CODE APPENDIX

```
%========================================================================
%== ASSIGNMENT: hw1
%== AUTHOR: Jared Hansen
%== DUE: Thursday, 09/12/2019
%========================================================================



clear all; close all; clc;
%========================================================================
%========================================================================
%== PROBLEM 1
%========================================================================
%========================================================================

%========================================================================
%== 1(B)
%========================================================================
% Coefficients of the objective function (price per serving in $ for
% carrots, potatoes, bread, cheese, and peanut butter respectively)
f = [0.14  0.12  0.20  0.75  0.15];
% Creating the matrix that specifies caloric and macro-nutrient
 content
A(1,:) = [-23   -171   -65   -112   -188];   % >= 2000 calorie
 constraint
A(2,:) = [-0.1  -0.2   0     -9.3   -16];    % >= 50g fat constraint
A(3,:) = [-0.6  -3.7   -2.2  -7.0   -7.7];   % >= 100g protein
 constraint
A(4,:) = [-6    -30    -13   0      -2];     % >= 250g carbs
 constraint
b = [-2000; -50; -100; -250];                % RHS of ineq constraints
lb = zeros(5,1);                  % Lower bounds for # of servings
ub = [inf; inf; inf; inf; inf;];  % Upper bounds for # of servings

%========================================================================
%== 1(C)
%========================================================================
% Use linprog function to solve (cont_soln = continuous solution)
cont_soln = linprog(f, A, b, [], [], lb, ub);
% Display the continuous solution
disp("Continuous solution:")
disp(cont_soln);

%========================================================================
%== 1(D)
%========================================================================
% Use intlinprog function to solve (int_soln = integer solution)
int_soln = intlinprog(f, [1:5], A, b, [], [], lb, ub);
% Display the integer solution
disp("Integer solution:")
disp(int_soln);
```

```matlab
clear all; close all; clc;
%==========================================================================
%==========================================================================
%== PROBLEM 3
%==========================================================================
%==========================================================================
% DIRECTIONS: in MATLAB create a noisy data set based on a quadratic
% function with coefficients all equal to 1.
x  = linspace(0,1)';
y  = 1 + 1*x + 1*x.^2;
ym = y + 0.1*randn(100,1);

%==========================================================================
%== 3(A)
%==========================================================================
figure, plot(x, y, '-', x, ym, 'o'), grid on

%==========================================================================
%== 3(C)
%==========================================================================
% Matrix containing polynomial terms for each x
M = [x.^2, x, ones(100,1)];
% The quadratic term, H, for the quadprog function
H = (M')*(M);
% The linear term, f, for the quadprog function
f = -((ym')*(M));
% Estimate the coefficients d=[a,b,c] where (y_hat = ax^2 + bx + c)
fitted = quadprog(H,f)
disp(fitted)
% Just for fun, let's see how well our estimate does VS the true model
truth_error = norm((1*x.^2 + 1*x + 1) - (ym))
model_error = norm((fitted(1)*x.^2 + fitted(2)*x + fitted(3)) - (ym))
% The model has a lower error than the true data generating function.
```

```
clear all; close all; clc;
%==========================================================================
%==========================================================================
%== PROBLEM 7
%==========================================================================
%==========================================================================

%==========================================================================
%== 7(B)
%==========================================================================
% NOTE: I did borrow much of this from MATLBAB's documentation for
%       fmincon but tried to make comments to reflect my
 understanding.
%       https://www.mathworks.com/help/optim/ug/fmincon.html

% The Rosenbrock function in terms of the 2-dim vector x = [x(1),
 x(2)]
fctn = @(x) (1-x(1))^2 + 100*(x(2) - x(1)^2)^2;
% This initial guess took too many iterations so MATLAB gave up
% x0 = [500,1000];
% This initial guess allows the the solver to converge to an answer w/
in
% acceptable number of iterations (since it doesn't quit before
 solving)
x0 = [3,3];
% Use fmincon to solve
soln = fmincon(fctn,x0);
% Display the solution: ends up being x = [x(1)=1, x(2)=1] w/o constr
disp(soln)

% With ineq. and eq. constraints of: [x(1)+2x(2) <= 1], [2x(1)+x(2) =
 1]
A = [1,2];      % Matrix for the inequality constraint
b = 1;          % Vector for the inequality constraint
Aeq = [2,1];    % Matrix for the equality constraint
beq = 1;        % Vector for the equality constraint
soln_2 = fmincon(fctn, x0, A, b, Aeq, beq);
disp(soln_2);
```

```
Optimal solution found.

Continuous solution:
         0
    7.7147
         0
         0
    9.2800

LP:                Optimal objective value is 2.317755.
```

```
Heuristics:        Found 1 solution using rounding.

                   Upper bound is 2.460000.

                   Relative gap is 4.05%.


Branch and Bound:

   nodes      total   num int        integer       relative

explored  time (s)  solution           fval        gap (%)

       5      0.00         2   2.430000e+00   0.000000e+00


Optimal solution found.

Intlinprog stopped because the objective value is within a gap
 tolerance of the
optimal value, options.AbsoluteGapTolerance = 0 (the default value).
 The intcon
variables are integer within tolerance, options.IntegerTolerance =
 1e-05 (the
default value).

Integer solution:
     0
     9
     0
     0
     9


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.


fitted =

    1.0396
    0.9724
    1.0049

    1.0396
    0.9724
    1.0049
```

```
truth_error =

    0.9564


model_error =

    0.9543


Local 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.

    1.0000    1.0000


Local 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.

    0.4149    0.1701
```

Published with MATLAB® R2019a