Mathematical Programming Project

A quantity $\boldsymbol{b}$ is known to to depend upon another quantity $\boldsymbol{a}$. A set of corresponding values have been collected for $\boldsymbol{a}$ and $\boldsymbol{b}$ and are presented in vector format below:

$$\boldsymbol{a} = (0, 0.5, 1, 1.5, 1.9, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.6, 7, 7.6, 8.5, 9, 10)^\top$$
$$\boldsymbol{b} = (1, 0.9, 0.7, 1.5, 2.0, 2.4, 3.2, 2, 2.7, 3.5, 1, 4, 3.6, 2.7, 5.7, 4.6, 6, 6.8, 7.3)^\top$$

In particular, $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors in $\mathbb{R}^{19}$. We wish to find the quadratic polynomial

$$p_{\boldsymbol{x}}(t) = p_{(x_1,x_2,x_3)}(t) = x_1 t^2 + x_2 t + x_3$$

whose graph bests fits the set of data points in the sense that it produces the smallest sum of absolute deviations of each observe value of $\boldsymbol{b}$ from the predicted value $p_{\boldsymbol{x}}(\boldsymbol{a})$. In other words, we wish to solve the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{19} |p_{\boldsymbol{x}}(a_i) - b_i| \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^3. \end{array}$$

An optimal solution $\boldsymbol{x}^* = (x_1^*, x_2^*, x_3^*)^\top$ to this optimization problem will given us a quadratic polynomial $p_{\boldsymbol{x}^*}(t) = x_1^* t^2 + x_2^* t + x_3^*$ whose graph $C_{\boldsymbol{x}^*}$ best fits the data in the sense described above. By expressing $p_{\boldsymbol{x}}(t)$ in terms of its coefficients, we see that this optimization problem has the form:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{19} |a_i^2 x_1 + a_i x_2 + x_3 - b_i| \\ \text{subject to} & \boldsymbol{x} \in \mathbb{R}^3. \end{array}$$

At the moment, this optimization problem is not a linear programming problem because there is an absolute value in objective function; however, we can convert the optimization problem into a linear programming problem by indroducing new variables $x_4, x_5, \ldots, x_{22} \geq 0$ and setting $x_{i+3} = a_i^2 x_1 + a_i x_2 + x_3 - b_i$ or all $1 \leq i \leq 19$. We obtain a new optimization problem which has the form:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{i=4}^{22} x_i & \\ \text{subject to} & a_i^2 x_1 + a_i x_2 + x_3 - x_{i+3} = b_i & \text{for all } 1 \leq i \leq 19 \\ & x_{i+3} \geq 0 & \text{for all } 1 \leq i \leq 19 \\ & x_1, x_2, x_3 \in \mathbb{R}. & \end{array}$$

This new optizimation problem has the correct form for it to be considered a linear programming problem. It is easy to see that $\widetilde{\boldsymbol{x}}^* = (x_1^*, x_2^*, x_3^*, \ldots, x_{22}^*)^\top$ is an optimal solution to the new linear programming problem if and only if $\boldsymbol{x}^* = (x_1^*, x_2^*, x_3^*)^\top$ is an optimal solution to our original optimalization problem.

We will find an optimal solution to this linear programming problem using MATLAB, which has a built-in function whose purpose is to solve linear programming problems like this. The syntax for this function is [x,cval] = linprog(c,Ain,bin,Aeq,beq), where the linear program solver assumes that the linear program has the form

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} & A_{\text{eq}} \boldsymbol{x} = \boldsymbol{b}_{\text{eq}} \\ & A_{\text{in}} \boldsymbol{x} \leq \boldsymbol{b}_{\text{in}} \\ & \boldsymbol{x} \in \mathbb{R}^{22} \end{array}$$

So in order to use this funciton, we need to place our linear program in to this form. Let V be the 19×3 Vandermonde matrix given by

$$V = \begin{pmatrix} a_1^2 & a_1 & 1 \\ \vdots & \vdots & \vdots \\ a_i^2 & a_i & 1 \\ \vdots & \vdots & \vdots \\ a_{19}^2 & a_{19} & 1 \end{pmatrix},$$

let A_{eq} be the 19×22 matrix given by $A_{\text{eq}} = \begin{pmatrix} V & -\mathrm{I}_{19} \end{pmatrix}$ where I_{19} is the 19×19 identity matrix, and let $\boldsymbol{b}_{\text{eq}} = \boldsymbol{b} \in \mathbb{R}^{19}$. Then

$$A_{\text{eq}}\boldsymbol{x} = \begin{pmatrix} a_1^2 & a_1 & 1 & -1 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & 0 & \ddots & \ddots & & \vdots \\ a_i^2 & a_i & 1 & \vdots & \ddots & -1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \ddots & 0 \\ a_{19}^2 & a_{19} & 1 & 0 & \cdots & \cdots & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_{22} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_{19} \end{pmatrix} = \boldsymbol{b}_{\text{eq}}$$

gives us our equality constraint. Next, let A_{in} be the 19×22 matrix $A_{\text{in}} = \begin{pmatrix} 0 & 0 & 0 & -\mathrm{I}_{19} \end{pmatrix}$ and let $\boldsymbol{b}_{\text{in}} = 0 \in \mathbb{R}^{19}$. Then

$$A_{\text{in}}\boldsymbol{x} = \begin{pmatrix} 0 & 0 & 0 & -1 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & \vdots & \ddots & -1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_{22} \end{pmatrix} \leq \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{pmatrix} = \boldsymbol{b}_{\text{in}}$$

gives us our inequality constraints. Finally, let $\boldsymbol{c}$ be the vector $\boldsymbol{c} = (0, 0, 0, 1, \ldots, 1, \ldots, 1)^\top \in \mathbb{R}^{22}$. Then $\boldsymbol{c}^\top \boldsymbol{z}$ gives us our objective function. We are now ready to work in MATLAB. We write a MATLAB function function [x,p1,l1,p2,l2,plot1,plot2] = OptimalPolynomialFittingDataL1L2(a,b,deg) which solves the more general problem (where $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$ and $p_x(t)$ has degree m with $m \leq n$) and then apply it our special case. The function is given in the code below:

```
function [x,p1,l1,p2,l2] = OptimalPolynomialFittingDataL1L2(a,b,deg)

% We assume that length(a)=length(b)>=deg.

m = deg;
n = length(a);

% Find optimal l1 solution using linprog. The vector p1 returns the coefficients
% of the polynomial which has optimal l1 distance from b.

beq = b;
bin = zeros(n,1);
Aeq = [a.^(m:-1:0), -eye(n)];
Ain = [zeros(n,m+1), -eye(n)];
c = [zeros(m+1,1); ones(n,1)];
[x,l1] = linprog(c,Ain,bin,Aeq,beq);
p1 = (x(1:m+1));

% Find optimal l2 solution using polyfit. The vector p2 returns the coefficients
% of the polynomial which has optimal l2 distance from b.

[p2,l2] = polyfit(a,b,m);
```

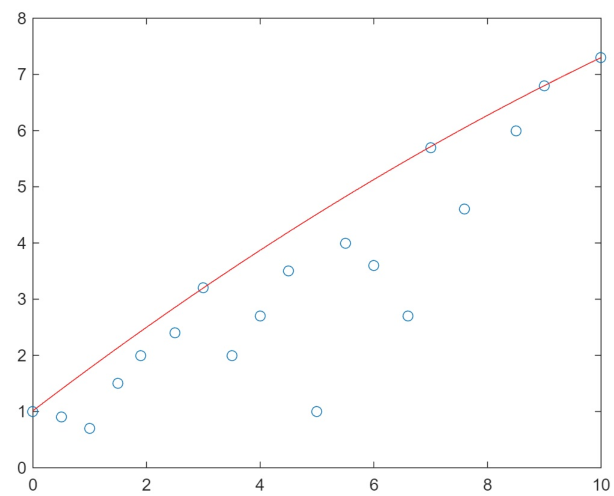
Let us now make use of this function. First we find the optimal solution to our original problem:

```
% Initial data

a = [0; 0.5; 1; 1.5; 1.9; 2.5; 3; 3.5; 4; 4.5; 5; 5.5; 6; 6.6; 7; 7.6; 8.5; 9; 10];
b = [1; 0.9; 0.7; 1.5; 2; 2.4; 3.2; 2; 2.7; 3.5; 1; 4; 3.6; 2.7; 5.7; 4.6; 6; 6.8; 7.3];
deg = 2;

% Use OptimalPolynomialFittingDataL1L2 function to find p1 and p2

[p1,l1,p2,l2] = OptimalPolynomialFittingDataL1L2(a,b,deg);

% Plot our optimal solution to visualize how it fits the data

t0 = min(a);
t1 = max(a);
t = linspace(t0,t1);
plot(a,b,'o',t,polyval(p1,t),'r');
plot(a,b,'o',t,polyval(p1,t),'r',t,polyval(p2,t),'b');
```

MATLAB tells us that the optimal ℓ_1 solution is given by the polynomial

$$p(t) = -0.0143t^2 + 0.7714t + 1.0143,$$

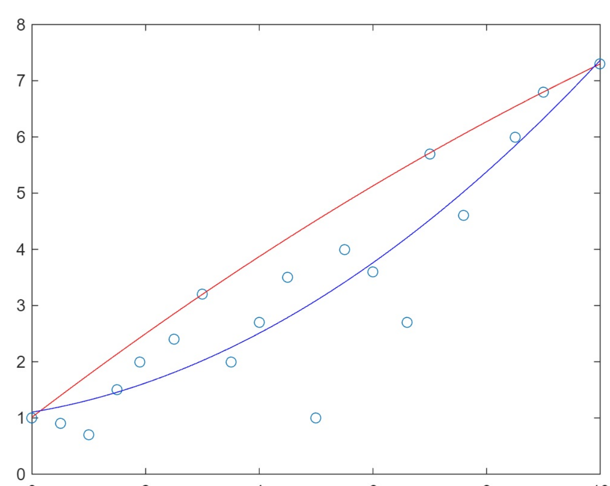
corresponding to point $\boldsymbol{x}^* = (-0.0143, 0.7714, 1.0143)$. MATLAB also gives us the following plot:



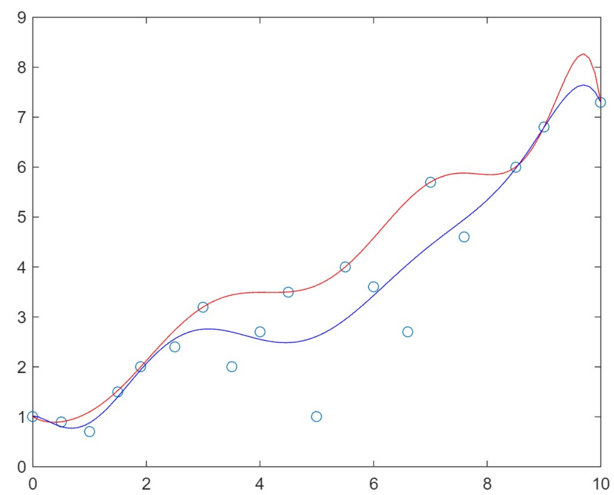
where the red curve is the graph of $p(t)$ and the data points are plotted using blue circles. Our function also returns an optimal ℓ_2 solution, which is given by the polynomial

$$q(t) = 0.0458t^2 + 0.168t + 1.1036.$$

Let's plot the graph of $q(t)$ together with the graph of $p(t)$ and the data points. MATLAB gives us the following plot:



where the blue curve is the graph of $q(t)$. Note that our function finds the optimal ℓ_1 and optimal ℓ_2 solutions in the space of degree $\leq m$ polynomials where $1 \leq m \leq \text{length}(\boldsymbol{a})$. Let's see what the degree ≤ 9 optimal solutions look like. To do this, we simply set deg=9 and run the code agove again. MATLAB outputs the following plot:



What's happening here is that the ℓ_2 optimal solution is much more sensisitive to the "outlier" data than the ℓ_1 solution is. In fact, the ℓ_1 optimal solution is perfectly happy ignoring outliers, so long as stays very close to most of the data points (and the outliers aren't *too* far away). The ℓ_2 optimal solution on the other hand tries to be much more inclusion, taking into account all of the outliers (as well as the majority).