```
% Demo code of Newton's method
% Xiaojing Ye (xye@gsu.edu)

% set up a problem and choose zero as initial guess
% [f, Df, Hf, x_dim] = test_prob1();
% x_init = 0;

% set up a problem and choose zero as initial guess
[f, Df, Hf, x_dim] = test_prob2();
x_init = zeros(x_dim,1);

% termination parameters for Newton's method
max_iter = 100;     % for stopping criterion using maximum iteration number
relchg_tol = 1e-5;  % for stopping criterion using relative change

% run Newton's method
[x_opt, f_opt] = newton_minf(f, Df, Hf, x_init, max_iter, relchg_tol);

% print final output x in command window
fprintf('Final output x: %s. \n', sprintf('%f ', x_opt));
fprintf('Final output f: %f. \n', f_opt);
```

```
function [x_opt, f_opt] = newton_minf(f, Df, Hf, x_init, max_iter, relchg_tol)
% Newton's method to solve minimize f(x), where f(x) is twice
% differentiable with gradient Df and Hessian Hf
%
% INPUT:
%       Df: gradient of f
%       Hf: Hessian of f
%       x_init: initial guess
%       max_iter: maximum iteration number as a termination criterion (TC)
%       relchg_tol: tolerance of relative change in iterates as TC
%
% OUTPUT:
%       x_opt: approximate optimal solution x* of minimize f(x)
%       f_opt: approximate optimal function value f(x*)

% iteration counter
iter = 1;

% set current iterate as initial guess
x_curr = x_init;

% start main loop
while iter <= max_iter % check if max number of iterations is reached

    % compute gradient and Hessian at current iterate x_curr
    g = Df(x_curr);
    H = Hf(x_curr);

    % Newton update step to find the next itearte x
    % A\b is the solution of Ax=b in MATLAB (very slow for large A)
    % and hence H\g = H^{-1}*g as needed in Newton's method
    x = x_curr - H\g;

    % print the iteration number and value of x in command window
    fprintf('iter = %d: f(x) = %.4f, x = %s. \n', iter, f(x), sprintf('%.4f', x(:)') );

    % check stopping criterion based on relative change
    if norm(x - x_curr) / norm(x) < relchg_tol
        break; % if met, terminate the iteration
    end

    % update current iterate x_curr for next iteration
    x_curr = x;

    % increase iteration number
    iter = iter + 1;

end

% check if the max number of iterations is used
if iter == max_iter
    disp('Maximum number of iteration reached when exited.');
end

x_opt = x;
f_opt = f(x);

return;
```

```
function [f, Df, Hf, x_dim] = test_prob1()
% produce a test problem 1:
%       minimize f(x) = 0.5*x^2 - sin(x)
%       where x is a 1-dimensional vector (scalar)
% OUTPUT:
%       f: function handle of f
%       Df: function handle of the gradient of f
%       Hf: function handle of the Hessian of f
%       x_dim: dimension of x

% define function f of x
f = @(x) 0.5*x^2 - sin(x);

% define gradient of f
Df = @(x) x - cos(x);

% define Hessian of f
Hf = @(x) 1 + sin(x);

% set dimension of x
x_dim = 1;

return;
```

```
function [f, Df, Hf, x_dim] = test_prob2()
% produce a test problem 1:
%       minimize f(x) = 100*(x(2) - x(1)^2)^2 + (1 - x(1))^2
%       where x is a 2-dimensional vector
% OUTPUT:
%       f: function handle of f
%       Df: function handle of the gradient of f
%       Hf: function handle of the Hessian of f
%       x_dim: dimension of x

% define function f of x
f = @(x) 100*(x(2)-x(1)^2)^2 + (1-x(1))^2;

% define gradient of f
Df = @(x) [(400*x(1)^3 + 2*x(1) - 400*x(1)*x(2) - 2); (-200*x(1)^2 + 200*x(2))];

% define Hessian of f
Hf = @(x) [(1200*x(1)^2 + 2 - 400*x(2)), -400*x(1); -400*x(1), 200];

% set dimension of x
x_dim = 2;

return;
```

```
>> demo_newton
iter = 1: f(x) = 100.0000, x = 1.00000.0000.
iter = 2: f(x) = 0.0000, x = 1.00001.0000.
iter = 3: f(x) = 0.0000, x = 1.00001.0000.
Final output x: 1.000000 1.000000 .
Final output f: 0.000000.
>>
```