```
% Jim Vargas
% MTH 410 HW 2
format compact
clc

% SET 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
"Problem 1"; disp("PROBLEM 1 ################################");
A=[1 1/2;
    1/2 1];
b=[1 -1]';
c=0;
epsilon=10^(-4);
x0=[5 10]';
"(a)";
    disp("BM ELS");
    [x,fvalue,iters]=GradientMethodE(A,b,c,x0,epsilon);
    disp("Approx. optimal solution x:"), disp(x);
    fprintf("Associated optimal value f(x): %6.4f\n", fvalue);
    fprintf("Number of iterations: %6.0f\n", iters);
"(b)";
    disp(newline+"GM const step size");
    t=.1;
    [x,fvalue,iters]=GradientMethod(A,b,c,x0,t,epsilon);
    disp("Approx. optimal solution x:"), disp(x);
    fprintf("Associated optimal value f(x): %6.4f\n", fvalue);
    fprintf("Number of iterations: %6.0f\n", iters);
"(c)";
    disp(newline+"GM backtracking");
    alpha=.5; beta=.5; s=1; % parameters
    [x,fvalue,iters]=GradientMethodB(A,b,c,x0,epsilon,alpha,beta,s);
    disp("Approx. optimal solution x:"), disp(x);
    fprintf("Associated optimal value f(x): %6.4f\n", fvalue);
    fprintf("Number of iterations: %6.0f\n", iters);


"Problem 2"; disp(newline+""+newline+"PROBLEM 2 ################################");
A=hilb(5); % 5x5 Hilbert Matrix: A_{i,j} = \frac{1}{i+j-1}, i=1,2,...,5
b=[0,0,0,0,0]';
c=0;
epsilon=10^(-2);
x0=[1,2,3,4,5]';
% Two newline concatenated does not seem to work (?)
"(a)";
    disp("BM ELS");
    [x,fvalue,iters]=GradientMethodE(A,b,c,x0,epsilon);
    disp("Approx. optimal solution x:"), disp(x);
    fprintf("Associated optimal value f(x): %6.4f\n", fvalue);
    fprintf("Number of iterations: %6.0f\n", iters);
"(b)";
    disp(newline+"GM const step size");
    t=.1;
```

```
    [x,fvalue,iters]=GradientMethod(A,b,c,x0,t,epsilon);
    disp("Approx. optimal solution x:"), disp(x);
    fprintf("Associated optimal value f(x): %6.4f\n", fvalue);
    fprintf("Number of iterations: %6.0f\n", iters);
"(c)";
    disp(newline+"GM backtracking");
    alpha=.5; beta=.5; s=1; % parameters
    [x,fvalue,iters]=GradientMethodB(A,b,c,x0,epsilon,alpha,beta,s);
    disp("Approx. optimal solution x:"), disp(x);
    fprintf("Associated optimal value f(x): %6.4f\n", fvalue);
    fprintf("Number of iterations: %6.0f\n", iters);


"Problem 3"; disp(newline+""+newline+"PROBLEM 3 ################################");
"(a)";
    % Wish to use linear regression to predict house price
    % Relies on "Housing.txt" for data
    % Data in .txt is formatted [SQUARE_FEET (ft^2), BEDROOMS (integer), PRICE ($)]
    X=load("Housing.txt"); % the full data in matrix form
    X1=X(:,1:2); % input data
    y1=X(:,3); % output data

    % Training sets
    sz=size(X1);
    m=round(.8*sz(1)); % use about 80% of data for training; this is 38 in this case
    Xtrain=X1(1:m,:);
    y=y1(1:m);
    O=ones(m,1);
    A=[O,Xtrain];

    f=@(u,z) u'*z; % The linear-regressed function.
                   % Inputs to f are u: the data variable (like x) and
                   % z: the coeffs. found with minimizer.
                   % Order of inputs does not really matter though
                   % with inner product for real vectors
    w0=[4.608717759244*10^4,1.520512967679*10^2,-1.554406070409*10^3]';
    epsilon=10^(-2);
    % w0 chosen judiciously here to be nearly the optimal solution
    % Not sure why so many digits are needed...
    % Only four sig figs shoul be needed

    disp(newline+"GM const step size");
    [w, iters]=GMregression(A,y,w0,m,epsilon);
    disp("Approx. optimal coefficients w_0, w_1, w_2:"), disp(w);
    fprintf("Number of iterations: %6.0f\n", iters);

    disp("Prediction for price of house with 2080 ft^2, 4 bedrooms:");
    prediction_vector=[1,2080,4]';
    disp(f(prediction_vector,w)); % $356,232.

    % TODO: implement a test for error with testing data (not used in training)
```

```
    % TODO: implememt linear regression with exact line and backtracking
    % TODO: part (b) of the bonus problem with "TwinCityHomes.csv"




% SET 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
"Problem 1"; disp(newline+""+newline+"PROBLEM 1, SET 2
################################");
"(c)";
    epsilon=10^(-2);
    x0=[2,5]';
    alpha=.25; beta=.5; s=2;

    % Basically rewriting GM backtracking for spefic f
    x=x0;
    fvl=@(u) 100*(u(2) - (u(1))^2)^2 +(1 - u(1))^2;
    gd=@(u) [400*(u(1))^3 - 400*u(1)*u(2) + 2*u(1) - 2 , 200*(u(2) - (u(1))^2)]';
    grad=gd(x);
    iterations=0;

    while (norm(grad)>epsilon)
        t=s;
        while (fvl(x)-fvl(x-t*grad)<alpha*t*(norm(grad))^2)
            iterations=iterations+1;
            t=beta*t;
        end % outside this while loop there seems to be fewer iterations

        x=x-t*grad;
        grad=gd(x);
        fvalue=fvl(x);
    end

    % Optimal value calculated analytically to be x=(1,1) with f(x)=0
    disp("Approx. optimal solution x:"), disp(x);
    fprintf("Associated optimal value f(x): %6.4f\n", fvalue);
    fprintf("Number of iterations: %6.0f\n", iterations);
    % this returns x=(1.0095,1.0192) with f(x)=.0001 in 80 iterations

    % Possible TODO: re-write the gradient methods so that f, grad(f) can
    % be passed in
    % Would this be redundant since in practice, f, grad(f) may not have
    % closed form?
```