1 – Linear Programming

1.1 – Introduction

Linear programming is a method for the optimization of a linear objective function, subject to linear equality and inequality constraints. The constraints typically define a convex polytope in R^n, and a linear programming algorithm finds a point in the polytope where the linear function has a maximum or minimum value, if such a point exists.

The problem of solving a system of linear inequalities has a long history, dating back to Fourier, who in 1827 published an approach for solving them. The first linear programming formulation of a problem in the modern sense was provided by Leonid Kantorovich in 1939. He developed it during World War II to plan expenditure and returns to reduce costs for the army and increase costs incurred by the enemy. Another pivotal figure in the area was George B. Danzig, who in 1947 invented the Simplex Method as a means to efficiently tackle linear programming problems in most cases.

Linear programming is widely used to solve engineering and economics problems. Industries that benefit from the use of this method are as diverse as transportation, energy and manufacturing.

1.2 – Standard Formulation

In linear programming, there are three key terms – linear objective function, decision variables, and constraints. The linear objective function is the n dimensional linear function

$$f(x_1, \dots, x_n) = \ a_1 x_1 + \cdots + \ a_n x_n, \ a_1, \dots a_n \ \in \ R$$

in R^n that needs to be optimized. The decision variables are the $\mathbf{a_i}$ variables that define the linear objective function. The constraints are half spaces in R^n given by

$$bx_1 + \cdots + \ b_n x_n \leq constant, \ b_1, \dots b_n \ \in \ R$$

Linear programming seeks to find a point in the region where the linear objective function is optimized. As an example, Figure 1.2.1(a) shows a linear program with an objective function $\mathbf{750x_1 + 1000x_2}$ in R^2, decision variables $\mathbf{x_1, x_2}$, the associated constraints, the goal being to maximize the function in the region defined by the constraints. In general, linear programs can be expressed in matrix form as in Figure 1.2.1(b), where **c** denotes the components of

the objective function in column vector form, **A** denotes the LHS coefficients of the inequalities, **b** denotes the RHS coefficients of the inequalities, and **x** denotes the column vector of decision variables. Note that the goal in each case is to maximize the objective function. If instead we seek to minimize the objective function, we need only consider the negative of the function, since the point at which this new function is maximized corresponds to the point at which the original function is minimized.

Figure 1.2.1 - Linear Program

(a) Example

$$\begin{aligned} \text{Maximize} \quad & 750x_1 + 1000x_2 \\ \text{Subject to} \quad & \\ & x_1 + x_2 \leq 10 \\ & x_1 + 2x_2 \leq 15 \\ & 4x_1 + 3x_2 \leq 25 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

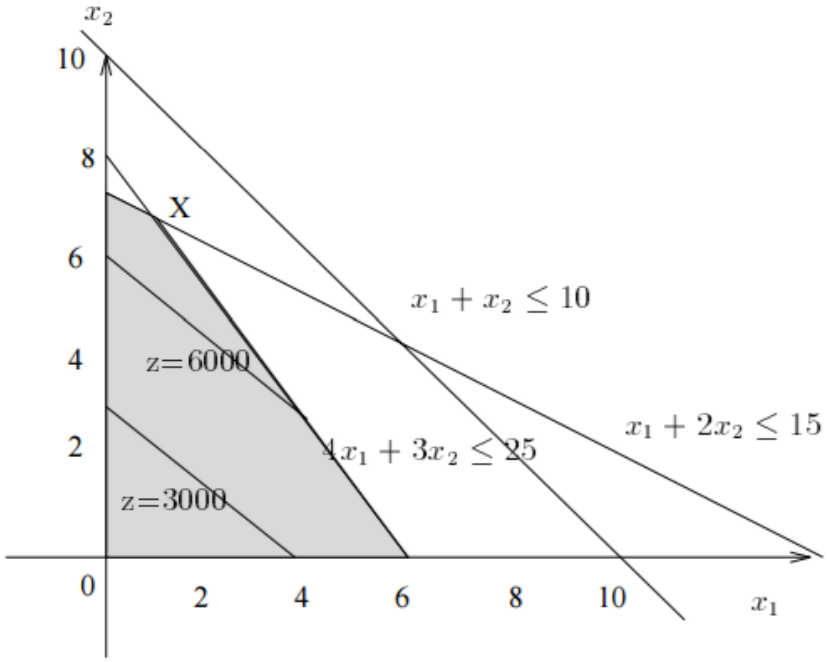
(b) General Form

$$\begin{aligned} \textbf{maximize} \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \textbf{subject to} \quad & A\mathbf{x} \leq \mathbf{b} \\ \textbf{and} \quad & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

1.3 – Graphical Method

While solving linear programs in general requires an algorithmic approach, it is quite easy to solve them in R^2 or R^3 using the graphical method. Consider for example Figure 1.3.1. It graphically solves the linear program of Figure 1.2.1. The region bounded by the constraints is indicated in grey.

Figure 1.3.1 – Graphical Method



Suppose we set the objective function **z** to a constant, and plot the associated iso – lines (so called because **z** is constant along these lines) within the grey region. This is done for **z = 6000**, and **z = 3000**. It is observed that as we move 'rightward', the value of **z** increases, and in particular z attains a maximum at point **X**. Thus this point **X**, which is at the boundary of the bounded region, is the solution to our linear program.

What is interesting is that in general, the solution to any linear program, if it exists, will always be found at the boundary of the region, unless the objective function is a constant, in which case it is found everywhere within the region. To understand this, we must consider the notions of convexity and concavity in R^n. But first, we must be more specific about our 'region'.

1.4 – Convex Polytope

In R^n, a **closed half space** is defined as the linear inequality

$$a_1x_1 + \cdots + a_nx_n \leq b,\ a_1, \ldots a_n \in R$$

A **closed convex polytope** is the set of solutions to a system of such inequalities

$$a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1$$

$$\vdots \qquad\qquad \vdots \qquad \vdots$$

$$a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m$$

Figure 1.2.1 tells us that the constraints of the linear program define a closed convex polytope.

1.5 – Convexity and Concavity

There are three basic definitions that need to be stated.

Definition 1.5.1 – *A subset C of R^n is convex if given any two points u and v in C, the set of convex combinations of u and v,*

$$\{w_t \in R^n \,|\, w_t = ut + (1-t)v, 0 \leq t \leq 1\}$$

are in C.

Definition 1.5.2 – *Let $C \subseteq R^n$ be a convex set, and let f: C → R. Then f(x) is convex on C if*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

for all x, y ∈ C and t ∈ [0, 1]

Definition 1.5.3 – *Let $C \subseteq R^n$ be a convex set, and let f: C → R. Then f(x) is concave on C if*

$$f(tx + (1-t)y) \geq tf(x) + (1-t)f(y)$$

for all x, y ∈ C and t ∈ [0, 1]

It follows trivially that any linear function in R^n is both concave and convex.

We now consider the following two related theorems.

Theorem 1.5.1 – *Let $C \subseteq R^n$ be a convex set, and let f: C → R be a convex function. Let x^* be a local minimizer of f. This implies that there exists a p > 0 such that for every x ∈ C,*

$$\|x - x^*\| \leq p \Rightarrow f(x) \geq f(x^*)$$

Then x^* is a global minimizer of f.

Theorem 1.5.2 – *Let $C \subseteq R^n$ be a convex set, and let f: C → R be a concave function. Let x^* be a local maximizer of f. This implies that there exists a p > 0 such that for every x ∈ C,*

$$\|x - x^*\| \leq p \Rightarrow f(x) \leq f(x^*)$$

Then x^* is a global maximizer of f.

The proofs of these theorems are symmetric, so we prove the first theorem here.

Proof of Theorem 1.5.1 –

Assume that there exists an $\mathbf{x_0}$ such that $\mathbf{f(x_0) < f(x^*)}$. Then for any **t** ∈ (0, 1], we have

$$f\big((1-t)x^* + tx_0\big) \leq (1-t)f(x^*) + tf(x_0) < (1-t)f(x^*) + tf(x^*) = f(x^*) \quad (1)$$

Suppose we choose **t** small enough so that

$$\|(1-t)x^* + tx_0 - x^*\| < p$$

But from (1),

$$f\big((1-t)x^* + tx_0\big) < f(x^*)$$

Then $\mathbf{(1-t)x^* + tx_0}$ is a new minimizer of **f.** This contradicts the fact that $\mathbf{x^*}$ is a local minimizer of **f**. Hence there can exist no such $\mathbf{x_0}$, and we have our result.

In summary, if we find a local maxima or minima for our linear objective function, since the function is both concave and convex, we have found global maxima and minima for the function.

We must now demonstrate that for a non – constant linear objective function, the optima will always be found at the boundary of the closed convex polytope.

1.6 – Maximum and Minimum Principle

We reconsider the definitions of convex and concave functions. The definition of convexity tells us that for t ∈ (0,1),

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \leq \max\big(f(x), f(y)\big)$$

So when you evaluate the function at the convex combination of two points, the value is not greater than the max of values of the function at the two points.

Now every interior point of our closed convex polytope can be expressed as convex combinations of other interior points. Moreover, given an interior point, every other interior point can be used in the construction of convex combinations for the given interior point. Thus, no interior point can be a maximizer of the objective function since in that case every interior point would be a maximizer, and the objective function would be a constant. The only points exempt from this are boundary points of the polytope that cannot be expressed as convex combinations. In particular, these boundary points must be 'corners' or 'vertices' of the polytope since if they lie along the 'flat' (half space) sides, they could be expressed as convex combinations of other boundary points. This is the maximum principle.

It should be noted in hindsight that it was crucial to define the polytope as being closed, since this has allowed us to work with its boundary points. If we instead used strict inequalities in its construction, we would have an open polytope, restricting access to its boundary points.

The minimum principle is defined and proved in a symmetric manner, using concave functions instead of convex functions.

Thus, our linear objective function will always have its optimum value at the 'vertices' of the closed convex polytope. These vertices are called basic solutions.

There are circumstances however, when these solutions may not exist.

1.7 – Existence of Solutions

It is not necessary for solutions to the linear program to exist. For example, inconsistent constraints like **x ≥ 2** and **x ≤ 1** cannot jointly be satisfied.

Secondly, while the polytope is closed, it need not be bounded. If it is unbounded in the direction of the gradient of the objective function (the gradient is the vector or coefficients of the objective function), it may have no optimal value. This is because the gradient vector can be scaled by setting all decision variables equal to a constant. This can be done indefinitely since the region is unbounded, and the function would attain more extreme values.

Linear programs with no solutions are said to be 'infeasible'.

2 – Simplex Method

2.1 – Standard Formulation & Simplex Tableau

Prior to the application of the simplex method, it is necessary to express the linear program in the form shown in Figure 2.1. This can be achieved by following the procedure –

Figure 2.1 – Linear Program In Standard Form

$$\begin{aligned} & \text{Max } c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ \text{subject to } & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ & \qquad \ldots \qquad \ldots \qquad \ldots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \\ & x_1 \geq 0, \ldots x_n \geq 0 \end{aligned}$$

- Given objective function **z**, if the problem is **min z**, convert it to **max –z**
- If the constraint is an inequality, convert it to an equality by introducing non negative slack variables $\mathbf{s_i}$ in the following manner –

$$a_{i1}x_1 + \cdots + a_{in}x_n \leq b_i \rightarrow a_{i1}x_1 + \cdots + a_{in}x_n + s_i = b_i$$

$$a_{i1}x_1 + \cdots + a_{in}x_n \geq b_i \rightarrow a_{i1}x_1 + \cdots + a_{in}x_n - s_i = b_i$$

- If some variable **x** is unrestricted, replace it in all equation with $\mathbf{x_i} - \mathbf{x_j}$ with both $\mathbf{x_i}$ and $\mathbf{x_j}$ non negative.

Having done this, if we as in Figure 1.2.1, assume that the constraints can be expressed in the matrix form Ax = b, we formulate the simplex tableau as in Figure 2.2.

Figure 2.2 – Simplex Tableau

$$\begin{bmatrix} 1 & -\mathbf{c}^T & 0 \\ 0 & \mathbf{A} & \mathbf{b} \end{bmatrix}$$

The key idea in this simplex tableau is that each column is associated with a variable in the linear program. An example makes this clearer.

In Figure 2.3, the linear program in (a) has been converted to the required form in (b) with the introduction of the slack variables $\mathbf{x_3}$ and $\mathbf{x_4}$. It is then expressed in the simplex tableau form in (c).

Figure 2.3 - Simplex Tableau Formulation

(a)
$$\begin{array}{lrrcl} max & x_1 & +x_2 & & \\ & 2x_1 & +x_2 & \leq & 4 \\ & x_1 & +2x_2 & \leq & 3 \\ & x_1 \geq 0, & x_2 \geq 0 & & \end{array}$$

(b)
$$\begin{array}{lrrrrcl} \max & x_1 & +x_2 & & & & \\ & 2x_1 & +x_2 & +x_3 & & = & 4 \\ & x_1 & +2x_2 & & +x_4 & = & 3 \\ & x_1 \geq 0, & x_2 \geq 0 & x_3 \geq 0, & x_4 \geq 0 & & \end{array}$$

(c)
$$\begin{array}{lrrrrcll} z & -x_1 & -x_2 & & & = & 0 & \text{Row } 0 \\ & 2x_1 & +x_2 & +x_3 & & = & 4 & \text{Row } 1 \\ & x_1 & +2x_2 & & +x_4 & = & 3 & \text{Row } 2 \end{array}$$

*The key observation that needs to be made is that by converting the system of constraints to a system of linear equations, we can apply elementary row operations without modifying the solution set. Note that the addition of the objective function to this system does not make any difference to the solution set since the variable **z** is unbounded.*

2.2 – Basic Solutions

There are two kinds of variables in the simplex tableau – basic and non-basic. Basic variables are the variables that feature only once in their respective columns. All other variables are non-basic. For example, in Figure 2.3(c), **z**, $\mathbf{x_3}$ and $\mathbf{x_4}$ are basic, and $\mathbf{x_1}$ and $\mathbf{x_2}$ are non-basic. A basic solution is obtained by setting all non-basic variables to zero, and basic variables to the value on the RHS of the equation.

In Section 1.6 it was stated that the vertices of the polytope are the basic solutions of the linear program, and that for a non – trivial objective function, the optimum would be found at one of these vertices.

What is important is that the basic solutions discussed in the context of the simplex tableau and the polytope are synonymous. Hence, in the simplex method, the idea is to apply elementary row operations to move from basic solution (vertex) to basic solution (vertex), until the optimum is found.

There are certain rules however, that need to be followed in moving from basic solution to basic solution.

2.3 – Rules of Implementation

We continue working with the example in Figure 2.3. The basic solution is $\mathbf{x_1 = 0}$, $\mathbf{x_2 = 0}$, $\mathbf{x_3 = 4}$, $\mathbf{x_4 = 3}$, and $\mathbf{z = 0}$.

The first question that needs to be asked is whether the objective function **z** can be increased by increasing the value of one of the variables $\mathbf{x_i}$ in Row 0. This is possible if the coefficient of the variable is negative. This leads to the first rule of the simplex method –

Rule 1 *If all variables have a nonnegative coefficient in Row 0, the current basic solution is optimal. Otherwise, pick a variable x_i with a negative coefficient in Row 0.*

The variable $\mathbf{x_i}$ selected is called the *entering variable,* and the column to which it belongs is called the *pivot column*. Suppose $\mathbf{x_1}$ is selected. Our next step is to pivot, in order to make the non-basic variable $\mathbf{x_1}$ basic. To do this, we must select a *pivot element*. This is selected using **Rule 2** of the simplex method. To provide intuition, suppose we select the element in

Row 1 $2x_1$ as our pivot element. Applying elementary row operations, we obtain the tableau in Figure 2.4(a). The basic solution is $z = 2$, $x_1 = 2$, $x_2 = 0$, $x_3 = 0$, $x_4 = 1$.

Figure 2.4 - Pivoting

(a)

$$
\begin{array}{llllll}
z & -\frac{1}{2}x_2 & +\frac{1}{3}x_3 & & = 2 & \text{Row 0} \\
x_1 & +\frac{1}{2}x_2 & +\frac{1}{2}x_3 & & = 2 & \text{Row 1} \\
 & \frac{3}{2}x_2 & -\frac{1}{2}x_3 & +x_4 & = 1 & \text{Row 2}
\end{array}
$$

(b)

$$
\begin{array}{llllll}
z & +x_2 & & +x_4 & = 3 & \text{Row 0} \\
 & -3x_2 & +x_3 & -2x_4 & = -2 & \text{Row 1} \\
x_1 & +2x_2 & & +x_4 & = 3 & \text{Row 2}
\end{array}
$$

Suppose instead we used x_1 in **Row 2** as the pivot element. Applying elementary row operations, we obtain the tableau in Figure 2.4(b). The basic solution is $z = 3$, $x_1 = 3$, $x_2 = 0$, $x_3 = -2$, $x_4 = 0$.

The pivot element that must be used is $2x_1$ since using x_1 leads to negative, and hence infeasible values for the other variables. To determine ahead of time which pivot element would lead to a feasible solution, we need only determine the row with smallest positive ratio

$$\frac{RHS}{Entering\ Variable\ Coefficient}$$

The variable in this row of the pivot column would serve as the pivot variable. This leads to **Rule 2** of the simplex method.

Rule 2 *For each Row i, where there is a strictly positive 'entering variable coefficient', compute the ratio of the RHS to the 'entering variable coefficient'. Choose the pivot row as being the one with the minimum ratio.*

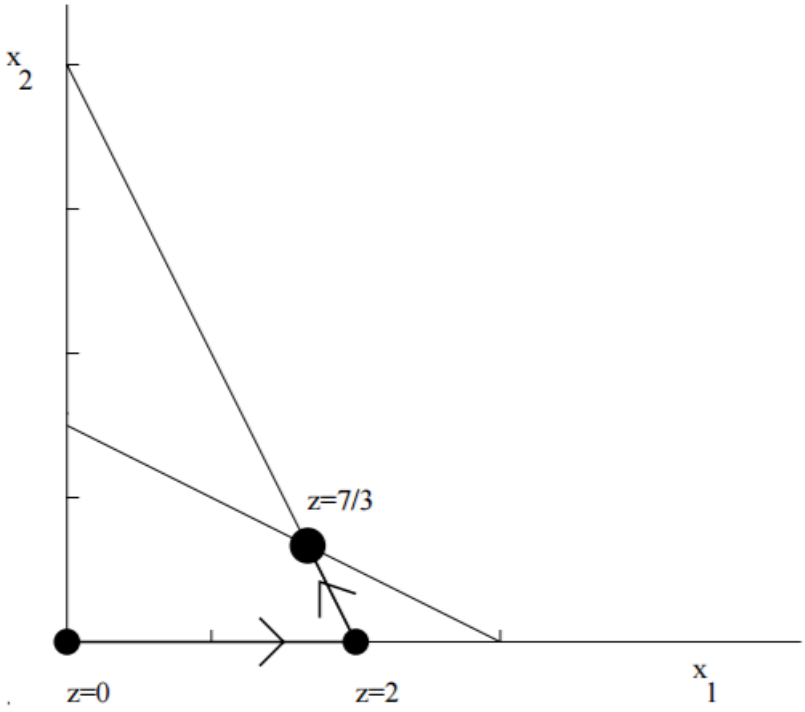
These two rules of the simplex method are applied alternately until we have an optimal solution. Figure 2.5 summarizes the implementation of these rules to obtain the solution $z = 7/3$, $x_1 = 5/3$, $x_2 = 2/3$, $x_3 = 0$, $x_4 = 0$.

Figure 2.5 - Complete Solution

z	x_1	x_2	x_3	x_4	RHS	Basic solution	
1	−1	−1	0	0	0	basic	$x_3 = 4 \quad x_4 = 3$
0	2	1	1	0	4	nonbasic	$x_1 = x_2 = 0$
0	1	2	0	1	3	$z = 0$	
1	0	$-\frac{1}{2}$	$\frac{1}{2}$	0	2	basic	$x_1 = 2 \quad x_4 = 1$
0	1	$\frac{1}{2}$	$\frac{1}{2}$	0	2	nonbasic	$x_2 = x_3 = 0$
0	0	$\frac{3}{2}$	$-\frac{1}{2}$	1	1	$z = 2$	
1	0	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{7}{3}$	basic	$x_1 = \frac{5}{3} \quad x_2 = \frac{2}{3}$
0	1	0	$\frac{2}{3}$	$-\frac{1}{3}$	$\frac{5}{3}$	nonbasic	$x_3 = x_4 = 0$
0	0	1	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	$z = \frac{7}{3}$	

Now Figure 2.6 shows the graphical interpretation of this method. As stated in Section 2.2, we have moved from vertex to vertex, until arriving at the optimum solution.

Figure 2.6 - Graphical Interpretation



1 Excel's Implementation: Karmarkar's Method

1.1 Overview and Intuition

Though powerful, the Simplex Algorithm is not a polynomial-time algortihm. If we start from a particular vertex, then we may end up visiting all the vertices of the feasible region before reaching the solution. This worst-case scenario has exponential running time since the number of vertices could be exponential in number. In 1984, a 28-year-old mathematician named Narendra Karmarkar developed a polynomial-time algorithm to solve LP problems. The Solver tool in Excel implements the Karmarker method.

We begin by exploring the intuition behind Karmarkar's method. A standard LP problem can be geometrically interpreted as a geodesic dome. In two dimensions, the feasible region is a plane; the addition of each new variable adds another dimension to the feasible region. Beyond three dimensions, this is impossible to visualize graphically. Therefore, for an n-dimensional LP problem, a geodesic dome serves as a useful interpretation of the geometry of the problem. Each of the dome's corners serves as a possible solution. With the simplex method, the algorithm lands on one corrner and inspects it; it then visits adjacent corners to find a better answer. If it finds a better answer, then it proceeds in that direction and performs another iteration. The procedure is repeated until the program no longer finds a better solution.

Karmarkar's method employs the same intuition, but follows a radically different tactic. Rather than initialize to a conrner, the algorithm starts from a point within the feasible region. From this interior point, the feasible region is transformed and projected to reconfigure its shape. Suppose that the LP problem exists in the x-space. The transformation T then translates the problem into the y-space, and importantly, the the current point x_k is transformed in such a way that it is in the center of the feasible region in the y-space. This guarantees that the improvement in the optimal solution will be significant. Then, the method determines which direction the solution lies in and makes a step of length l in that direction. The region is then transformed into its original configuration, and the program uses the new point as the interior point for the next iteration.

1.2 Assumptions

Karmarkar's algorithm follows three specific assumptions. The first assumption is standard, namely that the lienar program has a strictly feasible point, and that the set of optimal points is bounded. The second assumption is that the linear program has a special, "canonical" form. Specifically:

$$\text{minimize} \quad z = c^T x \tag{1}$$
$$\text{subject to } Ax = 0 \tag{2}$$
$$a^T x = 1 \tag{3}$$
$$x \geq 0 \tag{4}$$

Note that this assumption isn't restrictive, since any LP problem can be written in such a form. If we consider a problem in standard, Simplex form:

$$\text{minimize} \quad c'^T x \tag{5}$$
$$\text{subject to } A'x' = b \tag{6}$$
$$x' \geq 0 \tag{7}$$

as assume that x' has dimension $n-1$, we can convert this formulation into canonical form by introducing a new variable, $x_n = 1$. Then, we can write the problem as:

$$\text{minimize} \quad z = c'^T x' \tag{8}$$
$$\text{subject to } A'x' - bx_n = 0 \tag{9}$$
$$x_n = 1 \tag{10}$$
$$x' \geq 0 \tag{11}$$

Here, $A = [A', -b]$, $x = [x', x_n]^T$, and a is the n^{th} unit vector e_n. The third assumption is that the value of the objective at the optimum is known and equal to 0. This assumption is unlikely to hold, and indeed it is possible to adapt the method to solve problem swhere the optimal objective value is unknown.

1.3 Mathematical Formulation

We now explore the mathematical formulation of Karmarkar's algorithm. During each iteration, the algorithm performs four steps. First, the problem is transformed via a projective transformation. Then, the projected steepest-descent direction is computed. Next, a step is taken in that direction, and finally, the resulting point is mapped back to the original space via the inverse transformation. The point serves as the beginning of the next iteration. The projective transformation T is defined as follows:

$$y = T(x) = \frac{X^{-1}x}{e^T X^{-1} x}$$

where $X = \text{diag}(x_k)$. The transformation maps the current point x_k to $e/n = (1/n, ..., 1/n)^T$. The corresponding inverse transformation T^{-1} is defined as follows:

$$x = T^{-1}(y) = \frac{Xy}{e^T Xy}$$

Recall that the original linear program was in homogenous form. Substituting the inverse transformation into the original program yields a program in the y-space. That is, the new problem P' is defined as follows:

$$\text{minimize} \quad z = \frac{c^T Xy}{e^T Xy} \tag{12}$$
$$\text{subject to } AXy = 0 \tag{13}$$
$$e^T y = 1 \tag{14}$$
$$y \geq 0 \tag{15}$$

Importantly, P' is not a linear program. However, since we assumed that the value of the objective at the optimum is 0 in the $x - space$, it is correspondiy true in the y-space. Thus, the objective can be rewritten as: minimize$c^T Xy$. The constraints remain the same.

We now compute the steepest-descent direction in the y-space. We can denote the constraint matrix as:

$$\mathrm{B} = \begin{bmatrix} AX \\ e^T \end{bmatrix}$$

The corresponding orthogonal project matrix is $P_B = I = B^T(BB^T)^{-1}B$. And, since $(AX)e = Ax_k = 0$,

$$P_B = P' - \frac{1}{n}ee^T$$

The project steepest-descent direction is $\Delta y = -P_B c^T X$. Then, using the relation that $e^T c^T X = c^T x_k$, the steepest descent direction is:

$$\Delta y = -P' c^T X + \frac{c^T x_k}{n} e$$

Starting from e/n, we now take a step of length α along the projected steepest-descent direction. Our choice of α is importnat. Any step length less than α_{max}, the step to the boundary, fulfills non-negativity constraints, but this alone does not guarantee polynomial complexity. Karmarkar determined that a suitable measure for $\alpha = (1/3)(n(n-1))^{-1/2}$. Therefore:

$$y_{k+1} = \frac{e}{n} + \frac{(n(n-1))^{-1/2}}{3}\Delta y$$

The final step of the iteration is the map y_{k+1} back to the x-space, which yields the new estimate:

$$x_{k+1} = \frac{Xy_{k+1}}{e^T X y_{k+1}}$$

2 MATLAB's Optimization Toolbox: Interior Point Legacy Algorithm

2.1 Linprog

MATLAB's linear programming capabilities are programmed into the command **linprog**. The command allows the user to specify which of three algorithms it wishes to use to solve the LP problem: the interior point legacy algorithm, the interior point algorithm, and the dual simplex method. The default algorithm is the interior point legacy algorithm. We now explore the mathematical properties of this algorithm.

2.2 Duality in Linear Programming

To-do; this information is important in order to understand MATLAB's LP algorithm.

2.3 Interior Point Legacy Algorithm

2.3.1 Preprocessing

The default interior-point-legacy method is based on LIPSOL, which is a variant of Mehrotra's predictor-corrector algorithm, a primal-dual interior point method.

The algorithm begins by applying a series of preprocessing steps. These preprocessing steps function to remove redundancies and simplify constraints. The tasks performed during this step include[1]:

- If any variables have equal upper and lower bounds, check for feasibility, and then fix and remove the variables
- If any linear inequality constraint involves just one variable, check for feasibility, and change the linear constraint to a bound
- Check if any linear equality constraint involves just one variable. If so, check for feasibility, and then fix and remove the variable
- Check if any linear constraint matrix has zero rows. If so, check for feasibility and delete the rows
- Change any linear inequality constraints to linear equality constraints by adding slack variables

If the algorithm detects an infeasible or unbounded problem, then it halts and returns an appropriate error message. Alternatively, if the algorithm arrives at a single feasible point, this point represents the solution.

2.3.2 Primal and Dual Problem

After preprocessing (also referred to as presolve), the problem has the form:

$$\begin{aligned} \text{minimize} \quad & f^T(x) && (16)\\ \text{subject to } & Ax = b && (17)\\ & 0 \leq x \leq u && (18) \end{aligned}$$

With the addition of slack variables s, the inequality constraits can be written as equalities. That is, the problem can be re-written as:

$$\begin{aligned} \text{minimize} \quad & f^T(x) && (19)\\ \text{subject to } & Ax = b && (20)\\ & x + s = u && (21)\\ & x \geq 0, s \geq 0 && (22) \end{aligned}$$

[1]MATLAB Documentation. Linear Programming Algorithms.

This is referred to as the primal problem, where x consists of the primal variables and s consists of the primal slack variables. The dual problem can also be specified:

$$\text{maximize } b^T y - u^T w \tag{23}$$

$$\text{subject to } A^T y - w + z = f \tag{24}$$

$$z \geq 0, w \geq 0 \tag{25}$$

In the dual problem, y and w consist of the dual variables and z consists of the dual slacks. The linear program is therefore defined by both the primal problem and the dual problem. The optimality constraints for the linear program are:

$$F(x, y, z, s, w) = \begin{bmatrix} Ax - b \\ x + s - u \\ A^T y - w + z - f \\ x_i z_i \\ s_i w_i \end{bmatrix} = 0$$

where $x_i z_i$ and $s_i w_i$ represent component-wise multiplication.

2.3.3 Primal-Dual Algorithm

The Interior Point Legacy Algorithm is variant of the predictor-corrector algorithm proposed my Mehrotra. It is, by design, a primal-dual algorithm, which means that the primal and dual programs are solved simultaneously. Consider the iterate $v = [x, y, z, s, w]^T$, where $[x, z, s, w] > 0$. The algorithm first computes the prediction and correction direction:

$$\Delta v_p = -(F^T(v))^{-1} F(v)$$
$$\Delta v_c = -(F^T(v))^{-1} F(v + \Delta v_p) - \mu \hat{e}$$

where μ is the centering parameter and $\hat{e}$ is binary vector with the ones corresponding to the quadratic equations in $F(v)$. The algorithm then takes a step of length α in the combined direction:

$$v^+ = v + \alpha(\Delta v_p + \Delta v_c)$$

The step-length parameter α is chosen so that $v^+ = [x^+, y^+, z^+, s^+, w^+]^T$ satisfies $[x^+, z^+, s^+, w^+] > 0$. The algorithm then loops until the iterates converge. The algorithm employs a standard stopping criterion:

$$\frac{\|r_b\|}{\max(1, \|b\|)} + \frac{\|r_f\|}{\max(1, \|f\|)} + \frac{\|r_u\|}{\max(1, \|u\|)} + \frac{|f^T x = b^T y + u^T w|}{\max(1, |f^T x|, |b^T y - u^T w|)} \leq tol$$

where *tol* is some tolerance, $r_b = Ax - b$, $r_f = A^T y - w + z - f$ and $r_u = x + s - u$. These constitute the primal residual, dual residual, and upper-bound feasibility, respectively.

3 Further Material for Final Draft

- Examples of both Karmarker's algorithm and the Primal-Dual Algorithm
- Explanation of duality in linear programming
- Sensitivity