Linear Programming: A Discussion of its Applications in Operations Research, Curve Fitting, and an Explanation of the Simplex Algorithm

Victor Wright

May 2017

Introduction

In an multivariable optimization problem we seek to discover if a function in two or more variables can attain a global optimum on its domain [1, 2]. A multivariable optimization problem is constrained if the function must be optimized subject to side conditions [1, 2, 3]. Linear programming problems are a class constrained multivariable optimization problems [1, 3]. Solutions to these types of multivariable optimization problems are usually found by utilizing matrix algebra or computational software [4]. In this paper, we give a detailed description of the linear programming problem, an in-depth explanation of the simplex algorithm, and present an example of a linear program to fit a linear and quadratic model to a data set.

Optimization Problems and Linear Programming

In this section we describe optimization in data analysis and the petroleum, agriculture, and health care industries. Linear programming models can be built to represent an organizations *logistic*, *economic*, or *social* process [4]. Linear programming models are popular in a multitude of industries [4]. One

industry in which they are popular is the petroleum industry [4]. This is because they can be formulated and solved to discover optimal *distribution routes* of crude oil [1, 4]. Analysis of linear programming models can also determine optimal agricultural product prices or minimize crop fertilizer and livestock feed costs in the agricultural industry [4]. Finally, linear programming models have been formulated and solved to determine nurse schedules in clinics, how components of medical devices used to treat malignant tumors should function resulting in higher quality health care, or data analysis where the goal is to find the 'best' fit lines and quadratic curves to a data set [4].

An organizations logistic, economic, or social process can be constrained to labor capacity, advertising budgets, scarce resources, or hours of operation [4]. To accurately represent constrained logistic, economic, or social processes of an organization, the constraints are required to appear in the algebraic foundation of the model [4]. Nonetheless, the key point here is that in different industries these models can be used to *optimize* an organizations logistic, economic, or social processes. Thus, these models represent *optimization problems*. Essentially, we wish to represent the organizations logistic, economic, or social operations as a *mathematical program* since they are destined to be mathematical in nature and must be optimized [4]. It is here that we define what a mathematical program is, how it should be constructed to represent the organizations operating limitations, and its class before we attempt to solve these optimization problems.

To optimize the logistic, economic, or social process of an organiztion the first step is to identify and correctly write down the *objective* that describes some arbitrary quantity the organization feels should attain the smallest or largest viable value [4]. The objective is the part of the mathematical program that we wish to optimize. However, the objective is optimized when *decisions* are made. Therefore, the decisions that the organization must make at a single time are represented by *independent variables* which we also call *decision variables* [1]. In a mathematical program, a decision variable can represent a route in a logistics problem, an agricultural good being produced, the number of nurses scheduled to work a particular shift, or a set of data points. Since we seek a small or large objective quantity, we choose to mathematically display the objective as some function

$$f(x_1, x_2, ..., x_n),$$

where all f are assumed to be differentiable [1]. A solution that will

optimize the objective will be found by solving a *multivariable optimization problem* [1]. This is the definition of a *mathematical program* due to the fact that we represent the objective as a function we wish to optimize [4]. However, the amount of decisions that can be made are often *constrained* by some operating limits [4]. Thus, it is suitable to write the optimization problem with side conditions as:

optimize

$$f(x_1, x_2, ..., x_n), \tag{1}$$

subject to

$$A\vec{x} \leq \vec{b}, \tag{2}$$

and $x_1, x_2, ..., x_n \geq 0$. Here, A is an $m \times n$ coefficient matrix, $\vec{x}$ is a column vector of our decision variables x_i for $i = 1, 2, ..., n$, and $\vec{b}$ is a column vector of *data* b_j for $j = 1, 2, ..., m$ [3]. Next, assume that the objective can be written as a *linear* function $f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n$ with *objective data* c_i. The optimization problem reads

optimize

$$f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n, \tag{3}$$

subject to

$$\begin{cases} a_{1,1}x_1 + a_{12}x_2 + ... + a_{1,n}x_n \leq b_1, \\ a_{2,1}x_1 + a_{2,2}x_2 + ... + a_{2,n}x_n \leq b_2, \\ \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + ... + a_{m,n}x_n \leq b_m, \end{cases} \tag{4}$$

and $x_1 \geq 0$, $x_2 \geq 0$, ..., $x_n \geq 0$ [1, 3]. This model is formally referred to as a *linear programming model* written in *inequality* or *standard* form [1, 3].

Now the column vector of data $\vec{b}$ in (4) is often referred to as the linear programs *side conditions* [3]. For our purposes we choose to refer to the data in $\vec{b}$ as *constraint data*. When all values in $\vec{b}$ are known, this means that the organizations operational limit will be represented in the linear program. For

example, if a organization fabricates i goods and each decision variable x_i in f represents good i being manufactured subject to the previously mentioned data, we wish to conclude what possible quantities of finished good i will optimize the objective in question [4]. In this example, the number of decisions that could be made here are bounded from above by limiting values imposed by such data [4]. We now ask: how do we optimize f subject to $A\vec{x} \leq \vec{b}$?

$A\vec{x} \leq b$ is formally known as a *constraint system* or *feasible set* [1]. To make the problem easier to solve, an appropriate manipulation of the weak inequality forcing $A\vec{x} = \vec{b}$ is a wise choice since it forces the model to be solved on the boundary of the feasible set [1, 3]. Due to linearity, the feasible set exhibits *corner points* and thus it is desirable to search the feasible sets corners for a global maxima which is an *n-tuple* [1, 3]. In order to implement this iterative method, we write $A\vec{x} = \vec{b}$ as

$$\begin{cases} a_{1,1}x_1 + a_{12}x_2 + ... + a_{1,n}x_n + x_{n+1} = b_1, \\ a_{2,1}x_1 + a_{2,2}x_2 + ... + a_{2,n}x_n + x_{n+2} = b_2, \\ \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + ... + a_{m,n}x_n + x_{m+n} = b_n, \end{cases} \tag{5}$$

and $x_1, x_2, ..., x_{n+m-1}, x_{n+m} \geq 0$ [1, 3]. Now the idea of the new variables, x_{n+m} for $j = 1, 2, ..., m$, is to introduce some slack into each constraint row and are thus they are called *slack variables* [1]. The inequality constraints are transformed by introducing slack variables. They are introduced to solve linear programs that are written in inequality form. When the linear program is written in equality form and solutions attempts are made on it, candidate *n-tuples* are commonly calculated by using a variant of the *simplex algorithm* [1].

Explanation of the Simplex Algorithm

In this section we illustrate the simplex algorithm to solve a linear programming problem. Consider the following two-dimensional example.

Maximize

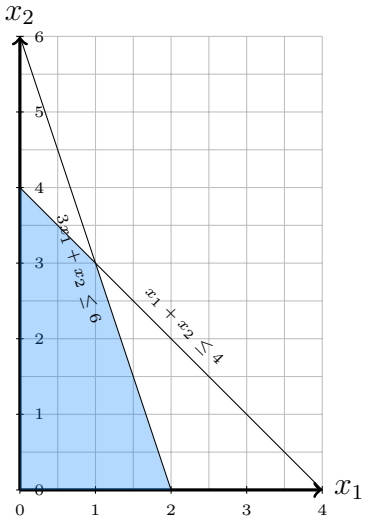

Figure 1: Illustration of the linear programs feasible set

$$f(x_1, x_2) = 2x_1 + x_2,$$

subject to

$$\begin{aligned} x_1 + x_2 &\leq 4, \\ 3x_1 + x_2 &\leq 6, \end{aligned}$$

and $x_1, x_2 \geq 0$ [3]. Using the constraint system, we can draw the feasible set and solve the problem graphically [3, 4].

"Scanning" the feasible set (Fig. 1) reveals three corners which are (0,4), (2,0), and (1,3). Checking each corner point by evaluating f at them reveals that (1,3) is the best choice to make. Similarly, the simplex algorithm scans the feasible set by checking and subsequently changing corners [4]. We next give a rigorous explanation of how a simplex algorithm would proceed to solve (3) subject to (4) in the context of maximization.

Let us refer to our objective function by a name and so we shall call the objective function μ [5]. Next, we introduce slack variables into (4) [5]. Let the slack variables in (5) be written as $w_j = x_{n+m}$ for $j = 1, 2, ..., m$ constraints and the decision variables retain their original appearance [5]. Then, the problem can be rewritten as

maximize

$$\mu = c_1x_1 + c_2x_2 + ... + c_nx_n, \tag{6}$$

subject to

$$\begin{cases} w_1 = b_1 - a_{1,1}x_1 - a_{1,2}x_2 - ... - a_{1,n}x_n, \\ w_2 = b_2 - a_{2,1}x_1 - a_{2,2}x_2 - ... - a_{2,n}x_n, \\ \quad \vdots \\ w_m = b_m - a_{m,1}x_1 - a_{m,2}x_2 - ... - a_{m,n}x_n, \end{cases} \tag{7}$$

and $x_1, x_2, ..., x_n, w_1, w_2, ..., w_m \geq 0$ [5]. We refer to (6) and (7) as a *dictionary* [5]. When the problem is written in this form, the simplex algorithm searches the feasible set $\vec{w} = \vec{b} - A\vec{x}$ for a unique optimal solution that will create the largest increase in μ [3, 5]. This search consists of two *phases*. In *phase one*, an *initial feasible solution* is constructed by setting the slack and decision variables to

$$\begin{cases} x_i = 0, \\ w_j = b_j, \end{cases} \tag{8}$$

for all i and j [5]. This choice satisfies the dictionary's non-negativity requirement and yields $\mu = 0$ [5]. However, this gives rise to the question,

is $\mu = 0$ the best we can do? If not, how does a simplex algorithm find a unique optimal solution that creates the largest increase in μ?

Suppose μ can be increased beyond the initial feasible solution. That is, μ is not optimal. When this is the case, the simplex method enters *phase two*. Phase two of the algorithm identifies a set $P = \{c_k > 0\}$, picks some x_k for $k \in i = 1, 2, ..., n$, and computes the *positive* values that fulfill the dictionary's constraints and would cause the objective value of μ to increase [5]. Once this x_k has been selected, the first iteration of phase two "transforms" the current dictionary into

$$\begin{cases} \mu = c_k x_k, \\ w_1 = b_1 - a_{1,k} x_k, \\ w_2 = b_2 - a_{2,k} x_k, \\ \vdots \\ w_k = 0 \\ \vdots \\ w_j = b_j - a_{j,k} x_k, \end{cases} \tag{9}$$

where w_k is a *leaving variable* and each w_j, $j \neq k$, are functions of the *entering variable* x_k [5]. However, the range of positive values that x_k can take must be feasible. That is $b_j - a_{j,k} x_k \geq w_k$ is required for each j that corresponds to every positive $a_{j,k} \in A$ [5]. This is the selection criterion for non-optimal μ and x_k. That is, when μ is not optimal, the choice of values that x_k can take are computed to satisfy the dictionarys non-negativity requirement such that the maximum feasible value of x_k will create the biggest increase in μ [5]. The non-negativity requirement imposed on the w_js will be satisfied when the feasible value of the leaving variable, w_k, is known [5]. To determine this value, a simplex algorithm sets each w_j equal to the minimum feasible value of w_k, which is zero, and solves $x_k = b_j/a_{j,k}$ j times [5]. After this step is completed, this data is stored in a set $M = \{b_j/a_{j,k} > 0\}$ [5]. Finally, the selection criterion for the feasible value of w_k is as follows: select an $l \in M$ such that each $b_j - a_{j,k} x_k \geq w_k$ when each w_j is evaluated at $x_k = b_l/a_{l,k}$, $b_l/a_{l,k}$ is the smallest element in M, l is chosen such that $b_l/a_{l,k}$ defines the *upperbound* of x_k's feasible range, and $(0, ..., b_l/a_{l,k}, ..., 0)$ is a *corner* of the feasible set [3, 5]. Furthermore, $\mu = c_k * b_l/a_{l,k}$ is the

improvement [3, 5]. If $P = \emptyset$ after this iteration, then the objective function value cannot be further increased [3, 5]. The corner is the unique optimal solution [3, 5]. On the other hand, if $P \neq \emptyset$, the change of basis as described above is repeated until the unique corner that will maximize μ is found [3, 5].

Suppose that after this iteration, P is nonempty. When this is the case, the next iteration begins with x_k and w_k introduced in (6) and (7). However, k is an *inactive* index due to the fact that μ has already been increased by $b_l/a_{l,k}$ [3, 5]. Thus, the simplex algorithm must create a new dictionary that contains the information from previous iteration [5]. For this to happen, x_k must become the *leaving variable* and w_k becomes the *entering variable*. For this to happen in phase two of the simplex method, the simplex algorithm computes a *change of basis* and then constructs the new dictionary. This change of basis is $\vec{w} = \vec{b} - A\vec{x} \longrightarrow \tilde{\vec{w}} = \vec{b} - A\tilde{\vec{x}}$, where

$$\begin{aligned} \tilde{\vec{w}} &= \begin{bmatrix} w_1 & w_2 & ... & x_k & ... & w_j \end{bmatrix}^T, \\ \tilde{\vec{x}} &= \begin{bmatrix} x_1 & x_2 & ... & w_k & ... & x_n \end{bmatrix}^T, \end{aligned} \tag{10}$$

and $x_k = \frac{b_k}{a_{k,k}} - \frac{1}{a_{k,k}} \sum_{i=1}^{k-1} a_{k,i}x_i - \frac{1}{a_{k,k}} w_k - \frac{1}{ak,k} \sum_{i=k+1}^{n} a_{k,i}x_i$ in the new constraint system $\vec{b} - A\tilde{\vec{x}}$ and $\tilde{\mu}$ as well [5]. Note that we solved for x_k in the kth constraint row of $\vec{w} = \vec{b} - A\vec{x}$. Thus, the new dictionary becomes

maximize

$$\tilde{\mu} = c_1x_1 + c_2x_2 + ... + c_kx_k + ... + c_nx_n$$

subject to

$$\begin{cases} w_1 = b_1 - a_{1,1}x_1 - a_{1,2}x_2 - ... + \dfrac{a_{1,k}}{a_{k,k}}x_k - ... - a_{1,n}x_n, \\ w_2 = b_2 - a_{2,1}x_1 - a_{2,2}x_2 - ... + \dfrac{a_{2,k}}{a_{k,k}}x_k - ... - a_{2,n}x_n, \\ \quad \vdots \\ x_k = \dfrac{b_k}{a_{k,k}} - \dfrac{a_{k,1}}{a_{k,k}}x_1 - \dfrac{a_{k,2}}{a_{k,k}}x_2 - ... - \dfrac{1}{a_{k,k}}w_k - ... \dfrac{a_{k,n}}{a_{k,k}}x_n, \\ \quad \vdots \\ w_m = b_m - a_{m,1}x_1 - a_{m,2}x_2 - ... + \dfrac{a_{m,k}}{a_{k,k}}x_k - ... - a_{m,n}x_n, \end{cases} \tag{11}$$

[5]. The change of basis is essentially a *row operation* that *swaps* w_k and x_k forcing them to play different roles in this new dictionary [5]. This change of variables is formally known as the Gauss-Jordan Algorithm where the pivot element was $a_{k,k}x_k$ in $\vec{w} = \vec{b} - A\vec{x}$ [3, 5]. Finally, the simplex method will enter phase one and then phase two until a successive iteration cannot further increase $\tilde{\mu}$. Hence, $\tilde{P} = \emptyset$ and $\tilde{\mu}$ is optimal [3, 5].

An Application of Linear Programming: Formulation of a Curve Fitting Problem and its Solution

In this section we discuss how to formulate a linear program to fit $\hat{y}_i = \hat{a}x_i + \hat{b}$, and quadratic curve, $\hat{y}_i = \hat{a}x_i^2 + \hat{b}x_i + \hat{c}$, to a collection of points, (x_i, y_i), in a data set [4]. We desire that these models will explain most of the true linear or quadratic relationship, $y_i = ax_i + b$ and $y_i = ax_i^2 + bx_i + c$, that exists between all of the data points (x_i, y_i) [4]. In other words, we wish to fit the 'best' $\hat{a}x_i + \hat{b} \approx y_i$ and $\hat{a}x_i^2 + \hat{b}x_i + \hat{c} \approx y_i$ to the data by formulating a linear program and solving it [4].

Consider the data set

Table 1: Table 1 is the data set that contains the collection of points (x_i, y_i) we wish to use linear programming to find a linear model $\hat{a}x_i + \hat{b}$ and quadratic curve $\hat{a}x_i^2 + \hat{b}x_i + \hat{c}$ that will have the 'best' fit to these data.

x	0.0	0.5	1.0	1.5	1.9	2.5	3.0	3.5	4.0	4.5
y	1.0	0.9	0.7	1.5	2.0	2.4	3.2	2.0	2.7	3.5
x	5.0	5.5	6.0	6.6	7.0	7.6	8.5	9.0	10.0	
y	1.0	4.0	3.6	2.7	5.7	4.6	6.0	6.8	7.3	

However, the parameters *a,* *b,* and c are all unknown and need to be approximated from these data. There are two approaches that we can use to find the 'best' estimates of *a,* *b,* and c [4]. The 'best' estimates can be found by

- Case I: $min! \sum_{i=1}^{19} |y_i - \hat{y}_i|$. In doing so $\hat{y}_i = \hat{a}x_i + \hat{b}$ and $\hat{y}_i = \hat{a}x_i^2 +$

$\hat{b}x_i + \hat{c}$ will be the best approximations of the true linear or quadratic relationship that exists in (Table 1).

- Case II: For each case we consider $|y_i - \hat{y}_i|$ in (Table 1) and the linear and quadratic estimations of these relationships. The 'best' fit straight line and quadratic curve can be found by minimizing the maximum deviation that occurs at each case i. That is we can find the best approximations to the true linear or quadratic relationship, $\hat{y}_i = \hat{a}x_i + \hat{b}$ and $\hat{y}_i = \hat{a}x_i^2 + \hat{b}x_i + \hat{c}$, in the data set (Table 1) by $min!(max|y_i - \hat{y}_i|)$.

[4]. We first discuss case I and how to formulate linear programming models that minimize the sum of *absolute deviations* for both models we wish to estimate from the data in (Table 1). Let u_i define the *horizontal* deviation between the true linear or quadratic relationship that exists between (x_i, y_i) and the models we wish to estimate for $i = 1, ..., 19$ cases [4]. Similarly, let v_i define the *vertical* deviation for $i = 1, ..., 19$ cases [4]. However for each case i, the deviation of u_i or v_i can be positive or negative [4]. Hence, *a, b,* and c must be defined to be *free* variables due to the fact that we are seeking the smallest $\sum_{i=1}^{19} |y_i - \hat{y}_i|$ [4]. In other words, we must the solve the linear programming model for all possible sign combinations of *a, b,* and c to find the smallest $\sum_{i=1}^{19} |y_i - \hat{y}_i|$ [3]. The linear programming model that must be solved to approximate $y_i = ax_i + b$ in the case I approach is

minimize

$$f(u_1, u_2, ..., u_{19}) + g(v_1, v_2, ..., v_{19}) = \sum_{i=1}^{19} u_i + \sum_{i=1}^{19} v_i \tag{12}$$

subject to

$$\begin{cases} ax_1 + b + u_1 - v_1 = y_1, \\ ax_2 + b + u_2 - v_2 = y_2, \\ \vdots \\ ax_{19} + b + u_{19} - v_{19} = y_{19}, \end{cases} \tag{13}$$

and $u_1, u_2, ..., u_{19}, v_1, v_2, ..., v_{19} \geq 0$ [4]. To estimate $ax_i^2 + bx_i + c$ in the case I approach we minimize (12) subject to (14).

$$\begin{cases} ax_1^2 + bx_1 + c + u_1 - v_1 = y_1, \\ ax_2^2 + bx_2 + c + u_2 - v_2 = y_2, \\ \vdots \\ ax_{19}^2 + bx_{19} + c + u_{19} - v_{19} = y_{19}, \end{cases} \tag{14}$$

and $u_1, u_2, ..., u_{19}, v_1, v_2, ..., v_{19} \geq 0$ [4]. But since *a, b,* and *c* are free variables we must solve (12) subject to (13) four times and (12) subject to (14) eight times in order to determine the optimal value of (12) for all possible sign combinations of *a, b,* and *c* [3]. Furthermore, since the constraints in (13) and (14) are equality constraints and the objective of case I is to $min! \sum_{i=1}^{19} |y_i - \hat{y}_i|$ we will obtain $\hat{a}x_i + \hat{b} \approx y_i$ and $\hat{a}x_i^2 + \hat{b}x_i + \hat{c} \approx y_i$ for each possible sign combination of *a, b,* and *c* when the models are solved. We solve the linear programming models with a simplex algorithm and list the results in the tables below.

Table 2: The optimal values in (Table 2) are the minimized sum of absolute deviations for all four attempts to find the 'best' fit line. $\hat{a}$ and $\hat{b}$ are the parameter estimates for $\hat{y}_i = \hat{a}x_i + \hat{b}$ obtained by using a simplex algorithm on (12) subject to (13). The 'best' fit line is $\hat{y}_i = 0.6375x_i + 0.5812$ since $\tilde{f} + \tilde{g} = 11.5000$ in the case I solution.

Results of Minimizing the Sum of Absolute Deviations			
Cases	Optimum	$\hat{a}$	$\hat{b}$
Case I: $a \geq 0$ and $b \geq 0$	11.5000	0.6375	0.5812
Case II: $a \geq 0$ and $b \leq 0$	12.6727	0.7273	0.0000
Case III: $a \leq 0$ and $b \geq 0$	30.5000	0.0000	2.7000
Case IV: $a \leq 0$ and $b \leq 0$	61.6000	0.000	0.000

Table 3: The optimal values in (Table 3) are the minimized sum of absolute deviations for all eight attempts to find the 'best' fit quadratic curve. $\hat{a}$, $\hat{b}$, and $\hat{c}$ are the parameter estimates for $\hat{y}_i = \hat{a}x_i^2 + \hat{b}x_i + \hat{c}$ obtained by using a simplex algorithm on (12) subject to (14). The 'best' fit quadratic curve is $\hat{y}_i = 0.0337x_i^2 + 0.2945x_i + 0.9823$ since $\tilde{f} + \tilde{g} = 10.4600$ in the case I solution.

Results of Minimizing the Sum of Absolute Deviations				
Cases	Optimum	$\hat{a}$	$\hat{b}$	$\hat{c}$
Case I: $a \geq 0$, $b \geq 0$, and $c \geq 0$	10.4600	0.0337	0.2945	0.9823
Case II: $a \geq 0$, $b \geq 0$, and $c \leq 0$	12.6439	0.0033	0.6967	0.0000
Case III: $a \geq 0$, $b \leq 0$, and $c \leq 0$	20.0299	0.0830	0.0000	0.0000
Case IV: $a \leq 0$, $b \geq 0$, and $c \geq 0$	11.5000	0.0000	0.6375	0.5812
Case V: $a \leq 0$, $b \geq 0$, and $c \leq 0$	12.6727	0.0000	0.7273	0.0000
Case VI: $a \geq 0$, $b \leq 0$, and $c \geq 0$	11.7421	0.6423	0.0000	1.3553
Case VII: $a \leq 0$, $b \leq 0$, and $c \geq 0$	30.5000	0.0000	0.0000	2.7000
Case VIII: $a \leq 0$, $b \leq 0$, and $c \leq 0$	59.6000	0.0000	0.0000	0.0000

We next discuss case II. The objective of case II is to minimize the *maximum deviation* that occurs at each case i. For each case i, $\hat{a}x_i + \hat{b}$ and $\hat{a}x_i^2 + \hat{b}x_i + \hat{c}$ will deviate from y_i by some horizontal distance u_i and vertical distance v_i. Note that it is possible to have $u_i = v_i = 0$ for some of the cases. In the event that u_i and v_i are both zero for some case i, this implies that $y_i = \hat{y}_i$. When $\hat{y}_i$ is not a perfect fit to y_i, then u_i and v_i must be nonzero. For the $i = 1, 2, ..., 19$ cases, we assume that u_i and v_i will all be nonzero for each case.

Let h be arbitrary. Let $h - u_i \geq 0$ and $h - v_i \geq 0$ be the observable horizontal and vertical deviations of $\hat{a}x_i + \hat{b}$ and $\hat{a}x_i^2 + \hat{b}x_i + \hat{c}$ from y_i for each case i. Thus our objective is to minimize h and obtain $min!(max|y_i - \hat{y}_i|)$ which is the optimal value of h. Furthermore, the optimal value of h is the largest possible deviation of $\hat{a}x_i + \hat{b}$ and $\hat{a}x_i^2 + \hat{b}x_i + \hat{c}$ from y_i. The linear programming model that must be solved to approximate $ax_i + b$ in the case II approach is

minimize

$$D(h) = h \tag{15}$$

subject to

$$\begin{cases} ax_1 + b + u_1 - v_1 = y_1, \\ ax_2 + b + u_2 - v_2 = y_2, \\ \vdots \\ ax_{19} + b + u_{19} - v_{19} = y_{19}, \\ h - u_1 \geq 0, \\ h - v_1 \geq 0, \\ h - u_2 \geq 0, \\ h - v_2 \geq 0, \\ \vdots \\ h - u_{19} \geq 0, \\ h - v_{19} \geq 0, \end{cases} \tag{16}$$

and $h, u_1, u_2, ..., u_{19}, v_1, v_2, ..., v_{19} \geq 0$ [4]. To estimate $ax_i^2 + bx_i + c$ in the case II approach we minimize (15) subject to (17).

$$\begin{cases} ax_1^2 + bx_1 + c + u_1 - v_1 = y_1, \\ ax_2^2 + bx_2 + c + u_2 - v_2 = y_2, \\ \vdots \\ ax_{19}^2 + bx_{19} + c + u_{19} - v_{19} = y_{19}, \\ h - u_1 \geq 0, \\ h - v_1 \geq 0, \\ h - u_2 \geq 0, \\ h - v_2 \geq 0, \\ \vdots \\ h - u_{19} \geq 0, \\ h - v_{19} \geq 0, \end{cases} \tag{17}$$

and $h, u_1, u_2, ..., u_{19}, v_1, v_2, ..., v_{19} \geq 0$ [4]. We solve the linear programming models with a simplex algorithm and list the results in the tables below.

Table 4: The optimal values in (Table 4) are the minimized maximum deviation for all four attempts to find the 'best' fit line. $\hat{a}$ and $\hat{b}$ are the parameter estimates for $\hat{y}_i = \hat{a}x_i + \hat{b}$ in each case when the objective is to minimize the maximum deviations. The 'best' fit line is $\hat{y}_i = 0.6250x_i - 0.4000$ since $\tilde{D} = 1.7250$ in the case II solution.

Results of Minimizing the Maximum deviations			
Cases	Optimum	$\hat{a}$	$\hat{b}$
Case I: $a \geq 0$ and $b \geq 0$	1.7920	0.5833	0.0000
Case II: $a \geq 0$ and $b \leq 0$	1.7250	0.6250	-0.4000
Case III: $a \leq 0$ and $b \geq 0$	3.3000	0.0000	4.0000
Case IV: $a \leq 0$ and $b \leq 0$	7.3000	0.0000	0.0000

Table 5: The optimal values in (Table 5) are the minimized maximum deviation for all eight attempts to find the 'best' fit quadratic curve. $\hat{a}$, $\hat{b}$, and $\hat{c}$ are the parameter estimates for $\hat{y}_i = \hat{a}x_i^2 + \hat{b}x_i + \hat{c}$ in each case when the objective is to minimize the maximum deviations. The 'best' fit quadratic curve that is obtained by minimizing the maximum deviations is $\hat{y}_i = 0.1250x_i^2 - 0.6250x_i + 2.4750$ since $\tilde{D} = 1.4750$ in the case VI solution.

Results of Minimizing the Maximum Deviation				
Cases	Optimum	$\hat{a}$	$\hat{b}$	$\hat{c}$
Case I: $a \geq 0$, $b \geq 0$, and $c \geq 0$	1.6000	0.0625	1.0375	0.0000
Case II: $a \geq 0$, $b \geq 0$, and $c \leq 0$	1.6902	0.0174	0.4511	0.0000
Case III: $a \geq 0$, $b \leq 0$, and $c \leq 0$	2.3330	0.0963	0.0000	0.0000
Case IV: $a \leq 0$, $b \geq 0$, and $c \geq 0$	1.79166	0.0000	0.5583	0.0000
Case V: $a \leq 0$, $b \geq 0$, and $c \leq 0$	1.7250	0.0000	0.6250	-0.40000
Case VI: $a \geq 0$, $b \leq 0$, and $c \geq 0$	1.4750	0.1250	-0.6250	2.4750
Case VII: $a \leq 0$, $b \leq 0$, and $c \geq 0$	3.3000	0.0000	0.0000	4.0000
Case VIII: $a \leq 0$, $b \leq 0$, and $c \leq 0$	7.3000	0.0000	0.0000	0.0000

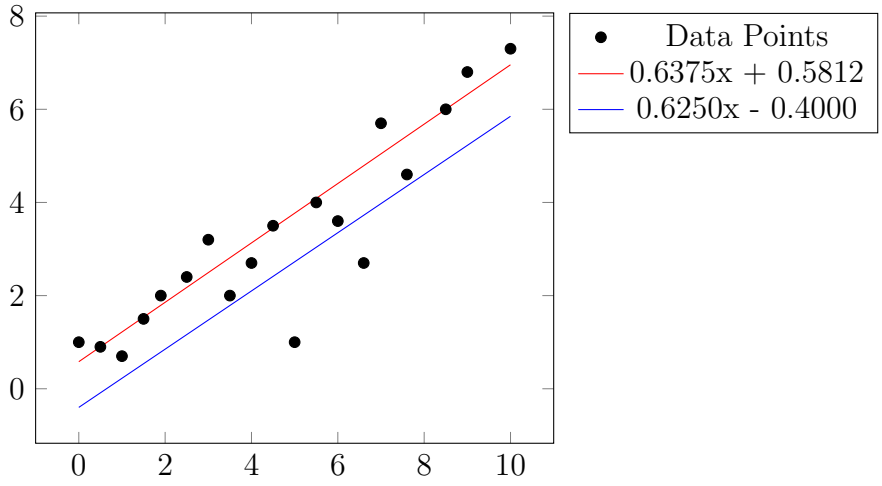

Figure 2: Plot of the 'best' fit lines to the data in (Table 1). The red 'best' fit line in (Fig. 2) was obtained by minimizing (12) subject to (13). The estimates of a and b for the red 'best' line are listed in (Table 2) and correspond to the case I solution. The blue 'best' fit line was obtained by minimizing (15) subject to (16). The estimates of a and b for the blue 'best' fit line are listed in (Table 4) and correspond to the case II solution.

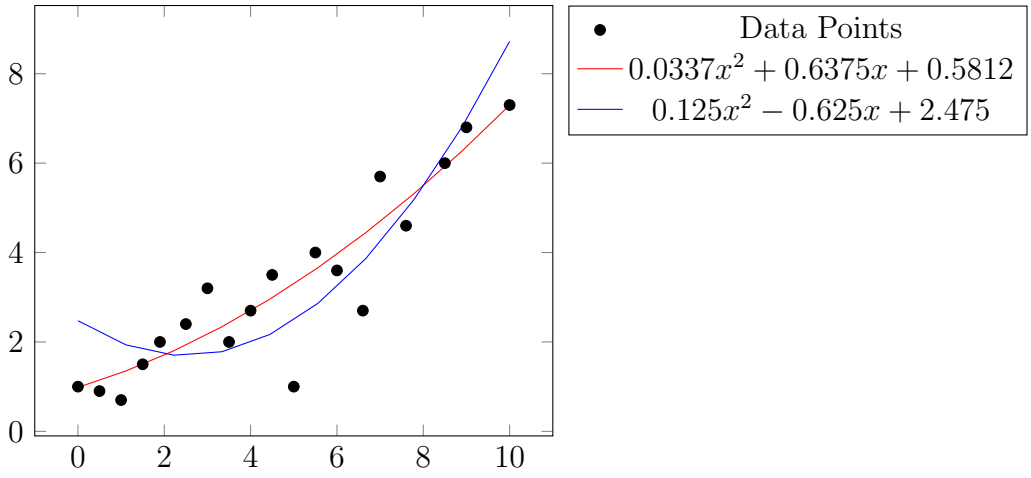

Figure 3: The red 'best' fit quadratic curve in (Fig. 3) was obtained by minimizing (12) subject to (14). The estimates of *a, b,* and *c* for the red 'best' fit quadratic curve are listed in (Table 3) and correspond to the case I solution. The blue 'best' fit quadratic curve in (Fig. 3) was obtained by minimizing (15) subject to (17). The estimates of *a, b,* and *c* for the blue 'best' fit quadratic curve are listed in (Table 5) and correspond to the case VI solution.

Conclusion

We gave examples of optimization in the petroleum, agriculture, and health care industries. Secondly, we gave an in depth but generalized description of a linear programming problem that was written in inequality form. It was also discovered that some two-dimensional linear programs can be solved graphically or by using a simplex algorithm to discover an optimal *n-tuple*. Next, the mechanisms of the simplex algorithm were described in great detail. In the last section of this paper, we saw that goal programming models can also be built and solved to find linear and quadratic curves in a single predictor that are 'best' fits to a data set. Thus, it is easy to see that linear programs have a wide application.

Since linear programming models have wide application, certain circumstances imposed on the objective, constraints or decision variables are common. For example, when the decision variables are constrained to only take integer values, these linear programs are classified as *integer programs* [1, 3, 4]. If they are constrained to only take the values 0 and 1, they are known as *binary integer programs* [1, 3, 4]. In some cases, constraints can take the form $Pr[\sum_{i=1}^{n} a_{m,i}x_i \leq b_m] \geq \beta$ where β is a probability for any particular constraint row m [4]. In other words, these are called *chance constraints*. These are used in a constraint system when we must be $\alpha\%$ sure that $Pr[\sum_{i=1}^{n} a_{m,i}x_i \leq b_m]$ holds at least $\beta\%$ of the time [4]. Finally, if it so happens that the objective contains decision variables that must be raised to a power or functions of decision variables such as e^{x_1}, this program is *nonlinear* [4].

References

[1] Mark M. Meerschaert. *Mathematical modeling.* Academic Press/Elsevier, 2013.

[2] James Stewart. *Calculus.* Brooks/Cole, 2003.

[3] Matthias S. Maier. Linear programming lecture notes. `https://ay15.moodle.umn.edu/course/view.php?id=13393`.

[4] H. P. Williams. *Model building in mathematical programming.* Wiley, 1999.

[5] Robert J. Vanderbei. *Linear programming: foundations and extensions.* Springer, 2014.