

# Chapter 8

## Linear programming

This is the most important problem of constrained optimization in terms of applications. A lot of effort has been invested over the decades in perfecting methods and codes for Linear Programming (LP) problems. There are special courses devoted to this problem, both in the Department of Mathematics and in the Division of Management Science. Nocedal & Wright [23] devote two numerically mature chapters to it (13 & 14). Here we will be much faster (and necessarily lighter) than any of these sources.

Recall from Example 7.6 that a linear programming problem in *primal form* reads

$$\min_{\mathbf{x}} \quad \mathbf{c}^T \mathbf{x} \quad (8.1a)$$

$$s.t. \quad A\mathbf{x} = \mathbf{b} \quad (8.1b)$$

$$\mathbf{x} \geq \mathbf{0} \quad (8.1c)$$

where  $\mathbf{c}$ ,  $\mathbf{b}$  and  $A$  are given. The  $m \times n$  matrix  $A$  typically has more columns than rows. Assume it has a full row rank. Assume also (for simplicity) that the feasibility set  $\Omega$  is nonempty and that there is a bounded solution for the problem.

The solution(s) of this optimization problem are known to equivalently satisfy the KKT conditions, known in the present context as the *complementarity conditions*

$$A^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \quad (8.2a)$$

$$A\mathbf{x} = \mathbf{b} \quad (8.2b)$$

$$X\mathbf{S}\mathbf{e} = \mathbf{0} \quad (8.2c)$$

$$\mathbf{x} \geq \mathbf{0}, \mathbf{s} \geq \mathbf{0} \quad (8.2d)$$

where  $X = \text{diag}\{x_1, \dots, x_n\}$ ,  $S = \text{diag}\{s_1, \dots, s_n\}$ ,  $\mathbf{e} = (1, \dots, 1)^T$ .

Although we will not use this directly in our algorithms, let us mention also the important *dual form* of LP,

$$\max_{\boldsymbol{\lambda}} \quad \mathbf{b}^T \boldsymbol{\lambda} \quad (8.3a)$$

$$s.t. \quad A^T \boldsymbol{\lambda} \leq \mathbf{c}. \quad (8.3b)$$

Multiplying a transposed (8.3b) by  $\mathbf{x}$  and using the primal equalities (8.1b) we see that for any feasible  $\mathbf{x}$  for the primal (8.1) and any feasible  $\boldsymbol{\lambda}$  for the dual (8.2),

$$\mathbf{c}^T \mathbf{x} \geq \boldsymbol{\lambda}^T A \mathbf{x} = \mathbf{b}^T \boldsymbol{\lambda}.$$

Moreover, at the optimum  $\mathbf{c}^T \mathbf{x}^* = \mathbf{b}^T \boldsymbol{\lambda}^*$  because of (8.2c). Now, the KKT conditions (8.2a), (8.2b), (8.2d) are seen to merely state the feasibility constraints of both primal and dual forms. The additional, optimality requirement is the complementarity (8.2c). A quantitative feeling of how far a given pair of feasible solutions  $\mathbf{x}$  and  $\boldsymbol{\lambda}$  are from being optimal is given by the *duality gap*

$$\mu = \frac{1}{n} \mathbf{x}^T (\mathbf{c} - A^T \boldsymbol{\lambda}) = \frac{1}{n} \mathbf{x}^T \mathbf{s}.$$

Using the techniques explored in Section 7.3 we can imagine using the equality constraints (8.1b) to express  $m$  variables  $\mathbf{x}_B$  in terms of the other  $n - m$  variables  $\mathbf{x}_N$ . Only  $n$  linear inequality constraints then remain. The feasibility set of the problem (8.1) is then a *polytope* (an intersection of half spaces) in  $n - m$  dimensions. Moreover, since the level sets of  $f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$  are hyperplanes, the situation is as depicted in Figure 8.1. Clearly the minimum

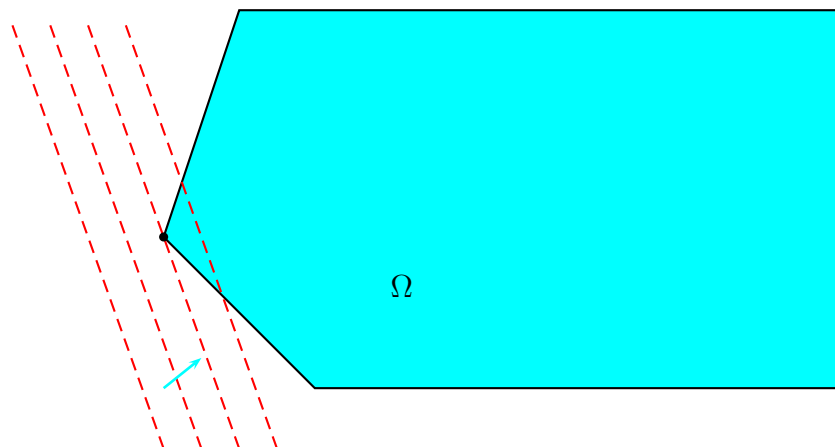


Figure 8.1: LP feasibility region and objective level sets (hyperplanes).

always occurs at a vertex of this polytope.<sup>1</sup>

Each vertex of this feasibility polytope is characterized by picking a set of  $m$  linearly independent columns of  $A$ , forming from them the nonsingular basis matrix  $B$  as described in Section 7.3, and setting

$$\mathbf{x}_N = \mathbf{0}, \text{ hence } \mathbf{x}_B = B^{-1}\mathbf{b} \geq \mathbf{0}.$$

This is called a *basic feasible solution*. We can write  $\mathbf{x} = P \begin{pmatrix} B^{-1}\mathbf{b} \\ \mathbf{0} \end{pmatrix}$ , where  $P$  is a permutation matrix indicating which columns of  $A$  were chosen for the basis  $B$ . Instead of actually forming  $P$ , of course, we just keep track of the indices of the columns forming  $B$  is a set  $\mathcal{B}$ .

### Example 8.1

Let  $m = 2$ ,  $n = 5$ ,

$$A = \begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 0 & -2 & -3 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4 \\ 2 \end{pmatrix}.$$

There are  $\frac{5!}{2!3!} = 10$  different ways to pick two out of the five columns of  $A$  to form a  $2 \times 2$  matrix  $B$ . For instance, picking the second and fifth columns yields

$$B = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}, \quad \text{hence}$$

$$\mathbf{x}_B = B^{-1}\mathbf{b} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

The matrix

$$P^T = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

---

<sup>1</sup>In a special situation the objective level sets are parallel to a face of the polytope, so there are several optimal vertices. Then their linear combinations also are optimal. Thus, not every minimizer is necessarily at a vertex, although at least one always is, unless there is no solution at all.

(which we would never actually form in practice) indicates that the first column of  $B$  is the second of  $A$  and the second column of  $B$  is the 5th of  $A$ . Alternatively,  $\mathcal{B} = \{2, 5\}$ . Then the corresponding basic feasible solution  $\mathbf{x}$  is

$$\mathbf{x} = P \begin{pmatrix} 1 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 2 \end{pmatrix}.$$

◆

The resulting  $\mathbf{x}$  is feasible if  $\mathbf{x}_B \geq \mathbf{0}$  (componentwise). The problem then becomes *combinatorial*: pick the best basic feasible solution (in terms of minimizing  $f$ ) out of possibly as many as  $\binom{n}{m} = \frac{n!}{m!(n-m)!}$  candidates.

This sudden shift to discrete mathematics leads to a meeting of worlds and disciplines that normally exist and thrive separately.

Of course, despite the finiteness of the problem an explicit enumeration of all the exponentially many vertices is out of the question for non-baby problems.

## 8.1 Simplex method

This method was proposed by G.B. Dantzig in the late 1940's, and marks the start of the modern era of optimization. It is an active set method, where we move from one vertex on  $\partial\Omega$  to its neighbor in order to reduce the objective value  $f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$  towards its minimum.

Suppose that at iteration  $k$  the current vertex (i.e., basic feasible solution) is given by  $\mathcal{B}$ , i.e.,

$$\mathbf{x}_B = B^{-1}\mathbf{b}, \quad \mathbf{x}_N = \mathbf{0}, \quad B = [\mathbf{a}_j]_{j \in \mathcal{B}},$$

satisfying  $\mathbf{x}_B \geq \mathbf{0}$ . Here and below,  $\mathbf{a}_j$  denotes the  $j$ th column of  $A$ . Of course,  $\mathcal{B} = \mathcal{B}_k$ , and hence all the rest, depend on the iteration counter  $k$ , but we internalize this to simplify notation. Note that (8.2b) is satisfied by this basic feasible solution.

For the slack variables vector appearing in (8.2) we can write in an analogous notation

$$\mathbf{s} = P \begin{pmatrix} \mathbf{s}_B \\ \mathbf{s}_N \end{pmatrix},$$

and observe that the complementary slackness condition (8.2c) is satisfied by setting the basic slack variables to zero,  $\mathbf{s}_B = \mathbf{0}$ . We can partition the cost vector  $\mathbf{c}$  similarly. Thus, the variables  $\boldsymbol{\lambda}$  are determined by those rows of (8.2a) which correspond to the basis: we require

$$B^T \boldsymbol{\lambda} = \mathbf{c}_B, \quad (8.4a)$$

which yields  $\boldsymbol{\lambda}$ . The non-basic slack variables  $\mathbf{s}_N$  are then determined from the rest of the equations in (8.2a),

$$\mathbf{s}_N = \mathbf{c}_N - N^T \boldsymbol{\lambda}. \quad (8.4b)$$

Now, if  $\mathbf{s}_N \geq \mathbf{0}$  in (8.4b) then all conditions in the KKT equations (8.2) are satisfied and we therefore must be at a minimum. Otherwise, we continue, where there must be some nonbasic  $q$  such that  $s_q < 0$ .

It can be shown that increasing  $x_q$  would yield a reduction in  $f(\mathbf{x})$ . So, we bring column  $q$  of  $A$  into the basis. We set  $s_q = 0$  and increase  $x_q$  from its current value of 0, keeping all other nonbasic variables of  $\mathbf{x}_N$  at their current, zero level, and adjusting the basic variables to maintain the equality conditions  $A\mathbf{x} = \mathbf{b}$ . Denoting the updated  $\mathbf{x}$  by  $\mathbf{x}^+$ , etc., we obtain  $B\mathbf{x}_B^+ + \mathbf{a}_q x_q^+ = A\mathbf{x}^+ = \mathbf{b} = B\mathbf{x}_B$ . Hence,

$$\mathbf{x}_B^+ = \mathbf{x}_B - B^{-1} \mathbf{a}_q x_q^+ = \mathbf{x}_B - \hat{\mathbf{a}} x_q^+, \quad \text{where} \quad (8.4c)$$

$$B\hat{\mathbf{a}} = \mathbf{a}_q. \quad (8.4d)$$

We can increase  $x_q^+$  this way until the first currently basic  $x_l^+$  hits 0 in (8.4c), i.e.,

$$x_q^+ = \min_{\hat{a}_j > 0} (\mathbf{x}_B)_j / \hat{a}_j. \quad (8.4e)$$

(The rest are therefore still nonnegative, so feasibility is maintained.) Thus, the next basis matrix  $B_{k+1} = B^+$  is formed from  $B_k = B$  by introducing column  $q$  and removing column  $l$  of  $A$ .

The resulting change in the objective function can be verified to be

$$\mathbf{c}^T \mathbf{x}^+ = \mathbf{c}^T \mathbf{x} + s_q x_q^+. \quad (8.5)$$

Thus, we see that:

- Unless the problem is *degenerate*, i.e., unless there are basic feasible solutions whose basic variables are not all positive, there is a guaranteed decrease in the objective function according to (8.5) when moving from one basic feasible solution to its neighbour. Since there is a finite number of these vertices the algorithm terminates in a finite number of iterations. Note that the solution  $\mathbf{x}^*$  would be obtained *exactly* in the absence of roundoff error, as befits a direct method for a linear problem.
- During an iteration  $k$  there are usually several options for choosing  $q$ . By (8.5) we obviously want to choose  $q$  to maximize  $-s_q x_q^+$ , but that is not simple to achieve. The original Dantzig choice was to select  $q$  for which  $-s_q$  is largest, but that is not the same thing, and for large problems there are better choices; see [23].

### Example 8.2

We continue with Example 8.1 and add the objective

$$\min_{\mathbf{x}} \sum_{j=1}^5 x_j,$$

i.e.,  $\mathbf{c} = (1, 1, 1, 1, 1)^T$ . A first basic feasible solution has already been provided, and we proceed from there with full, gory detail.

$$1. \mathcal{B} = \{2, 5\}, B = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}, \mathbf{x}_B = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, f(\mathbf{x}) = 3.$$

$$\text{Now, (8.4a) yields } \boldsymbol{\lambda} = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}.$$

$$\text{Then by (8.4b), } \mathbf{s}_N = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 & 2 \\ 3 & -2 \\ 2 & -3 \end{pmatrix} \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix} = \begin{pmatrix} -1/2 \\ 1/2 \\ 3/2 \end{pmatrix}.$$

Here  $(\mathbf{s}_N)_1 = -1/2 < 0$ , so the current basic feasible solution is not optimal. There is only one negative  $s_q$ , so the choice of which column

should enter the basis is unique:  $q = 1$ , hence  $\mathbf{a}_q = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$ .

By (8.4d),  $\hat{\mathbf{a}} = \begin{pmatrix} -1/2 \\ 2 \end{pmatrix}$ . Hence, by (8.4e),  $x_1^+ = 2/2 = 1$ . Finally, by (8.4c),  $\mathbf{x}_B^+ = \begin{pmatrix} 1 \\ 2 \end{pmatrix} - \begin{pmatrix} -1/2 \\ 2 \end{pmatrix} \cdot 1 = \begin{pmatrix} 3/2 \\ 0 \end{pmatrix}$ , hence the new basic feasible solution is  $\begin{pmatrix} 3/2 \\ 1 \end{pmatrix}$ .

$$2. \mathcal{B} = \{2, 1\}, B = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}, \mathbf{x}_B = \begin{pmatrix} 3/2 \\ 1 \end{pmatrix}, f(\mathbf{x}) = 2.5.$$

Now, (8.4a) yields  $\boldsymbol{\lambda} = \begin{pmatrix} 1/2 \\ 1/4 \end{pmatrix}$ .

$$\text{Then by (8.4b), } \mathbf{s}_N = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 3 & -2 \\ 2 & -3 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1/2 \\ 1/4 \end{pmatrix} = \begin{pmatrix} 0 \\ 3/4 \\ 1/4 \end{pmatrix}.$$

Here  $\mathbf{s}_N \geq 0$ , so the current basic feasible solution is optimal and we're done:

$$\mathbf{x}^* = (1, 3/2, 0, 0, 0)^T.$$

Note that for the optimal solution found above,  $s_3 = 0$ . Thus, introducing the 3rd column of  $A$  into the basis, while not improving the objective value, will not increase it either. Indeed, the basic feasible solution

$$\mathbf{x}^a = (7/4, 0, 3/4, 0, 0)^T$$

is also optimal (check!). Therefore, for any  $0 < \theta < 1$ , the point

$$\mathbf{x} = \theta \mathbf{x}^* + (1 - \theta) \mathbf{x}^a$$

is optimal even though it is not basic. ◆

Thus, the basic Simplex update formula (8.4) turns out to be very simple, requiring solving linear systems with  $B$  and  $B^T$ . This is usually done with sparse variants of  $LU$ -decomposition. An update formula when changing basis is necessary too. Also needed is a way to obtain a first feasible solution (or determine that there is none). In addition there are techniques to handle

degeneracy and solution unboundedness. For all these and more see Chapter 13 of Nocedal & Wright [23].

The simplex method has been a source of wonder for theoreticians and practitioners alike for decades. Many textbooks have been centered around it. Its performance is usually very good, even though the worst case performance is very poor (*exponential* in  $m$  and  $n$ ). In the 1970's the complexity of the LP problem was considered to be “between P and NP-complete”, a view probably inspired by the behaviour of this dominant method. It was also shown that the simplex method's average performance is linear, rather than exponential in  $n$ , although a user's actual application is rarely average...

In 1978 a polynomially bounded ‘ellipsoid’ algorithm was found by L. Khachian. But it was never made to work competitively in practice (despite relentless efforts), thus adding to the frustration of theoreticians. Moreover, a different computational model taking into account the accuracy limit of the floating point system used (more precisely, “the number of digits in the input”) was needed.

N. Karmarkar's algorithm in 1984 has infused a new life into this field. It was an algorithm like Khachian's in the sense of not being combinatorial and of proving a polynomial bound complexity which depends on the number of digits in the input. But unlike Khachian's it also boasted practical results. M. Wright and others have subsequently discovered that Karmarkar's algorithm was an instance of a barrier method! Such methods were studied extensively in the 1960's, although Karmarkar's proof was not found then. Significant improvements of both interior point methods and simplex methods have followed since.

## 8.2 Interior point method

Much effort has been devoted in the past 20 years to developing interior point methods for linear programming; see, e.g. S. Wright [32, 23]. It appears that the best among such methods consider the primal-dual formulation defined by the complementarity conditions (8.2). Defining the *center path*, depending on a parameter  $\tau$ , by

$$A^T \boldsymbol{\lambda} + \mathbf{s} = \mathbf{c} \quad (8.6a)$$

$$A\mathbf{x} = \mathbf{b} \quad (8.6b)$$

$$XSe = \tau \mathbf{e} \quad (8.6c)$$

$$\mathbf{x} > \mathbf{0}, \mathbf{s} > \mathbf{0} \quad (8.6d)$$

the solution  $\mathbf{x}(\tau)$ ,  $\boldsymbol{\lambda}(\tau)$ ,  $\mathbf{s}(\tau)$  is feasible for  $\tau > 0$ , and methods can be constructed which follow this path to optimality as  $\tau \downarrow 0$ . See Figure 8.2.

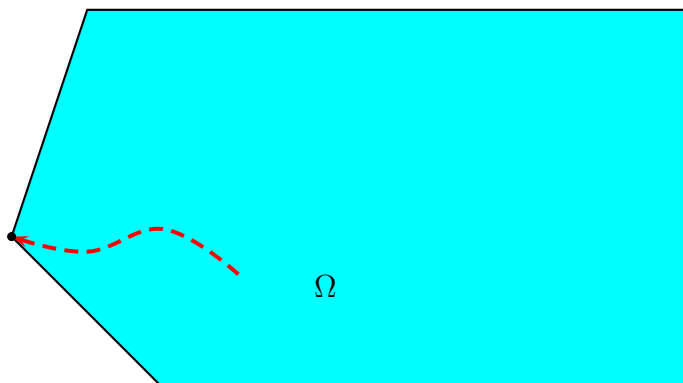


Figure 8.2: Center path in LP feasibility region.

For a current point  $\mathbf{z} = (\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s})$  satisfying (8.6d) set the *duality measure*, or *duality gap*

$$\mu = \frac{1}{n} \mathbf{x}^T \mathbf{s}. \quad (8.7)$$

On the center path, obviously  $\mu = \tau$ , but  $\mu$  is defined also off the center path. A Newton step for the equalities in (8.2)+(8.6) reads

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \\ \delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} \mathbf{c} - A^T \boldsymbol{\lambda} - \mathbf{s} \\ \mathbf{b} - A \mathbf{x} \\ \sigma \mu \mathbf{e} - X S \mathbf{e} \end{pmatrix} \quad (8.8)$$

where  $\sigma \in [0, 1]$  is a *centering parameter*. The value  $\sigma = 1$  yields a step towards the center path with  $\tau = \mu$ , while  $\sigma = 0$  is Newton for (8.2).

After the update we want (8.6d) to hold. A simple update is then to choose  $0 < \alpha \leq 1$  such that  $(x_i + \alpha \delta x_i)(s_i + \alpha \delta s_i) \geq \text{tol } \mu$ ,  $\forall i$ , and set

$$\begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \\ \mathbf{s} \end{pmatrix} = \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \\ \mathbf{s} \end{pmatrix} + \alpha \begin{pmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \\ \delta \mathbf{s} \end{pmatrix}.$$

With this it is possible to show convergence (see §14.4 of [23], or [32]). The proof is not mathematically deep, and it makes N. Karmarkar and many others look a little pompous in perspective. But that's not our concern here.

### 8.2.1 Solving the linear system

Let us write the basic linear algebra system we have to solve above as

$$H \delta \mathbf{z} \equiv \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \\ \delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{r}_\tau \end{pmatrix}. \quad (8.9)$$

We next discuss how to solve this linear system; then we turn into methods for determining (8.9).

1. We can just solve (8.9) directly, ignoring the zeros, if  $m$  and  $n$  are not too large. If iterative methods are required then the usual methods (e.g. GMRES [26, 7]) suggest themselves. Then a routine for calculating  $H\mathbf{v}$  for any given vector  $\mathbf{v}$  will incorporate sparsity.
2. It is also possible to symmetrize  $H$  by multiplying the last block row by  $S^{-1}$ :

$$H^* \delta \mathbf{z} \equiv \begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ I & 0 & S^{-1}X \end{pmatrix} \begin{pmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \\ \delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -S^{-1}\mathbf{r}_\tau \end{pmatrix}. \quad (8.10)$$

The system (8.10) is of size  $2n + m$  and it is symmetric (though indefinite). Algorithms such as MinRes [26, 7] and symmetric QMR [15, 16] may be used to solve the system for the large, sparse case.

3. Dreading to solve the system (8.9) directly, we can eliminate  $\delta \mathbf{s}$  first,

$$\delta \mathbf{s} = -\mathbf{r}_c - A^T \delta \boldsymbol{\lambda}. \quad (8.11a)$$

Then, upon reordering columns in (8.9) and multiplying the last row block by  $-X^{-1}$ , we have

$$\begin{pmatrix} 0 & A \\ A^T & -X^{-1}S \end{pmatrix} \begin{pmatrix} \delta \boldsymbol{\lambda} \\ \delta \mathbf{x} \end{pmatrix} = \begin{pmatrix} -\mathbf{r}_b \\ X^{-1}\mathbf{r}_\tau - \mathbf{r}_c \end{pmatrix}. \quad (8.11b)$$

The system (8.11b) is only of size  $n + m$  and it is symmetric (though indefinite). Algorithms such as MinRes and symmetric QMR [15, 16] may be used to solve the system for the large, sparse case.

Whereas (8.11) appears to be a tad cheaper to solve directly than (8.10), it is not clear to me why it would be better to solve iteratively, especially in the inexact Newton framework.

4. A third alternative for solving the linear system (8.9) is to eliminate also  $\delta \mathbf{x}$  from (8.11). Multiplying the last block row in (8.11b) by  $AS^{-1}X$  and adding the first block row we get

$$AS^{-1}XA^T \delta \boldsymbol{\lambda} = -\mathbf{r}_b + AS^{-1}(\mathbf{r}_\tau - X\mathbf{r}_c). \quad (8.12a)$$

The matrix involved is  $m \times m$  and s.p.d., although there is an apparent source of ill-conditioning near the boundary of the feasibility polytope,  $\partial\Omega$ . A Cholesky decomposition may be used for it. For very large, sparse problems a preconditioned CG algorithm (Section 5.1) suggests itself. Then  $\delta \mathbf{s}$  is obtained from (8.11a) and  $\delta \mathbf{x}$  is obtained from

$$\delta \mathbf{x} = -S^{-1}(X\delta \mathbf{s} + \mathbf{r}_\tau). \quad (8.12b)$$

## 8.2.2 Mehrotra's algorithm

A practical algorithm of the above sort is obtained with Mehrotra's predictor-corrector [22]. The idea can be described as follows.

Assuming a full Newton step ( $\alpha = 1$ ) can be taken we want ideally at the end of the step for (8.6)

$$\begin{aligned} A^T(\boldsymbol{\lambda} + \delta \boldsymbol{\lambda}) + (\mathbf{s} + \delta \mathbf{s}) &= \mathbf{c} \\ A(\mathbf{x} + \delta \mathbf{x}) &= \mathbf{b} \\ (X + \delta X)(S + \delta S)\mathbf{e} &= \tau \mathbf{e} \\ \mathbf{x} > \mathbf{0}, \mathbf{s} > \mathbf{0}. \end{aligned}$$

Here we write  $\tau = \sigma\mu$ , with  $\mu$  defined in (8.7) and  $\sigma$  the centering parameter. There are two questions with the above:

1. How to set  $\sigma$  such that a large step can be taken?
2. How to approximate the curvature term  $\Delta = \delta X \delta S$ ? This is the only nonlinear term here, the rest indeed is precisely a Newton (quasilinearization) step.

Both these questions are settled by a *predictor* step. We set  $\sigma = \sigma_p = 0$ ,  $\Delta_p = 0$ , and solve (8.8). This direction is meant to achieve maximum progress towards the optimum and would be successful if positivity constraints would not prove too restrictive. Denote the result  $\delta \mathbf{x}^p, \delta \mathbf{s}^p$ . For one thing this gives an approximation

$$\Delta_c = \delta X^p \delta S^p \quad (8.13a)$$

for the curvature term. We next figure out how far we can go in the predicted direction while still retaining positivity, by evaluating

$$\alpha_p = \min\left\{1, \min_{\delta x_i < 0} \frac{x_i}{-\delta x_i^p}\right\} \quad (8.13b)$$

$$\beta_p = \min\left\{1, \min_{\delta s_i < 0} \frac{s_i}{-\delta s_i^p}\right\}. \quad (8.13c)$$

If we took the maximum allowed step then the new duality measure would be

$$\mu_p = \frac{1}{m}(\mathbf{x} + \alpha_p \delta \mathbf{x}^p)^T (\mathbf{s} + \beta_p \delta \mathbf{s}^p). \quad (8.13d)$$

If  $\mu_p$  is much smaller than  $\mu$  then great progress can be made by the predictor. If not then a significant centering has to be added to the direction of search. This is captured by choosing

$$\sigma = \left(\frac{\mu_p}{\mu}\right)^3. \quad (8.13e)$$

(Ask not why cubed, said the wise elderly.)

With the value of the centering parameter thus fixed and with the curvature approximated by (8.13a) we next solve for the corrected direction

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{pmatrix} \begin{pmatrix} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \\ \delta \mathbf{s} \end{pmatrix} = \begin{pmatrix} \mathbf{c} - A^T \boldsymbol{\lambda} - \mathbf{s} \\ \mathbf{b} - A \mathbf{x} \\ \sigma \mu \mathbf{e} - (XS + \Delta_c) \mathbf{e} \end{pmatrix}. \quad (8.13f)$$

Note that the same matrix and most of the same right hand side are used for the predictor and the corrector.

Finally, we want the next solution point to be positive too, so the update reads

$$\mathbf{x} = \mathbf{x} + \alpha \delta \mathbf{x}, \quad \alpha = \min\left\{1, \min_{\delta x_i < 0} \frac{(1 - tol)x_i}{-\delta x_i}\right\} \quad (8.13g)$$

$$\mathbf{s} = \mathbf{s} + \beta \delta \mathbf{s}, \quad \beta = \min\left\{1, \min_{\delta s_i < 0} \frac{(1 - tol)s_i}{-\delta s_i}\right\} \quad (8.13h)$$

$$\boldsymbol{\lambda} = \boldsymbol{\lambda} + \beta \delta \boldsymbol{\lambda}. \quad (8.13i)$$

Set, e.g.,  $tol = 0.01$ .

When  $\mu$  gets small, e.g.  $\mu < 0.0001$ , it can be worthwhile to hop to a nearby vertex (basic solution) and check optimality. This can be done as follows:

1. Let  $\mathcal{B} \subset \{1, 2, \dots, n\}$  be the set of  $m$  indices corresponding to largest components of  $\mathbf{x}$ . This is the “suspected optimal basis”.
2. Set

$$\hat{x}_j = 0, \quad j \notin \mathcal{B}, \quad \hat{s}_j = 0 \quad j \in \mathcal{B}.$$

Compose the  $m \times m$  matrix  $B$  out of the  $m$  basic columns of  $A$  and likewise the vector  $\mathbf{c}_B$  out of  $\mathbf{c}$ .

3. Solve

$$B\hat{\mathbf{x}}_B = \mathbf{b}$$

and insert the values of  $\hat{\mathbf{x}}_B$  as the basic values of  $\hat{\mathbf{x}}$  for  $j \in \mathcal{B}$ .

4. Set

$$\begin{aligned} \hat{\boldsymbol{\lambda}} &= B^{-T} \mathbf{c}_B, \\ \hat{\mathbf{s}} &= \mathbf{c} - A^T \hat{\boldsymbol{\lambda}}. \end{aligned}$$

5. If  $\hat{\mathbf{x}} > -\epsilon \mathbf{e}$ ,  $\hat{\mathbf{s}} > -\epsilon \mathbf{e}$  and  $\frac{1}{m} \hat{\mathbf{s}}^T \hat{\mathbf{x}} < \epsilon$  for a very small positive tolerance  $\epsilon$  then an optimal solution has been found in  $(\hat{\mathbf{x}}, \hat{\boldsymbol{\lambda}}, \hat{\mathbf{s}})$ .

Here is my program. It does not do fancy checking for degenerate cases, infeasibility and unboundedness. Still, bearing the history of this field in mind it is amazing how simple it has all finally become.

```
function [x,gap,nbas] = lpm (A,b,c)
%
% function [x,gap,nbas] = lpm (A,b,c)
%
% solve the linear programming problem
% min c^T x   s.t. Ax = b, x >= 0 .
%
% use interior point method as per algorithm 14.3 of Nocedal-Wright
%
% A is l x m,  b is l x 1, c is m x 1.
% return solution x and duality gap (should be close to 0 if all well).
% Also, nbas is the number of hops to check basic solutions
% before optimality is reached.

[1,m] = size(A);
scaleb = norm(b) + norm(A,inf) + 1;
```

```

scalec = norm(c) + norm(A,inf) + 1;
tol = 0.01;
otol = 1-tol;
toln = 1.e-9;
tolp = 1.e-10;
nbas = 0;

% Initial guess
x = ones(m,1);
s = ones(m,1);
y = zeros(1,1);

fprintf('itn      gap      infeas      mu\n')
for it = 1:2*m+10 % iterate, counting pred-cor as one

% duality measure
mu = (x'*s)/m;
% predict correction
[dx,dy,ds] = newt1p(A,b,c,x,y,s,0);

% incorporate positivity into constraints
alfa = 1; beta = 1;
for i=1:m
    if dx(i) < 0
        alfa = min(alfa, -x(i)/dx(i));
    end
    if ds(i) < 0
        beta = min(beta, -s(i)/ds(i));
    end
end

% the would-be duality measure
muaff = ( (x+alfa*dx)' * (s+beta*ds) ) / m;
% centering parameter
sigma = (muaff/mu)^3;

% correct towards center path
smu = sigma * mu;
[dx,dy,ds] = newt1p(A,b,c,x,y,s,smu,dx,ds);

% incorporate positivity into constraints
alfa = 1; beta = 1;

```

```

for i=1:m
    if dx(i) < 0
        alfa = min(alfa, -otol*x(i)/dx(i));
    end
    if ds(i) < 0
        beta = min(beta, -otol*s(i)/ds(i));
    end
end

% update solution
x = x + alfa*dx;
s = s + beta*ds;
y = y + beta*dy;

% check progress
% infeas = norm(b - A*x)/sqrt(l) + norm(c - A'*y - s)/sqrt(m);
infeas = norm(b - A*x)/scaleb + norm(c - A'*y - s)/scalec;
gap = (c'*x - b'*y) / m;
if (infeas > 1.e+12)+(gap < -toln)
    fprintf('no convergence: perhaps no solution')
    return
end
fprintf('%d    %e    %e    %e\n',it,gap,infeas,mu)

if (abs(infeas) < toln)*(abs(gap) < toln)
    return
end

% hop to next basic solution
if gap < tol^2
    nbas = nbas + 1;
    [xx,sortof] = sort(-x);
    [xx,yy,ss] = basln(A,b,c,sortof);
    gap = (c'*xx - b'*yy) / m;
    if (norm(xx+tolp >= 0)^2 > m-1)*(norm(ss+tolp >= 0)^2 > m-1)*(abs(gap) < toln)
        x = xx;
        return
    end
end
end
end

```

```

function [dx,dy,ds] = newt1p(A,b,c,x,y,s,mu,dx,ds)
%
% function [dx,dy,ds] = newt1p(A,b,c,x,y,s,mu,dx,ds)
%
% a Newton step for lp
% Use one of three direct elimination methods
%

method = 3;

[l,m] = size(A);
rc = A'*y + s - c;
rb = A*x - b;
rt = x.*s - mu;
if nargin == 9
    rt = rt + dx.*ds;
end

if method == 1
    rhs = [-rc; -rb; -rt];
    Mat = [zeros(m,m), A', eye(m); ...
           A, zeros(1,1), zeros(1,m); ...
           diag(s), zeros(m,1), diag(x)];
    z = Mat \ rhs;
    dx = z(1:m);
    dy = z(m+1:m+1);
    ds = z(m+1+1:2*m+1);

elseif method == 2
    rhs = [-rb; rt./x - rc];
    Mat = [zeros(1,1), A; ...
           A', -diag(s./x)];
    z = Mat \ rhs;
    dy = z(1:1);
    dx = z(1+1:m+1);
    ds = -rc - A'*dy;

elseif method == 3
    rhs = [-rb + A * ((rt - x.*rc)./s)];
    Mat = A * diag(x./s) * A';
    dy = Mat \ rhs;
    ds = -rc - A'*dy;

```

```

    dx = -(x.*ds + rt)./s;

end

function [x,y,s] = basln(A,b,c,sort)
%
% function [x,y,s] = basln(A,b,c,sort)
%
% given a vector of indices, the first l
% indicate a basis out of A.
% find corresponding basic solution.

[l,m] = size(A);
B = zeros(l,l); cb = zeros(l,1);

% construct basis
for j=1:l
    B(:,j) = A(:,sort(j));
    cb(j) = c(sort(j));
end

xb = B \ b;
x = zeros(m,1);
for j=1:l
    x(sort(j)) = xb(j);
end

y = B' \ cb;

s = c - A'*y;

```

### Example 8.3

Let us revisit the pocket-size instance of Example 8.2. Here we get convergence in 3 iterations (starting from scratch!) The duality gaps of successive iterations are  $\mu_k = (1, 1.1e-1, 2.4e-3, 2.4e-5)$  and there was one hop to the optimal vertex (which turned out to be  $\mathbf{x}^*$ ) from the last interior iterate.  $\blacklozenge$

From my limited experimentation with this algorithm (which in particular does not include inexact-type modifications and iterative solution of linear

systems) it works beautifully. There is no complete theory to support it, though.

**Example 8.4**

For a larger example than the previous one we set  $m = 260$ ,  $n = 570$  and generate a random  $m \times n$  matrix  $A$ . Then we generate two more random (nonnegative)  $n$ -vectors,  $\hat{\mathbf{x}}$  and  $\mathbf{c}$ . Then set  $\mathbf{b} = A\hat{\mathbf{x}}$  and forget about  $\hat{\mathbf{x}}$ . This guarantees that the feasibility region is not empty.

Calling `lpm(A, b, c)` yields in 10 iterations the following sequence of duality gaps

$$\mu_k = (1, 1.4e - 1, 4.7e - 2, 1.8e - 2, 7.4e - 3, 2.3e - 3, \\ 8.6e - 4, 2.9e - 4, 5.8e - 5, 2.0e - 5, 3.9e - 6).$$

There were two unsuccessful hops to nearby vertices (i.e. the obtained vertices did not correspond to the optimal basic feasible solution).

Tightening the tolerance on the duality gap to  $toln = 1.e - 7$  produces two more iterations with  $\mu_{11} = 8.6e - 7$ ,  $\mu_{12} = 1.0e - 7$ . The optimal basic feasible solution is discovered this time after a total of 5 hops to nearby polytope vertices.

Note how much smaller than  $m$  the total number of iterations is.

