ME C231 Assignment: Optimization: 1

1. **Linear programming, by hand:** For the following problems, draw a sketch of the feasible region and a few objective function contours (level curves). Then, write down which traits the problem has from the following list:

   - feasible
   - infeasible
   - bounded
   - unbounded
   - has a unique optimum
   - has multiple optima

   (a)

$$\min_{z_1,z_2} \; 3z_1 + 2z_2 \tag{1a}$$
$$\text{s.t.} \; z_1 \geq 0 \tag{1b}$$
$$z_2 \geq 0 \tag{1c}$$

   (b)

$$\min_{z_1,z_2} \; z_1 \tag{2a}$$
$$\text{s.t.} \; z_1 \geq 0 \tag{2b}$$
$$z_2 \geq 0 \tag{2c}$$

   (c)

$$\min_{z_1,z_2} \; -5z_1 - 7z_2 \tag{3a}$$
$$\text{s.t.} \; -3z_1 + 2z_2 \leq 30 \tag{3b}$$
$$-2z_1 + z_2 \leq 12 \tag{3c}$$
$$z_1 \geq 0 \tag{3d}$$
$$z_2 \geq 0 \tag{3e}$$

   (d)

$$\min_{z_1,z_2} \; z_1 - z_2 \tag{4a}$$
$$\text{s.t.} z_1 - z_2 \geq 2 \tag{4b}$$
$$2z_1 + z_2 \geq 1 \tag{4c}$$
$$z_1 \geq 0 \tag{4d}$$
$$z_2 \geq 0 \tag{4e}$$

(e)

$$\min_{z_1,z_2} \; 3z_1 + z_2 \tag{5a}$$
$$\text{s.t. } z_1 - z_2 \leq 1 \tag{5b}$$
$$3z_1 + 2z_2 \leq 12 \tag{5c}$$
$$2z_1 + 3z_2 \leq 3 \tag{5d}$$
$$-2z_1 + 3z_2 \geq 9 \tag{5e}$$
$$z_1 \geq 0 \tag{5f}$$
$$z_2 \geq 0 \tag{5g}$$

2. **Quadratic Programming, by hand:** For the following problems, draw a sketch of the feasible region and a few objective function contours (level curves). Mark the optimal solution point. Finally, for each constraint, determine whether it is active or inactive.

(a)

$$\min_{z_1,z_2} \; z_1^2 + z_2^2 \tag{6a}$$
$$\text{s.t. } z_1 \geq 1 \tag{6b}$$
$$z_2 \geq 1 \tag{6c}$$

(b)

$$\min_{z_1,z_2} \; 2z_1^2 + 7z_2^2 \tag{7a}$$
$$\text{s.t. } z_1 \geq -3 \tag{7b}$$
$$2 \geq z_2 \tag{7c}$$

(c)

$$\min_{z_1,z_2} \; z_1^2 + z_2^2 \tag{8a}$$
$$\text{s.t. } x_1 \leq -3 \tag{8b}$$
$$x_2 \leq 4 \tag{8c}$$
$$0 \geq 4z_1 + 3z_2 \tag{8d}$$

3. **Approximately verifying optimality:** Consider the optimization problem:

$$\min \frac{1}{2}(z_1^2 + z_2^2 + 0.1z_3^2) + 0.55\; z_3 \tag{9}$$
$$\begin{aligned} \text{subject to} \quad & z_1 + z_2 + z_3 = 1 \\ & z_1 \geq 0 \\ & z_2 \geq 0 \\ & z_3 \geq 0 \end{aligned}$$

(a) The point $z^* = (0.5, 0.5, 0)$ is a local minimum. Compute its cost as well as the cost of at least 3 other points in the neighborhood of z^* to convince yourself, approximately, that z^* is a local minimum.

(b) Is z^* also a global minimum? Explain why, or why not.

4. **Using** `linprog.m` **to solve LPs:** Run

begin code
```
help linprog
```
end code

in Matlab, and read about its features and syntax. Then use `linprog.m` to solve the following problem. Let $x, y, z \in \mathbb{R}$.

$$\begin{aligned} \min_{x,y,z} \quad & x + y + z \\ \text{subject to } \; & 2 \leq x \\ & -1 \leq y \\ & -3 \leq z \\ & x - y + z \geq 4 \end{aligned}$$

To get you started, here one form of a solution.

begin code
```
f = ones(3,1);
A = [-1 0 0; 0 -1 0; 0 0 -1; -1 1 -1];
b = [-2;1;3;-4];
[x,fval] = linprog(f,A,b)
```
end code

You can also use the elementwise bounds to solve the same problem in a slightly different way.

begin code
```
f = ones(3,1);
A = [-1 1 -1];
b = -4;
LB = [2;-1;-3];
[x,fval] = linprog(f,A,b,[],[],LB,[])
```
end code

You might get different values of x, y, z but the same cost for each solution. Why?

5. **Using** `quadprog.m`**:** Run

begin code
```
help quadprog
```
end code

in Matlab, and read about its features and syntax. Constrained least-squares problems are useful when your decision parameters are known to reside in a constrained set. Here, we show how to solve a constrained least-squares as a quadratic program. The constrained least-squares problem is of the form

$$\begin{aligned} &\min_x \|Ax - b\|_2^2 \\ &\text{s.t. } l_i \leq x_i \leq u_i. \end{aligned}$$

Observe that

$$\|Ax - b\|_2^2 = (x^T A^T - b^T)(Ax - b) = x^T A^T A x - 2b^T A x + b^T b$$

We can drop the constant term $b^T b$ in the optimization program and add it back later. The constrained least-squares problem becomes

$$\begin{aligned} &\min_x \quad \frac{1}{2} x^T \left(2A^T A\right) x + (-2A^T b)^T x \\ &\text{s.t. } l_i \leq x_i \leq u_i, \end{aligned}$$

which is a quadratic program.

The Matlab commands below solves the constrained least-squares problem using the same matrices A and b as above with $l_i = -0.5$ and $u_i = 0.5$.

begin code

```
H = 2*A'*A;
f = -2*A'*b;
x_cls = quadprog(H, f, [eye(5);-eye(5)], [0.5*ones(5,1);0.5*ones(5,1)]);
```

end code

Compare the result to an ad-hoc approach: standard least-squares, followed by variable "clipping" to enforce constraints "after-the-fact."

begin code

```
x_ls = A\b;  % get standard least-squares solution
x_ls(x_ls>0.5) = 0.5;  % set any entries that are greater than 0.5 to 0.5
x_ls(x_ls<-0.5) = -0.5;   % set any entries that are less than -0.5 to -0.5
% Compare performance (ie., cost function) to direct solution from QUADPROG
disp(norm(A*x_cls-b))
disp(norm(A*x_ls-b))
```

end code

6. **Getting Data from sMAP:** sMAP stands for a Simple Measurement and Actuation Profile (`https://code.google.com/p/smap-data/`) and is a product of the LoCal project at UC Berkeley (`http://local.cs.berkeley.edu/wiki2/index.php/Main_Page`). We will use this tool to download real data from real buildings. For a quick view of the online interface, go to `http://new.openbms.org/plot/`.

   A few steps are required before being able to access data from sMAP in Matlab.

   (a) Download the JavaSmap jar file and related contents in the java folder of the sMAP repository, specifically

   - go to the sMAP repository: `https://github.com/SoftwareDefinedBuildings/smap`
   - Click on the green "Clone or download" button
   - Select "Download ZIP"
   - Once the download finishes, extract the `.zip` file to a new folder. For the code described below, I (Andy) saved the files in a folder called

     `C:\Users\Andy\Documents\smap-master`

     You will save in a similar (though differently named) location.

   (b) Next, we need to tell Matlab where to look to find the jar files we just added. Run the following in Matlab. The lines are too long to show as single lines, so I am listing them in a manner that will actually work if you cut/paste. In general, you can just type the long lines into the Matlab command window.

   begin code

```
javaaddpath('C:\Users\Andy\Documents\smap-master\java\JavaSmap_0.1.jar')
javaaddpath(['C:\Users\Andy\Documents\smap-master\java\' ...
      'JavaSmap\lib\javax.json-1.0-b06.jar'])
javaaddpath(['C:\Users\Andy\Documents\smap-master\java\' ...
     'JavaSmap\lib\json-simple-1.1.1.jar'])
```

   end code

   Run these 3 lines (adjusting properly for your specific path) everytime you start Matlab if you want to access `sMAP`. The easiest and most efficient approach is to put these in an m-file named `addSMAP.m`, and simply run that

   begin code

```
addSMAP
```

   end code

   whenever you want to access `sMAP`.

   (b) Download, and install JSONLab from the Matlab file exchange. You can easily find this by searching

   `JSON, Matlab file exchange, encode`

   in Google. If you want the URL directly, it is: `http://www.mathworks.com/matlabcentral/fileexchange/33381-jsonlab--a-toolbox-to-encode-decode-json-files`

   Specific instructions are as follows:

   - Login to your Mathworks account to download the file

- The file is named `jsonlab-1.2.mltbx`
- After downloading, click and drag the file into the Matlab command window. This will start the toolbox installation process. You will need to accept the license terms, and then it should be installed automatically.

(c) Carefully read through the `datesAndTimesTutorial.m` and `MatlabSmapReadExample.m`. Run both,

```
datesAndTimesTutorial
MatlabSmapReadExample
```

and study the results.

7. **System Identification of a Building Zone Temperature Model:** In Homework 1, you computed a discrete time model of a building zone temperature that looks like the following equation:

$$T(k+1) = (1 - p_1 u_1(k))T(k) + p_2 u_1(k) u_2(k) + q(k)$$

where p_1 and p_2 are now parameters of the model that we want to identify using a least-squares approach. The sMAP example retrieves the data of the states and inputs for:

- zone temperature T in Matlab struct `RoomTempData`
- air mass flow rate u_1 in Matlab struct `FanData`
- supply air temperature u_2 in Matlab struct `SupplyTempData`

Note that we have access to fan speed data as opposed to air mass flow rate data. This is an approximation which does not hold in general, but is okay for our purposes.

To determine the parameters p_1 and p_2, use data from a weekend day and assume that the heat load $q(k)$ is zero (when there are typically few graduate students in the lab to add heat to the system). Specifically, follow these steps:

(a) Download data from Saturday, August 30, 2014. Use the day's data from 10 AM to 4 pm to identify your model. Verify using `diff` and `hist` that the data is nearly collected with uniform sample time, but that there is a few percent variability. Interpolate the times so that the data is sampled on every minute of the day (start at 10 AM, then 10:01 AM, etc.). **Hint:** Remember that the times have been converted into a serial date, and 1 minute is $\frac{1}{1440}$ of a day. Use the Matlab command `interp1.m`. As a illustration of `interp1`, look/understand at the example below

```
xData = cumsum([0 0.1+rand(1,19)])
yData = [sort(3*rand(1,10)) fliplr(sort(3*rand(1,10)))];
xQuery = 0.1:0.2:max(xData);
yInterpLinear = interp1(xData, yData, xQuery);
yInterpSpline = interp1(xData, yData, xQuery, 'spline');
subplot(2,1,1)
plot(xData, yData, 'k*', xQuery, yInterpLinear, '-or')
```

```
legend('Data Points', 'Linear Interpolated Values', 'Location','Best');
ylabel('y')
subplot(2,1,2)
plot(xData, yData, 'k*', xQuery, yInterpSpline, '-or')
legend('Data Points', 'Spline Interpolated Values', 'Location','Best');
xlabel('x')
ylabel('y')
```
end code

(b) To do least squares parameter estimation, we want to solve the following problem:

$$\min_{a,b,e} \sum_{k=1}^{N} \|e(k)\|_2^2 \tag{10}$$

$$\text{s.t. } T_{data}(k+1) = (1 - p_1 u_1(k))T_{data}(k) + p_2 u_1(k) u_2(k) + q(k) + e(k) \tag{11}$$

$$\forall k = \{0, \ldots, N-1\} \tag{12}$$

where N is the number of data samples used, T_{data} is the actual data that you downloaded from sMAP, and $e(k)$ is the one step model estimation error of the temperature, whose norm (over time) is minimized by proper choice of parameters. Populate the appropriate matrices A and b using the known data such that

$$\begin{bmatrix} e(1) \\ \vdots \\ e(N) \end{bmatrix} = Ap - b$$

where $A \in \mathbf{R}^{N \times 2}, b \in \mathbf{R}^N$ and p is the 2×1 unknown parameter vector. Perform the least squares estimation of the parameters using `quadprog.m`. Implement this code in a function

begin code
```
function estParm = bldgIdentification(Tdata, u1Seq, u2Seq)
```
end code

The input arguments `u1Seq` and `u2Seq` are of length N, while `Tdata` has length $N+1$. The code should formulate and solve the lesat squares problem, and return the parameter estimate as a 2×1 vector.

(b) See how predictive the model is over a long horizon. On a single figure, plot the day's actual Temperature data from 10 AM to 4 PM. Hint: use `datetick('x')` to make sense of the time on the x-axis. Then, compute and plot T_{est} by propagating the the model forward as follows:

$$T_{est}(0) = T_{actual}(0)$$
$$T_{est}(k+1) = (1 - a u_1(k))T_{est}(k) + b u_1(k) u_2(k) + q(k)$$

using the values for a and b as obtained by the least-squares solution.

(c) Rather than quadprog, there is a way to use Matlab's backslash operator (e.g. `A \ b`) to compute the same solution. What would A and b be? Compute the solution in Matlab to confirm your answer.

8. **Regression with $\|\cdot\|_1$, $\|\cdot\|_\infty$, and linear constraints:** Suppose $A_1 \in \mathbf{R}^{m\times n}$ and $b_1 \in \mathbf{R}^m$, while $A_\infty \in \mathbf{R}^{p\times n}$ and $b_\infty \in \mathbf{R}^p$. Finally $A_c \in \mathbf{R}^{q\times n}$ and $b_c \in \mathbf{R}^q$. Consider the optimization problem

$$\min_{x\in\mathbf{R}^n} \ \|A_1x - b_1\|_1 + \|A_\infty x - b_\infty\|_\infty \tag{13a}$$

$$\text{s.t. } A_c x \le b_c \tag{13b}$$

(a) Write a function `reg1Inf.m`, with function declaration line

begin code
```
function [xOpt, J] = reg1Inf(A1, b1, Ainf, binf, Ac, bc)
```
end code

which reformulates (using slack variables) into a linear program, and calls `linprog` to get the solution. If the problem is infeasible, then `xOpt` should return empty and `J` should return as `inf`. **Hint:** Rewatch the videos on linear programming, especially the last few, which talk about $\|\cdot\|_1$ and $\|\cdot\|_\infty$ regression problems, and how to reformulate as LPs. This problem combines the two ideas, as well as imposing additional linear inequality constraints.

(b) Test your code on the following small example

$$\begin{aligned} &\min \ \left\| \begin{bmatrix} z_1 \\ z_2+5 \end{bmatrix} \right\|_1 + \left\| \begin{bmatrix} z_1-2 \\ z_2 \end{bmatrix} \right\|_\infty \\ \text{subject to} \quad & 3z_1 + 2z_2 \le -3 \\ & 0 \le z_1 \le 2 \\ & -2 \le z_2 \le 3 \end{aligned}$$