```matlab
% Solve the problem from Boyd and Vandenberghe using gradient descent
% and Newton's method.
%
% Author: Terrence Alsup
% Date: March 10, 2020
% File: BV_problem.m

% Load the data and get the dimensions.
data = open('Adata.mat');
A = data.A;
[n, m] = size(A);

% Set up the objective function.
fun = @(x) objective_fun(x, A);


% Set the parameters for gradient descent and run.
x0 = zeros(n, 1);
tol = 1e-6;
maxiter = 100;
[f_gd, gnorm_gd] = gradmeth(fun, x0, tol, maxiter);

% f_all is a row vector, so the last entry is the number of columns.
gd_iter = size(f_gd, 2) - 1;
fprintf('Gradient descent took %d iterations\n', gd_iter);
fprintf('Gradient descent optimal value p^* = %f\n', f_gd(gd_iter +
 1));

% Estimate the condition number M/m.
f_num = f_gd(2:gd_iter) - f_gd(gd_iter + 1);
f_den = f_gd(1) - f_gd(gd_iter + 1);
logc = max(log(f_num/f_den)./(1:gd_iter-1));
cond_num = 1/(1 - exp(logc));
fprintf('Estimated condition number = %f\n\n', cond_num);


% Set the parameters for Newton's method and run.
x0 = zeros(n, 1);
tol = 1e-8;
maxiter = 100;
[f_newt, gnorm_newt, m, M, L] = newtmeth(fun, x0, tol, maxiter);

% Print the results.
newt_iter = size(f_newt, 2) - 1;
fprintf('Newton method took %d iterations\n', newt_iter);
fprintf('Newton method optimal value p^* = %f\n', f_newt(newt_iter +
 1));

% Compute the theoretical number of iterations.
eta = min([1, 3*(1 - 2*0.25)]) * m^2 / L;
gamma = 0.25 * 0.5 * (eta^2) * m / M;
```

```
theor_it = (f_newt(1) - f_newt(newt_iter + 1))/gamma +
 log2(log2(2*m^3/(tol*L^2)));
fprintf('Theoretical upper bound on Newton iterations = %d\n',
 ceil(theor_it));



% Plot all the results.
figure(1);
semilogy(0:gd_iter-1, f_gd(1:gd_iter) - f_gd(gd_iter + 1), '-+');
hold on
semilogy(0:newt_iter-1, f_newt(1:newt_iter) - f_newt(newt_iter +
 1), '-o');
xlabel('Iteration $k$', 'interpreter', 'latex');
ylabel('Error $|f(x^{(k)}) - p^*|$', 'interpreter', 'latex');
title('Convergence of gradient descent and Newtons
 method', 'interpreter', 'latex');
legend('Gradient descent', 'Newton');
hold off


figure(2);
semilogy(0:gd_iter, gnorm_gd, '-+');
hold on
semilogy(0:newt_iter, gnorm_newt, '-o');
xlabel('Iteration $k$', 'interpreter', 'latex');
ylabel('Norm of gradient $\|\nabla f(x^{(k)})\|
_2$', 'interpreter', 'latex');
title('Convergence of gradient descent and Newtons
 method', 'interpreter', 'latex');
legend('Gradient descent', 'Newton');
hold off
```

Published with MATLAB® R2019b