 4x1^2.





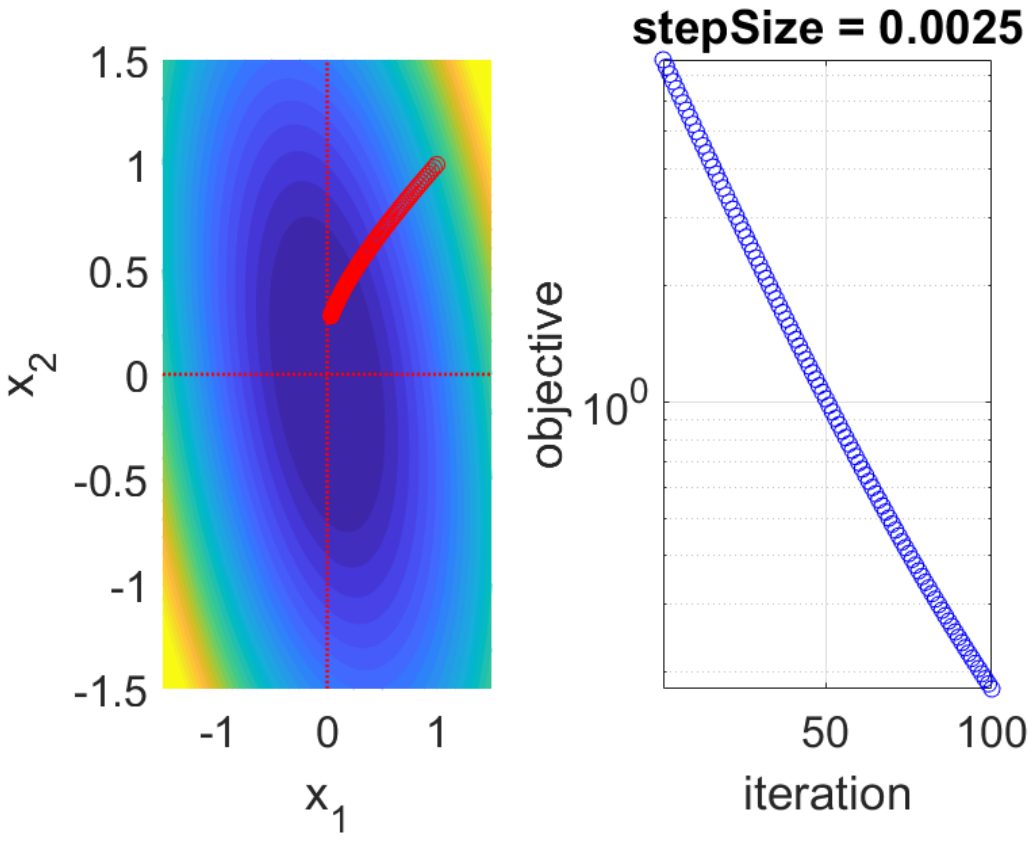


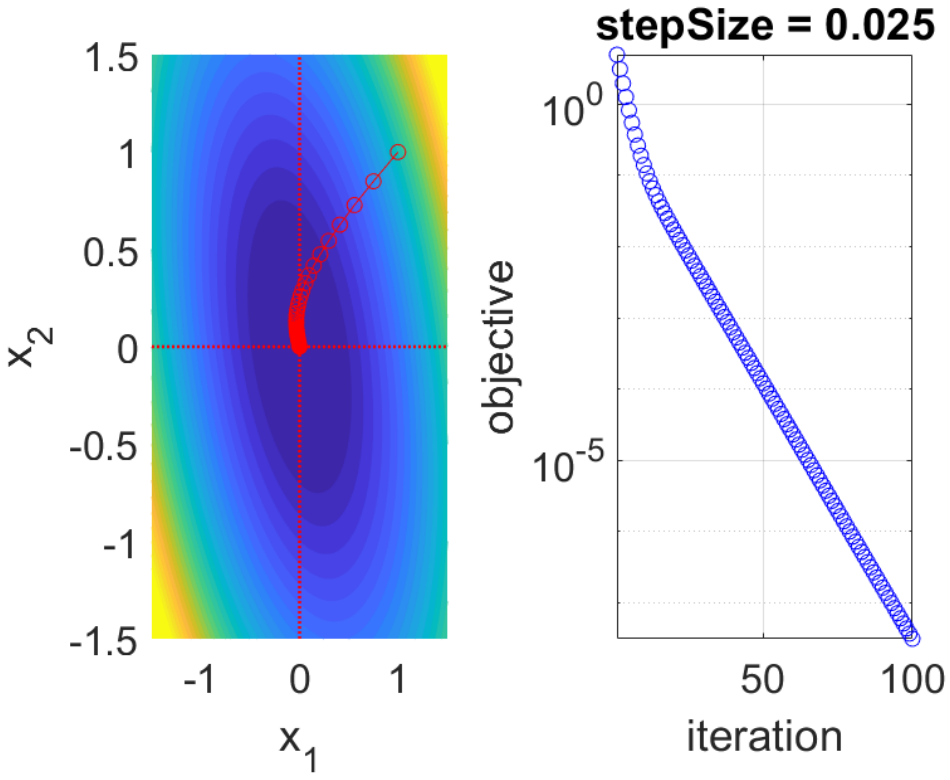

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%
% This script illustrates the gradient method (GM)
for
% solving unconstrained optimization problems.
%
% This script uses "column vector" layout for
vectors
% (as adopted in the lectures).
%
% U. S. Kamilov, CIG, WUSTL, 2020.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%% prepare workspace
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
```

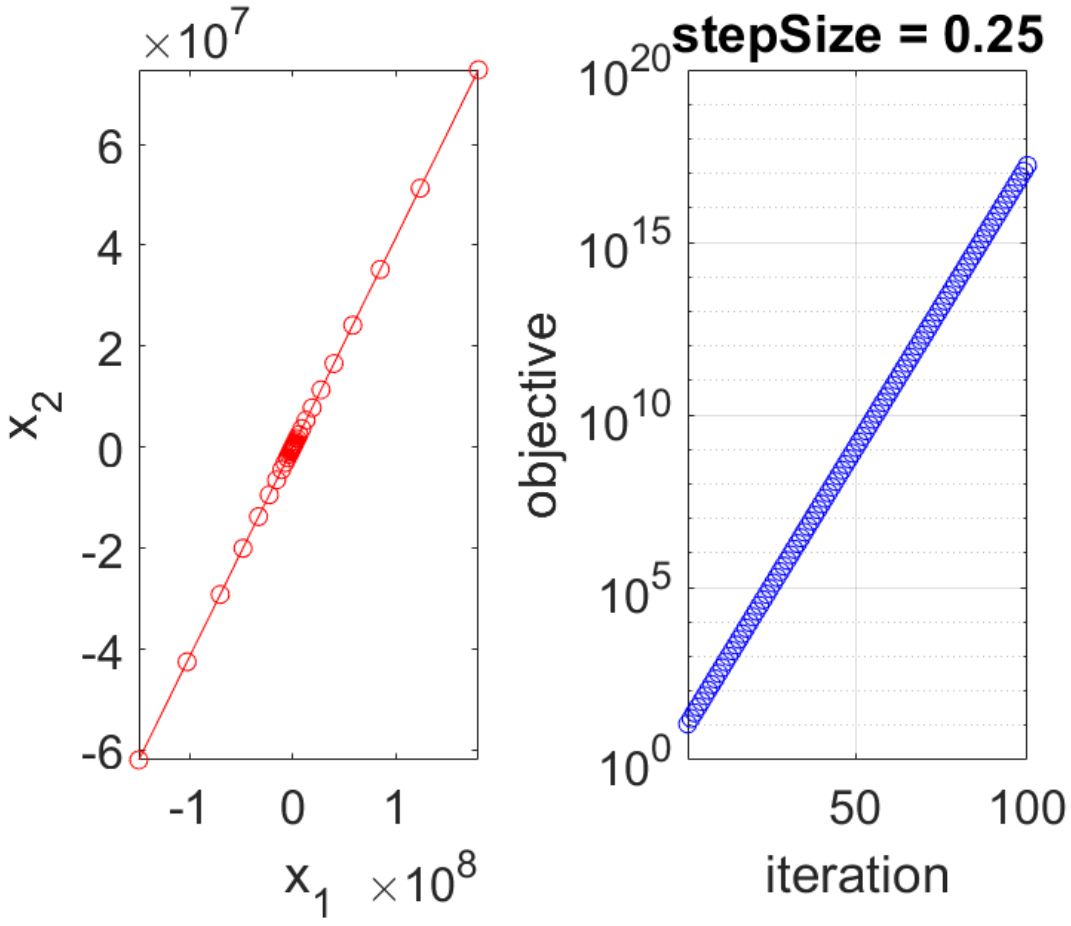
```
clear; close all; home;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%% define the function and the gradient (*** to be
completed ***)(done)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
evaluateFunc = @(x1, x2) 4*x1.^2 + 2*x1.*x2 +
2*x2.^2;
evaluateGrad = @(x1, x2) [8*x1 + 2*x2; 2*x1 +
4*x2];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%% plot the contours of the function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
figure('Color', 'w');
subplot(1, 2, 1);
fcontour(evaluateFunc, [-1.5, 1.5], 'Fill', 'on',
'LevelList', 0:0.5:15);
hold on;
plot([-1.5, 1.5], [0, 0], 'r:', 'LineWidth', 1.2)
plot([0, 0], [-1.5, 1.5], 'r:', 'LineWidth', 1.2)
xlabel('x_1');
ylabel('x_2');
set(gca, 'FontSize', 16);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%% parameters of the gradient method
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
xitInit = [1; 1]; % initialization
stepSize = 0.01; % step size
tol = 1e-6; % stopping tolerance
maxIter = 100; % maximum number of iterations
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
%% optimize
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%
% initialize
xit = xitInit;
% iterate
for iter = 1:maxIter

    % compute the next iterate (*** to be completed
***)(Done)
    xitNext = xit -
stepSize.*evaluateGrad(xit(1),xit(2));

    % plot the objective and the iterate evolution
    subplot(1, 2, 1);
    plot([xit(1), xitNext(1)], [xit(2),
xitNext(2)], 'ro-');
    hold on;
    subplot(1, 2, 2);
    semilogy(iter, evaluateFunc(xitNext(1),
xitNext(2)), 'bo');
    hold on;
    grid on;
    xlabel('iteration');
    ylabel('objective');
    xlim([1 maxIter]);
    set(gca, 'FontSize', 16);
    drawnow;

    % check termination tolerance
    if(norm(evaluateGrad(xit(1), xit(2))) < tol)
        break;
    end
    % update the iterate
    xit = xitNext;
end
```

Problem 5 Part d.






×10^7
6
4
2
0
-2
-4
-6
x_2
-1
0
1
x_1
×10^8
stepSize = 0.25
10^20
10^15
10^10
10^5
10^0
objective
50
100
iteration