7

Affine Scaling Algorithms

Since its introduction in 1984, Karmarkar's projective scaling algorithm has become the most notable interior-point method for solving linear programming problems. This pioneering work has stimulated a flurry of research activities in the field. Among all reported variants of Karmarkar's original algorithm, the *affine scaling* approach especially attracted researchers' attention. This approach uses the simple *affine transformation* to replace Karmarkar's original *projective transformation* and allows people to work on the linear programming problems in standard form. The special simplex structure required by Karmarkar's algorithm is relaxed.

The basic affine scaling algorithm was first presented by I. I. Dikin, a Soviet mathematician, in 1967. Later, in 1985, the work was independently rediscovered by E. Barnes and R. Vanderbei, M. Meketon, and B. Freedman. They proposed using the *(primal) affine scaling algorithm* to solve the (primal) linear programs in standard form and established convergence proof of the algorithm. A similar algorithm, the so-called *dual affine scaling algorithm*, was designed and implemented by I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga for solving (dual) linear programs in inequality form. Compared to the relatively cumbersome projective transformation, the implementation of both the primal and dual affine scaling algorithms become quite straightforward. These two algorithms are currently the variants subject to the widest experimentation and exhibit promising results, although the theoretical proof of polynomial-time complexity was lost in the simplified transformation. In fact, N. Megiddo and M. Shub's work indicated that the trajectory leading to the optimal solution provided by the basic affine scaling algorithms depends upon the starting solution. A bad starting solution, which is too close to a vertex of the feasible domain, could result in a long journey traversing all vertices. Nevertheless, the polynomial-time complexity of the primal and dual affine scaling algorithms can be reestablished by incorporating a logarithmic barrier function on the walls of the positive orthant to prevent an interior solution being "trapped" by the boundary behavior. Along this direction, a third variant, the so-called *primal-dual affine scaling algorithm*, was presented and analyzed by R. Monteiro, I. Adler, and M. G. C. Resende, also by M. Kojima, S. Mizuno, and A. Yoshise, in 1987. The theoretical issue of polynomial-time complexity was successfully addressed.

In this chapter, we introduce and study the abovementioned variants of affine scaling, using an integrated theme of iterative scheme. Attentions will be focused on the three basic elements of an iterative scheme, namely, (1) how to start, (2) how to synthesize a good direction of movement, and (3) how to stop an iterative algorithm.

7.1 PRIMAL AFFINE SCALING ALGORITHM

Let us consider a linear programming problem in its standard form:

$$\text{Minimize} \quad \mathbf{c}^T\mathbf{x} \tag{7.1a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \tag{7.1b}$$

where $\mathbf{A}$ is an $m \times n$ matrix of full row rank, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{x}$ are n-dimensional column vectors.

Notice that the feasible domain of problem (7.1) is defined by

$$P = \{\mathbf{x} \in R^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$$

We further define the *relative interior* of P (with respect to the affine space $\{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}\}$) as

$$P^0 = \{\mathbf{x} \in R^n \mid \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} > \mathbf{0}\} \tag{7.2}$$

An n-vector $\mathbf{x}$ is called an *interior feasible point*, or *interior solution*, of the linear programming problem, if $\mathbf{x} \in P^0$. Throughout this book, for any interior-point approach, we always make a fundamental assumption

$$P^0 \neq \phi$$

There are several ways to find an initial interior solution to a given linear programming problem. The details will be discussed later. For the time being, we simply assume that an initial interior solution $\mathbf{x}^0$ is available and focus on the basic ideas of the primal affine scaling algorithm.

7.1.1 Basic Ideas of Primal Affine Scaling

Remember from Chapter 6 the two fundamental insights observed by N. Karmarkar in designing his algorithm. Since they are still the guiding principles for the affine scaling algorithms, we repeat them here:

(1) if the current interior solution is near the center of the polytope, then it makes sense to move in the direction of steepest descent of the objective function to achieve a minimum value;

(2) without changing the problem in any essential way, an appropriate transformation can be applied to the solution space such that the current interior solution is placed near the center in the transformed solution space.

In Karmarkar's formulation, the special simplex structure

$$\Delta = \{\mathbf{x} \in R^n \mid x_1 + \ldots + x_n = 1, x_i \geq 0, i = 1, \ldots, n\}$$

and its center point $\mathbf{e}/n = (1/n, 1/n, \ldots, 1/n)^T$ were purposely introduced for the realization of the above insights. When we directly work on the standard-form problem, the simplex structure is no longer available, and the feasible domain could become an unbounded polyhedral set. All the structure remaining is the intersection of the affine space $\{\mathbf{x} \in R^n \mid \mathbf{Ax} = \mathbf{b}\}$ formed by the explicit constraints and the positive orthant $\{\mathbf{x} \in R^n \mid \mathbf{x} \geq \mathbf{0}\}$ required by the nonnegativity constraints. It is obvious that the nonnegative orthant does not have a real "center" point. However, if we position ourselves at the point $\mathbf{e} = (1, 1, \ldots, 1)^T$, at least we still keep equal distance from each facet, or "wall," of the nonnegative orthant. As long as the moving distance is less than one unit, any new point that moves from $\mathbf{e}$ remains in the interior of the nonnegative orthant. Consequently, if we were able to find an appropriate transformation that maps a current interior solution to the point $\mathbf{e}$, then, in parallel with Karmarkar's projective scaling algorithm, we can state a modified strategy as follows.

"Take an interior solution, apply the appropriate transformation to the solution space so as to place the current solution at $\mathbf{e}$ in the transformed space, and then move in the direction of steep descent in the null space of the transformed explicit constraints, but not all the way to the nonnegativity walls in order to remain as an interior solution. Then we take the inverse transformation to map the improved solution back to the original solution space as a new interior solution. Repeat this process until the optimality or other stopping conditions are met."

An appropriate transformation in this case turns out to be the so-called *affine scaling transformation*. Hence people named this variant the affine scaling algorithm. Also, because it is directly applied to the primal problems in standard form, its full name becomes the *primal affine scaling algorithm*.

Affine scaling transformation on the nonnegative orthant. Let $\mathbf{x}^k \in R^n$ be an interior point of the nonnegative orthant R^n_+, i.e., $x_i^k > 0$ for $i = 1, \ldots, n$. We define an $n \times n$ diagonal matrix

$$\mathbf{X}_k = \operatorname{diag}(\mathbf{x}^k) = \begin{bmatrix} x_1^k & 0 & \ldots & 0 \\ 0 & x_2^k & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & x_n^k \end{bmatrix} \tag{7.3}$$

It is obvious that matrix $\mathbf{X}_k$ is nonsingular with an inverse matrix $\mathbf{X}_k^{-1}$, which is also a diagonal matrix but with $1/x_i^k$ being its ith diagonal element for $i = 1, \ldots, n$.

The *affine scaling transformation* is defined from the nonnegative orthant R_+^n to itself by

$$\mathbf{y} = T_k(\mathbf{x}) = \mathbf{X}_k^{-1}\mathbf{x} \tag{7.4}$$

Note that transformation (7.4) simply *rescales* the ith component of $\mathbf{x}$ by dividing a positive number x_i^k. Geometrically, it maps a straight line to another straight line. Hence it was named the *affine scaling transformation.* Figure 7.1 illustrates the geometric picture of the transformation in two-dimensional space. Note that for the two-dimensional inequality constraints, such as the case depicted by Figure 7.1, the scaling variables include the slack variables, too. As a matter of fact, each edge of the polygon corresponds to a slack variable being set to zero. However, it is difficult to represent the whole picture in the same figure.

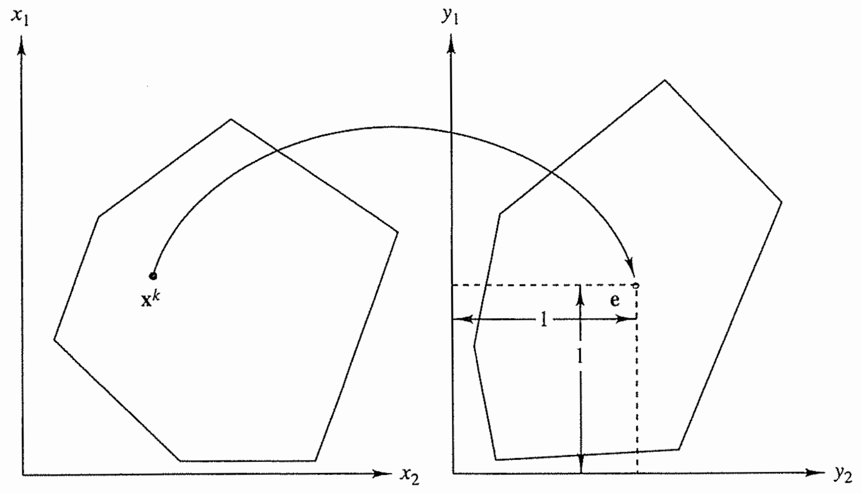

Figure 7.1

The following properties of T_k can be easily verified:

(T1) T_k is a well-defined mapping from R_+^k to R_+^k, if $\mathbf{x}^k$ is an interior point of R_+^n.

(T2) $T_k(\mathbf{x}^k) = \mathbf{e}$.

(T3) $T_k(\mathbf{x})$ is a vertex of R_+^n if $\mathbf{x}$ is a vertex.

(T4) $T_k(\mathbf{x})$ is on the boundary of R_+^n if $\mathbf{x}$ is on the boundary.

(T5) $T_k(\mathbf{x})$ is an interior point of R_+^n if $\mathbf{x}$ is in the interior.

(T6) T_k is a one-to-one and onto mapping with an inverse transformation T_k^{-1} such that

$$T_k^{-1}(\mathbf{y}) = \mathbf{X}_k\mathbf{y} \qquad \text{for each } \mathbf{y} \in R_+^n. \tag{7.5}$$

Primal affine scaling algorithm. Suppose that an interior solution $\mathbf{x}^k$ to the linear programming problem (7.1) is known. We can apply the affine scaling transformation T_k to "center" its image at $\mathbf{e}$. By the relationship $\mathbf{x} = \mathbf{X}_k\mathbf{y}$ shown in (7.5), in the transformed solution space, we have a corresponding linear programming problem

$$\text{Minimize} \quad (\mathbf{c}^k)^T\mathbf{y} \tag{7.1$'$a}$$

$$\text{subject to} \quad \mathbf{A}_k\mathbf{y} = \mathbf{b}, \quad \mathbf{y} \geq \mathbf{0} \tag{7.1$'$b}$$

where $\mathbf{c}^k = \mathbf{X}_k\mathbf{c}$ and $\mathbf{A}_k = \mathbf{A}\mathbf{X}_k$.

In Problem (7.1′), the image of $\mathbf{x}^k$, i.e., $\mathbf{y}^k = T_k(\mathbf{x}^k)$, becomes $\mathbf{e}$ that keeps unit distance away from the walls of the nonnegative orthant. Just as we discussed in Chapter 6, if we move along a direction $\mathbf{d}_y^k$ that lies in the null space of the matrix $\mathbf{A}_k = \mathbf{A}\mathbf{X}_k$ for an appropriate step-length $\alpha_k > 0$, then the new point $\mathbf{y}^{k+1} = \mathbf{e} + \alpha_k\mathbf{d}_y^k$ remains interior feasible to problem (7.1′). Moreover, its inverse image $\mathbf{x}^{k+1} = T_k^{-1}(\mathbf{y}^{k+1}) = \mathbf{X}_k\mathbf{y}^{k+1}$ becomes a new interior solution to problem (7.1).

Since our objective is to minimize the value of the objective function, the strategy of adopting the steepest descent applies. In other words, we want to project the negative gradient $-\mathbf{c}^k$ onto the null space of matrix $\mathbf{A}_k$ to create a good direction $\mathbf{d}_y^k$ with improved value of the objective function in the transformed space. In order to do so, we first define the *null space projection matrix* by

$$\mathbf{P}_k = \mathbf{I} - \mathbf{A}_k^T(\mathbf{A}_k\mathbf{A}_k^T)^{-1}\mathbf{A}_k = \mathbf{I} - \mathbf{X}_k\mathbf{A}^T(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{X}_k \tag{7.6}$$

Then, the moving direction $\mathbf{d}_y^k$, similar to (6.12), is given by

$$\mathbf{d}_y^k = \mathbf{P}_k(-\mathbf{c}^k) = -[\mathbf{I} - \mathbf{X}_k\mathbf{A}^T(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{X}_k]\mathbf{X}_k\mathbf{c} \tag{7.7}$$

Note that the projection matrix $\mathbf{P}_k$ is well defined as long as $\mathbf{A}$ has full row rank and $\mathbf{x}^k > \mathbf{0}$. It is also easy to verify that $\mathbf{A}\mathbf{X}_k\mathbf{d}^k = \mathbf{0}$. Figure 7.2 illustrates this projection mapping.

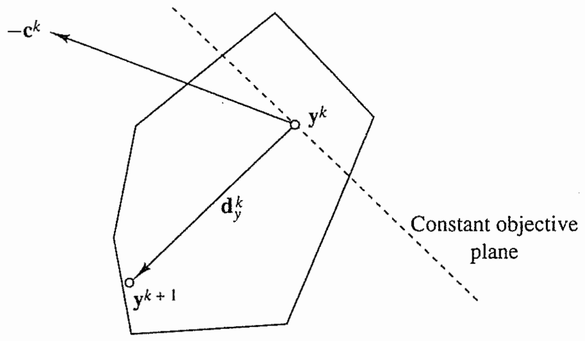

Figure 7.2

Now we are in a position to translate, in the transformed solution space, the current interior solution $\mathbf{y}^k = \mathbf{e}$ along the direction of $\mathbf{d}_y^k$ to a new interior solution $\mathbf{y}^{k+1} > \mathbf{0}$ with an improved objective value. In doing so, we have to choose an appropriate step-length $\alpha_k > 0$ such that

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \alpha_k\mathbf{d}_y^k > \mathbf{0} \tag{7.8}$$

Notice that if $\mathbf{d}_y^k \geq \mathbf{0}$, then α_k can be any positive number without leaving the interior region. On the other hand, if $(d_y^k)_i < 0$ for some i, then α_k has to be smaller than

$$\frac{y_i^k}{-(d_y^k)_i} = \frac{1}{-(d_y^k)_i}$$

Therefore we can choose $0 < \alpha < 1$ and apply the minimum ratio test

$$\alpha_k = \min_i \left\{ \frac{\alpha}{-(d_y^k)_i} \,\middle|\, (d_y^k)_i < 0 \right\} \tag{7.9}$$

to determine an appropriate step-length that guarantees the positivity of $\mathbf{y}^{k+1}$. When α is close to 1, the current solution is moved "almost all the way" to the nearest positivity wall to form a new interior solution in the transformed space. This translation is also illustrated in Figure 7.2.

Our next task is to map the new solution $\mathbf{y}^{k+1}$ back to the original solution space for obtaining an improved solution $\mathbf{x}^{k+1}$ to problem (7.1). This could be done by applying the inverse transformation T_k^{-1} to $\mathbf{y}^{k+1}$. In other words, we have

$$\begin{aligned}
\mathbf{x}^{k+1} &= T_k^{-1}\left(\mathbf{y}^{k+1}\right) = \mathbf{X}_k \mathbf{y}^{k+1} \\
&= \mathbf{x}^k + \alpha_k \mathbf{X}_k \mathbf{d}_y^k \\
&= \mathbf{x}^k - \alpha_k \mathbf{X}_k \mathbf{P}_k \mathbf{X}_k \mathbf{c} \\
&= \mathbf{x}^k - \alpha_k \mathbf{X}_k \left[\mathbf{I} - \mathbf{X}_k \mathbf{A}^T \left(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T\right)^{-1} \mathbf{A}\mathbf{X}_k\right] \mathbf{X}_k \mathbf{c} \\
&= \mathbf{x}^k - \alpha_k \mathbf{X}_k^2 \left[\mathbf{c} - \mathbf{A}^T \left(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T\right)^{-1} \mathbf{A}\mathbf{X}_k^2 \mathbf{c}\right] \\
&= \mathbf{x}^k - \alpha_k \mathbf{X}_k^2 \left[\mathbf{c} - \mathbf{A}^T \mathbf{w}^k\right]
\end{aligned} \tag{7.10}$$

where

$$\mathbf{w}^k = \left(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T\right)^{-1} \mathbf{A}\mathbf{X}_k^2 \mathbf{c} \tag{7.11}$$

This means the moving direction in the original solution space is $\mathbf{d}_x^k = -\mathbf{X}_k^2[\mathbf{c} - \mathbf{A}^T\mathbf{w}^k]$ and the step-length is α_k, while $\mathbf{d}_y^k = -\mathbf{X}_k[\mathbf{c} - \mathbf{A}^T\mathbf{w}^k]$ in the transformed space.

Several important observations can be made here:

Observation 1. Note that $\mathbf{d}_y^k = -\mathbf{P}_k\mathbf{c}^k$ and $\mathbf{d}_x^k = \mathbf{X}_k\mathbf{d}_y^k$. Since $\mathbf{P}_k$ is a projection mapping, we see that

$$\begin{aligned}
\mathbf{c}^T\mathbf{x}^{k+1} &= \mathbf{c}^T\mathbf{x}^k + \alpha_k \mathbf{c}^T \mathbf{X}_k \mathbf{d}_y^k \\
&= \mathbf{c}^T\mathbf{x}^k + \alpha_k (\mathbf{c}^k)^T \mathbf{d}_y^k \\
&= \mathbf{c}^T\mathbf{x}^k - \alpha_k (\mathbf{d}_y^k)^T \mathbf{d}_y^k \\
&= \mathbf{c}^T\mathbf{x}^k - \alpha_k \left|\left|\mathbf{d}_y^k\right|\right|^2
\end{aligned} \tag{7.12}$$

This implies that $\mathbf{x}^{k+1}$ is indeed an improved solution if the moving direction $\mathbf{d}_y^k \neq \mathbf{0}$. Moreover, we have the following lemmas:

Lemma 7.1. If there exists an $\mathbf{x}^k \in P^0$ with $\mathbf{d}_y^k > \mathbf{0}$, then the linear programming problem (7.1) is unbounded.

Proof. Since $\mathbf{d}_y^k$ is in the null space of the constraint matrix $\mathbf{AX}_k$ and $\mathbf{d}_y^k > \mathbf{0}$, we know $\mathbf{y}^{k+1} = \mathbf{y} + \alpha_k \mathbf{d}_y^k$ is feasible to problem (7.1′), for any $\alpha_k > 0$. Consequently, we can set α_k to be positive infinity, then Equation (7.12) implies that the limit of $\mathbf{c}^T\mathbf{x}^{k+1}$ approaches minus infinity in this case, for $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{X}_k \mathbf{d}_y^k \in P$.

Lemma 7.2. If there exists an $\mathbf{x}^k \in P^0$ with $\mathbf{d}_y^k = \mathbf{0}$, then every feasible solution of the linear programming problem (7.1) is optimal.

Proof. Remember that $\mathbf{P}_k$ is a null space projection matrix. For $\mathbf{d}_y^k = -\mathbf{P}_k\mathbf{X}_k\mathbf{c} = \mathbf{0}$, we know that $\mathbf{X}_k\mathbf{c}$ is in the orthogonal complement of the null space of matrix $\mathbf{AX}_k$. Since the orthogonal complement in this case is the row space of matrix $\mathbf{AX}_k$, there exists a vector $\mathbf{u}^k$ such that

$$(\mathbf{AX}_k)^T\mathbf{u}^k = \mathbf{X}_k\mathbf{c} \quad \text{or} \quad (\mathbf{u}^k)^T\mathbf{AX}_k = \mathbf{c}^T\mathbf{X}_k$$

Since $\mathbf{X}_k^{-1}$ exists, it follows that $(\mathbf{u}^k)^T\mathbf{A} = \mathbf{c}^T$. Now, for any feasible solution $\mathbf{x}$,

$$\mathbf{c}^T\mathbf{x} = (\mathbf{u}^k)^T\mathbf{Ax} = (\mathbf{u}^k)^T\mathbf{b}$$

Since $(\mathbf{u}^k)^T\mathbf{b}$ does not depend on $\mathbf{x}$, the value of $\mathbf{c}^T\mathbf{x}$ remains constant over P.

Lemma 7.3. If the linear programming problem (7.1) is bounded below and its objective function is not constant, then the sequence $\{\mathbf{c}^T\mathbf{x}^k \mid k = 1, 2, \ldots\}$ is well-defined and strictly decreasing.

Proof. This is a direct consequence of Lemmas 7.1, 7.2, and Equation (7.12).

Observation 2. If $\mathbf{x}^k$ is actually a vertex point, then expression (7.11) can be reduced to $\mathbf{w}^k = (\mathbf{B}^T)^{-1}\mathbf{c}_B$ which was defined as *"dual vector"* in Chapter 4. Hence we call $\mathbf{w}^k$ the *dual estimates* (corresponding to the primal solution $\mathbf{x}^k$) in the primal affine scaling algorithm. Moreover, in this case, the quantity

$$\mathbf{r}^k = \mathbf{c} - \mathbf{A}^T\mathbf{w}^k \tag{7.13}$$

reduces to $\mathbf{c} - \mathbf{A}^T(\mathbf{B}^T)^{-1}\mathbf{c}_B$, which is the so-called reduced cost vector in the simplex method. Hence we call $\mathbf{r}^k$ the *reduced cost vector* associated with $\mathbf{x}^k$ in the affine scaling algorithm.

Notice that when $\mathbf{r}^k \geq \mathbf{0}$, the dual estimate $\mathbf{w}^k$ becomes a dual feasible solution and $(\mathbf{x}^k)^T\mathbf{r}^k = \mathbf{e}^T\mathbf{X}_k\mathbf{r}^k$ becomes the duality gap of the feasible solution pair $(\mathbf{x}^k, \mathbf{w}^k)$, i.e.,

$$\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k = \mathbf{e}^T\mathbf{X}_k\mathbf{r}^k \tag{7.14}$$

In case $\mathbf{e}^T\mathbf{X}_k\mathbf{r}^k = 0$ with $\mathbf{r}^k \geq \mathbf{0}$, then we have achieved primal feasibility at $\mathbf{x}^k$, dual feasibility at $\mathbf{w}^k$, and complementary slackness conditions. In other words, $\mathbf{x}^k$ is primal optimal and $\mathbf{w}^k$ dual optimal.

Based on the above discussions, here we outline an iterative procedure for the primal affine scaling algorithm.

Step 1 (initialization): Set $k = 0$ and find $\mathbf{x}^0 > \mathbf{0}$ such that $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$. (Details will be discussed later.)

Step 2 (computation of dual estimates): Compute the vector of dual estimates

$$\mathbf{w}^k = (\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{X}_k^2\mathbf{c}$$

where $\mathbf{X}_k$ is a diagonal matrix whose diagonal elements are the components of $\mathbf{x}^k$.

Step 3 (computation of reduced costs): Calculate the reduced costs vector

$$\mathbf{r}^k = \mathbf{c} - \mathbf{A}^T\mathbf{w}^k$$

Step 4 (check for optimality): If $\mathbf{r}^k \geq \mathbf{0}$ and $\mathbf{e}^T\mathbf{X}_k\mathbf{r}^k \leq \epsilon$ (a given small positive number), then STOP. $\mathbf{x}^k$ is primal optimal and $\mathbf{w}^k$ is dual optimal. Otherwise, go to the next step.

Step 5 (obtain the direction of translation): Compute the direction

$$\mathbf{d}_y^k = -\mathbf{X}_k\mathbf{r}^k$$

Step 6 (check for unboundedness and constant objective value): If $\mathbf{d}_y^k > \mathbf{0}$, then STOP. The problem is unbounded. If $\mathbf{d}_y^k = \mathbf{0}$, then also STOP. $\mathbf{x}^k$ is primal optimal. Otherwise go to Step 7.

Step 7 (compute step-length): Compute the step-length

$$\alpha_k = \min_i \left\{ \frac{\alpha}{-(d_y^k)_i} \;\middle|\; (d_y^k)_i < 0 \right\} \qquad \text{where } 0 < \alpha < 1$$

Step 8 (move to a new solution): Perform the translation

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k\mathbf{X}_k\mathbf{d}_y^k$$

Reset $k \leftarrow k+1$ and go to step 2.

The following example illustrates the steps of primal affine scaling algorithm.

Example 7.1

$$\begin{aligned} \text{Minimize} \quad & -2x_1 + x_2 \\ \text{subject to} \quad & x_1 - x_2 + x_3 = 15 \\ & x_2 + x_4 = 15 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

In this case,

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \qquad \mathbf{b} = [15 \quad 15]^T, \qquad \text{and} \quad \mathbf{c} = [-2 \quad 1 \quad 0 \quad 0]^T$$

Let us start with, say, $\mathbf{x}^0 = [10 \quad 2 \quad 7 \quad 13]^T$, which is an interior feasible solution. Hence,

$$\mathbf{X}_0 = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 13 \end{bmatrix} \quad \text{and} \quad \mathbf{w}^0 = (\mathbf{A}\mathbf{X}_0^2\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{X}_0^2\mathbf{c} = [-1.33353 \quad -0.00771]^T$$

Moreover,

$$\mathbf{r}^0 = \mathbf{c} - \mathbf{A}^T\mathbf{w}^0 = [-0.66647 \quad -0.32582 \quad 1.33535 \quad -0.00771]^T$$

Since some components of $\mathbf{r}^0$ are negative and $\mathbf{e}^T\mathbf{X}_0\mathbf{r}^0 = 2.1187$, we know that the current solution is nonoptimal. Therefore we proceed to synthesize the direction of translation with

$$\mathbf{d}_y^0 = -\mathbf{X}_0\mathbf{r}^0 = [6.6647 \quad 0.6516 \quad -9.3475 \quad 0.1002]^T$$

Suppose that $\alpha = 0.99$ is chosen, then the step-length

$$\alpha_0 = \frac{0.99}{9.3475} = 0.1059$$

Therefore, the new solution is

$$\mathbf{x}^1 = \mathbf{x}^0 + \alpha_0\mathbf{X}_0\mathbf{d}_y^0 = [17.06822 \quad 2.13822 \quad 0.07000 \quad 12.86178]^T$$

Notice that the objective function value has been improved from -18 to -31.99822. The reader may continue the iterations further and verify that the iterative process converges to the optimal solution $\mathbf{x}^* = [30 \quad 15 \quad 0 \quad 0]^T$ with optimal value -45.

Convergence of the primal affine scaling algorithm. Our objective is to show that the sequence $\{\mathbf{x}^k\}$ generated by the primal affine scaling algorithm (without stopping at Step 6) converges to an optimal solution of the linear programming problem (7.1). In order to simplify our proof, we make the following assumptions:

1. The linear programming problem under consideration has a bounded feasible domain with nonempty interior.
2. The linear programming problem is both primal nondegenerate and dual nondegenerate.

The first assumption rules out the possibility of terminating the primal affine scaling algorithm with unboundedness, and it can be further shown that (see Exercise 7.5) these two assumptions imply that (i) the matrix $\mathbf{A}\mathbf{X}_k$ is of full rank for every $\mathbf{x}^k \in P$ and (ii) the vector $\mathbf{r}^k$ has at most m zeros for every $\mathbf{w}^k \in R^m$.

We start with some simple facts.

Lemma 7.4. When the primal affine scaling algorithm applies, $\lim_{k\to\infty} \mathbf{X}_k\mathbf{r}^k = \mathbf{0}$.

Proof. Since $\{\mathbf{c}^T\mathbf{x}^k\}$ is monotonically decreasing and bounded below (by the first assumption), the sequence converges. Hence Equations (7.12) and (7.9) imply that

$$0 = \lim_{k\to\infty}(\mathbf{c}^T\mathbf{x}^k - \mathbf{c}^T\mathbf{x}^{k+1}) = \lim_{k\to\infty}\alpha_k||\mathbf{d}_y^k||^2 \geq \lim_{k\to\infty}\frac{\alpha}{||\mathbf{d}_y^k||}||\mathbf{d}_y^k||^2$$

Notice that $\alpha > 0$ and $||\mathbf{d}_y^k|| \geq 0$, we have

$$\lim_{k\to\infty}||\mathbf{d}_y^k|| = \lim_{k\to\infty}||\mathbf{X}_k\mathbf{r}^k|| = 0.$$

The result stated follows immediately.

The reader may recall that the above result is exactly the complementary slackness condition introduced in Chapter 4. Let us define $C \subset P$ to be the set in which the complementary slackness holds. That is,

$$C = \left\{\mathbf{x}^k \in P \;\middle|\; \mathbf{X}_k\mathbf{r}^k = \mathbf{0}\right\} \tag{7.15}$$

Furthermore, we introduce $D \subset P$ to be the set in which the dual feasibility condition holds, i.e.,

$$D = \left\{\mathbf{x}^k \in P \;\middle|\; \mathbf{r}^k \geq \mathbf{0}\right\} \tag{7.16}$$

In view of the optimality conditions of the linear programming problem, it is easy to prove the following result.

Lemma 7.5. For any $\mathbf{x} \in C \cap D$, $\mathbf{x}$ is an optimal solution to the linear programming problem (7.1).

We are now ready to prove that the sequence $\{\mathbf{x}^k\}$ generated by the primal affine scaling algorithm does converge to an optimal solution of problem (7.1). First, we show that

Theorem 7.1. If $\{\mathbf{x}^k\}$ converges, then $\mathbf{x}^* = \lim_{k\to\infty}\mathbf{x}^k$ is an optimal solution to problem (7.1).

Proof. We prove this result by contradiction. First notice that when $\{\mathbf{x}^k\}$ converges to $\mathbf{x}^*$, $\mathbf{x}^*$ must be primal feasible. However, let us assume that $\mathbf{x}^*$ is not primal optimal.

Since $\mathbf{r}^k(\cdot)$ is a continuous function of $\mathbf{x}$ at $\mathbf{x}^k$, we know $\mathbf{r}^* = \lim_{k\to\infty}\mathbf{r}^k$ is well defined. Moreover, Lemma 7.4 implies that

$$\mathbf{X}^*\mathbf{r}^* = \lim_{k\to\infty}\mathbf{X}_k r^k = \mathbf{0}$$

Hence we have $\mathbf{x}^* \in C$. By our assumption and Lemma 7.5, we know that $\mathbf{x}^* \notin D$. Therefore, there exists at least one index j such that $r_j^* < 0$. Remembering that $\mathbf{x}^* \in C$, we have $x_j^* = 0$. Owing to the continuity of $\mathbf{r}^k$, there exists an integer K such that for any $k \geq K$, $\{r_j^k\} < 0$. However, consider that

$$x_j^{k+1} = x_j^k - \alpha_k(x_j^k)^2 r_j^k$$

Since $(x_j^k)^2 r_j^k < 0$, we have $x_j^{k+1} > x_j^k > 0, \forall\, k \geq K$, which contradicts the fact that $x_j^k \to x_j^* = 0$. Hence we know our assumption must be wrong and $\mathbf{x}^*$ is primal optimal.

The remaining work is to show that the sequence $\{\mathbf{x}^k\}$ indeed converges.

Theorem 7.2. The sequence $\{\mathbf{x}^k\}$ generated by the primal affine scaling algorithm is convergent.

Proof. Since the feasible domain is nonempty, closed, and bounded, owing to compactness the sequence $\{\mathbf{x}^k\}$ has at least one accumulation point in P, say $\mathbf{x}^*$. Our objective is to show that $\mathbf{x}^*$ is also the only accumulation point of $\{\mathbf{x}^k\}$ and hence it becomes the limit of $\{\mathbf{x}^k\}$.

Noting that $\mathbf{r}^k(\cdot)$ is a continuous function of $\mathbf{x}^k$ and applying Lemma 7.4, we can conclude that $\mathbf{x}^* \in C$. Furthermore, the nondegeneracy assumption implies that every element in C including $\mathbf{x}^*$ must be a basic feasible solution (vertex of P). Hence we can denote its nonbasic variables by $\mathbf{x}_N^*$ and define $\tilde{\mathbf{N}}$ as the index set of these nonbasic variables. In addition, for any $\delta > 0$, we define a "δ-ball" around $\mathbf{x}^*$ by

$$B_\delta = \left\{\mathbf{x}^k \in P \;\middle|\; \mathbf{x}_N^k < \delta \mathbf{e}\right\}$$

Let $\mathbf{r}^*$ be the reduced cost vector corresponding to $\mathbf{x}^*$. The primal and dual nondegeneracy assumption ensures us to find an $\epsilon > 0$ such that

$$\min_{j \in \tilde{\mathbf{N}}} |r_j^*| > \epsilon$$

Remember that the nondegeneracy assumption forces every member of C to be a vertex of P and there are only a finite number of vertices in P, hence C has a finite number of elements and we can choose an appropriate $\delta > 0$ such that

$$B_{2\delta} \cap C = \mathbf{x}^* \tag{7.17}$$

and

$$\min_{j \in \tilde{\mathbf{N}}} |r_j^k| > \epsilon, \qquad \forall\, \mathbf{x}^k \in B_\delta \tag{7.18}$$

Recalling that

$$\lim_{k\to\infty} \left|\left|\mathbf{d}_y^k\right|\right|^2 = \lim_{k\to\infty} \left|\left|\mathbf{X}_k \mathbf{r}^k\right|\right|^2 = 0$$

we have

$$\lim_{k\to\infty} \left[x_j^k r_j^k\right]^2 = 0$$

Owing to the boundedness assumption, we know that the step-length α_k at each iteration is a positive but bounded number. Therefore, for appropriately chosen ϵ and δ, if $\mathbf{x}^k \in B_\delta$, which is sufficiently close to $\mathbf{x}^*$, we see that $\alpha_k [x_j^k r_j^k]^2 < \epsilon\delta$. Hence we can define a set

$$S_{\epsilon,\delta} = \left\{\mathbf{x}^k \in B_\delta \;\middle|\; \alpha_k \left[x_j^k r_j^k\right]^2 < \epsilon\delta, \forall\, j \in \tilde{\mathbf{N}}\right\}$$

Now, for any $\mathbf{x}^k \in S_{\epsilon,\delta}$, (7.18) implies that

$$\alpha_k \left[x_j^k\right]^2 |r_j^k| < \delta, \quad \forall\, j \in \tilde{\mathbf{N}}$$

which further implies that

$$x_j^{k+1} = x_j^k - \alpha_k \left[\left(x_j^k\right)^2 r_j^k\right] < 2\delta, \qquad \forall\, j \in \tilde{\mathbf{N}}$$

This means $\mathbf{x}^{k+1} \in B_{2\delta}$ if $\mathbf{x}^k \in S_{\epsilon,\delta}$.

Now we are ready to show that $\mathbf{x}^*$ is the only accumulation point of $\{\mathbf{x}^k\}$ by contradiction. We suppose that $\{\mathbf{x}^k\}$ has more than one accumulation point. Since $\mathbf{x}^*$ is an accumulation point, the sequence $\{\mathbf{x}^k\}$ visits $S_{\epsilon,\delta}$ infinitely often. But because $\mathbf{x}^*$ is not the only accumulation point, the sequence has to leave $S_{\epsilon,\delta}$ infinitely often. However, each time when the sequence leaves $S_{\epsilon,\delta}$, it stays in $B_{2\delta}\backslash S_{\epsilon,\delta}$. Therefore, infinitely many elements of $\{\mathbf{x}^k\}$ fall in $B_{2\delta}\backslash S_{\epsilon,\delta}$. Notice that this difference set has a compact closure, and the subsequence of $\{\mathbf{x}^k\}$ belonging to $B_{2\delta}\backslash S_{\epsilon,\delta}$ must have an accumulation point in the compact closure. Noting the definition of C, we know that every accumulation point of $\{\mathbf{x}^k\}$ must belong to it. However, C is disjoint from the closure of $B_{2\delta}\backslash S_{\epsilon,\delta}$. This fact, together with Equation (7.17), causes a contradiction. Thus $\mathbf{x}^*$ is indeed the limit of the sequence $\{\mathbf{x}^k\}$.

More results on the convergence of the affine scaling algorithm under degeneracy have appeared recently. Some references are included at the end of this chapter for further information.

7.1.2 Implementing the Primal Affine Scaling Algorithm

Many implementation issues need to be addressed. In this section, we focus on the starting mechanisms, checking for optimality, and finding an optimal basic feasible solution.

Starting the primal affine scaling algorithm. Parallel to our discussion for the revised simplex method, here we introduce two mechanisms, namely, the *big-M method* and *two-phase method* for finding an initial interior feasible solution. The first method is more easily implemented and suitable for most of the applications. However, more serious commercial implementations often consider the second method for stability.

Big-M Method. In this method, we add one more artificial variable x^a associated with a large positive number M to the original linear program problem to make $(1, 1, \ldots, 1) \in R^{n+1}$ become an initial interior feasible solution to the following problem:

$$\text{Minimize} \quad \mathbf{c}^T\mathbf{x} + Mx^a \tag{7.19a}$$

$$\text{subject to} \quad [\mathbf{A} \mid \mathbf{b} - \mathbf{A}\mathbf{e}] \left[\frac{\mathbf{x}}{x^a}\right] = \mathbf{b}, \qquad \mathbf{x} \geq \mathbf{0}, \quad x^a \geq 0 \tag{7.19b}$$

where $\mathbf{e} = (1, 1, \ldots, 1)^T \in R^n$.

Comparing to the big-M method for the revised simplex method, here we have only $n+1$ variables, instead of $n+m$. When the primal affine scaling algorithm is applied to the big-M problem (7.19) with sufficiently large M, since the problem is feasible, we either arrive at an optimal solution to the big-M problem or conclude that the problem is unbounded. Similar to the discussions in Chapter 4, if the artificial variable remains positive in the final solution $(\mathbf{x}^*, x^{a^*})$ of the big-M problem, then the original linear programming problem is infeasible. Otherwise, either the original problem is identified to be unbounded below, or $\mathbf{x}^*$ solves the original linear programming problem.

Two-Phase Method. Let us choose an arbitrary $\mathbf{x}^0 > 0$ and calculate $\mathbf{v} = \mathbf{b} - \mathbf{A}\mathbf{x}^0$. If $\mathbf{v} = \mathbf{0}$, then $\mathbf{x}^0$ is an initial interior feasible solution. Otherwise, we consider the following Phase-I linear programming problem with $n+1$ variables:

$$\text{Minimize} \quad u \tag{7.20a}$$

$$\text{subject to} \quad [\mathbf{A} \mid \mathbf{v}] \begin{bmatrix} \mathbf{x} \\ - \\ u \end{bmatrix} = \mathbf{b}, \qquad \mathbf{x} \geq \mathbf{0}, \quad u \geq 0 \tag{7.20b}$$

It is easy to verify that the vector

$$\hat{\mathbf{x}}^0 = \begin{pmatrix} \mathbf{x}^0 \\ - \\ u^0 \end{pmatrix} = \begin{pmatrix} \mathbf{x}^0 \\ - \\ 1 \end{pmatrix} > \mathbf{0}$$

is an interior feasible solution to the Phase-I problem. Hence the primal affine scaling algorithm can be applied to solve this problem. Moreover, since the Phase-I problem is bounded below by 0, the primal affine scaling algorithm will always terminate with an optimal solution, say $(\mathbf{x}^*, u^*)^T$. Again, similar to the discussions in Chapter 4, if $u^* > 0$, then the original linear programming problem is infeasible. Otherwise, since the Phase-I problem treats the problem in a higher-dimensional space, we can show that, except for very rare cases with measure zero, $\mathbf{x}^* > \mathbf{0}$ will become an initial interior feasible solution to the original problem.

Note that the difference in dimensionality between the original and Phase-I problems could cause extra computations for a simpleminded implementation. First of all, owing to numerical imprecisions in computers, the optimal solution $\mathbf{x}^*$ obtained from Phase-I could become infeasible to the original problem. In other words, we need to restore primal feasibility before the second-phase computation. Second, the difference of dimensionality in the fundamental matrices $\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T$ (of the original problem) and $\hat{\mathbf{A}}\hat{\mathbf{X}}_k^2\hat{\mathbf{A}}^T$ (of the Phase-I problem) could prevent us from using the same "symbolic factorization template" (to be discussed in Chapter 10) for fast computation of their inverse matrices. Therefore, it would be helpful if we could operate the Phase-I iterations in the original n-dimensional space.

In order to do so, let us assume we are at the kth iteration of applying the primal affine scaling to the Phase-I problem. We denote

$$\hat{\mathbf{x}}^k = \begin{pmatrix} \mathbf{x}^k \\ - \\ u^k \end{pmatrix}$$

to be the current solution,

$$\hat{\mathbf{A}} = [\mathbf{A} \mid \mathbf{v}], \qquad \hat{\mathbf{X}}_k = \begin{bmatrix} \mathbf{X}_k & 0 \\ \mathbf{0} & u^k \end{bmatrix}, \quad \text{and} \quad \hat{\mathbf{A}}_k = \hat{\mathbf{A}}\hat{\mathbf{X}}_k$$

Remember that the gradient of the objective function is given by

$$\hat{\mathbf{c}} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$$

hence the moving direction in the original space of Phase-I problem is given by

$$\hat{\mathbf{d}}_x^k = -\hat{\mathbf{X}}_k[\mathbf{I} - \hat{\mathbf{A}}_k^T(\hat{\mathbf{A}}_k\hat{\mathbf{A}}_k^T)^{-1}\hat{\mathbf{A}}_k]\hat{\mathbf{c}}^k \tag{7.21}$$

where $\hat{\mathbf{c}}^k = \hat{\mathbf{X}}_k\hat{\mathbf{c}}$. If we further define

$$\mathbf{w}^k = (\hat{\mathbf{A}}_k\hat{\mathbf{A}}_k)^{-1}\hat{\mathbf{A}}_k\hat{\mathbf{c}}^k \tag{7.22}$$

then we have

$$\hat{\mathbf{d}}_x^k = -\hat{\mathbf{X}}_k(\hat{\mathbf{c}}^k - \hat{\mathbf{A}}_k^T\mathbf{w}^k) \tag{7.23}$$

Simple calculation results in

$$\hat{\mathbf{A}}_k\hat{\mathbf{A}}_k^T = [\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T + (u^k)^2\mathbf{v}\mathbf{v}^T] \tag{7.24}$$

and

$$\hat{\mathbf{A}}_k\hat{\mathbf{c}}^k = [\mathbf{A} \mid \mathbf{v}]\begin{bmatrix} \mathbf{X}_k & \mathbf{0} \\ 0 & u^k \end{bmatrix}\begin{bmatrix} \mathbf{0} \\ \hline u^k \end{bmatrix} = [\mathbf{A}\mathbf{X}_k \mid \mathbf{v}u^k]\begin{bmatrix} \mathbf{0} \\ \hline u^k \end{bmatrix} = (u^k)^2\mathbf{v} \tag{7.25}$$

Combining (7.22), (7.24), and (7.25), we see that

$$\mathbf{w}^k = [\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T + (u^k)^2\mathbf{v}\mathbf{v}^T]^{-1}(u^k)^2\mathbf{v} \tag{7.26}$$

Applying the Sherman-Woodbury-Morrison lemma (Lemma 4.2), we have

$$\mathbf{w}^k = \frac{1}{[(u^k)^{-2} + \mathbf{v}^T(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T)^{-1}\mathbf{v}]}(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T)^{-1}\mathbf{v} = \frac{1}{[(u^k)^{-2} + \gamma]}(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T)^{-1}\mathbf{v} \tag{7.27}$$

where $\gamma = \mathbf{v}^T(\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T)^{-1}\mathbf{v}$. Plugging corresponding terms into (7.23), we see that

$$\begin{aligned}
\hat{\mathbf{d}}_x^k &= -\hat{\mathbf{X}}_k(\hat{\mathbf{c}}^k - \hat{\mathbf{A}}_k^T\mathbf{w}^k) \\
&= -\hat{\mathbf{X}}_k\left(\begin{bmatrix} \mathbf{0} \\ \hline u^k \end{bmatrix} - \begin{bmatrix} \mathbf{X}_k\mathbf{A}^T \\ \hline u^k\mathbf{v}^T \end{bmatrix}\mathbf{w}^k\right) \\
&= -\begin{bmatrix} \mathbf{X}_k & \mathbf{0} \\ 0 & u^k \end{bmatrix}\begin{bmatrix} -\mathbf{X}_k\mathbf{A}^T\mathbf{w}^k \\ \hline u^k\left(1 - \mathbf{v}^T\mathbf{w}^k\right) \end{bmatrix} \\
&= -\begin{bmatrix} -\mathbf{X}_k^2\mathbf{A}^T\mathbf{w}^k \\ \hline (u^k)^2\left(1 - \mathbf{v}^T\mathbf{w}^k\right) \end{bmatrix}
\end{aligned} \tag{7.28}$$

Observing that

$$(u^k)^2\left(1 - \mathbf{v}^T\mathbf{w}^k\right) = (u^k)^2\left(1 - \frac{\gamma}{(u^k)^{-2} + \gamma}\right) = \frac{1}{(u^k)^{-2} + \gamma}$$

we further have

$$\hat{\mathbf{d}}_x^k = \frac{1}{(u^k)^{-2} + \gamma} \begin{bmatrix} \mathbf{X}_k^2 \mathbf{A}^T (\mathbf{A}\mathbf{X}_k^2 \mathbf{A}^T)^{-1} (\mathbf{b} - \mathbf{A}\mathbf{x}^0) \\ -1 \end{bmatrix} \tag{7.29}$$

Notice that the scalar multiplier in (7.29) will be absorbed into the step-length and the last element of the moving direction is -1. Hence we know that the algorithm tries to reduce u all the time. In this expression, we clearly see the computation of $\hat{\mathbf{d}}_x^k$ can be performed in the original $n-$dimensional space and the template for the factorization of $\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T$ can be used for both Phase I and II.

In order to compute the step-length, we consider that

$$\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \alpha_k \hat{\mathbf{d}}_x^k$$

Similar to the previous discussion, the step-length can be chosen as

$$\alpha_k = \min_i \left\{ \frac{\alpha}{-\left(\hat{d}_x^k\right)_i / \hat{x}_i^k} \,\middle|\, (\hat{d}_x^k)_i < 0 \right\} \qquad \text{for some } 0 < \alpha < 1$$

An interesting and important point to be observed here is that the Phase-I iterations may be initiated at any time (even during the Phase-II iterations). Once we detect that the feasibility of a current iterate is lost owing to numerical inaccuracies that stem from the finite word length of computers, Phase-I iterations can be applied to restore the feasibility. Hence sometimes we call it a "dynamic infeasibility correction" procedure. Sophisticated implementations should have this feature built in, since the primal method is quite sensitive to numerical truncations and round-off errors.

Having determined the starting mechanisms, we focus on the stopping rules for the implementation of the primal affine scaling algorithm.

Stopping rules. As we mentioned earlier, once the K-K-T conditions are met, an optimal solution pair is found. Hence we use the conditions of (1) primal feasibility, (2) dual feasibility, and (3) complementary slackness as the stopping rules. However, in real implementations these conditions are somewhat relaxed to accommodate the numerical difficulties due to limitations of machine accuracy.

Let $\mathbf{x}^k$ be a current solution obtained by applying the primal affine scaling algorithm. The primal feasibility condition requires that

(I) PRIMAL FEASIBILITY

$$\mathbf{A}\mathbf{x}^k = \mathbf{b}, \qquad \mathbf{x}^k \geq \mathbf{0} \tag{7.30}$$

In practice, the primal feasibility is often measured by

$$\sigma_p = \frac{||\mathbf{A}\mathbf{x}^k - \mathbf{b}||}{||\mathbf{b}|| + 1} \qquad \text{with} \quad \mathbf{x}^k \geq \mathbf{0} \tag{7.31}$$

Note that for $\mathbf{x}^k \geq \mathbf{0}$, if σ_p is small enough, we may accept $\mathbf{x}^k$ to be primal feasible. The addition of 1 in the denominator of (7.31) is to ensure numerical stability in computation.

(II) DUAL FEASIBILITY

The dual feasibility requires the nonnegativity of reduced costs, i.e.,

$$\mathbf{r}^k = \mathbf{c} - \mathbf{A}^T\mathbf{w}^k \geq \mathbf{0} \tag{7.32}$$

where $\mathbf{w}^k$ is the dual estimate defined in Equation (7.11). A practical measure of dual feasibility could be defined as

$$\sigma_d = \frac{||\mathbf{r}^k||}{||\mathbf{c}|| + 1} \tag{7.33}$$

where $||\mathbf{r}^k||$ and $||\mathbf{c}||$ are calculated only for those i such that $r_i^k < 0$. When σ_d is sufficiently small, we may claim the dual feasibility is satisfied by $\mathbf{w}^k$.

(III) COMPLEMENTARY SLACKNESS

The complementary slackness condition requires that

$$(\mathbf{x}^k)^T\mathbf{r}^k = \mathbf{e}\mathbf{X}_k\mathbf{r}^k = 0 \tag{7.34}$$

Since

$$\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k = \mathbf{e}^T\mathbf{X}_k\mathbf{r}^k \tag{7.35}$$

where $\mathbf{x}^k$ is primal feasible and $\mathbf{w}^k$ is dual feasible, we may define

$$\sigma_c = \mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k \tag{7.36}$$

to measure the complementary slackness condition.

In practice, we choose σ_p, σ_d and σ_c as sufficiently small positive numbers and use them to decide if the current iteration meets the stopping rules. According to the authors' experience, we have observed the following behavior for the primal affine scaling algorithm:

1. At each iteration, the computational bottleneck is due to the computation of the dual estimates $\mathbf{w}^k$.
2. Although the primal feasibility condition is theoretically maintained, the numerical truncation and round-off errors of computers could still cause infeasibility. Therefore, the primal feasibility needs to be carefully checked. Once the infeasibility is detected, we may apply the Phase-I "dynamic infeasibility correction" procedure to restore the primal feasibility.
3. The value of the objective function decreases dramatically in the early iterations, but the decreasing trend slows down considerably when the current solution becomes closer to optimality.

4. The algorithm is somewhat sensitive to primal degeneracy, especially when the iteration proceeds near optimality. But this is not universally true. In many cases, even with the presence of primal degeneracy, the algorithm still performs quite well.

Finding a basic feasible solution. Notice that, just like Karmarkar's algorithm, at each iteration, the current solution of the primal affine scaling algorithm always stays in the interior of the feasible domain P. In order to obtain a basic feasible solution, the purification scheme and related techniques described in Chapter 6 can be applied here.

7.1.3 Computational Complexity

Compared to Karmarkar's projective transformation, the affine scaling transformation is less complicated and more natural. The implementation of the primal affine scaling is also simple enough. It does not need the assumption of "zero optimal objective value" nor require the special "simplex structure." By far, it is one of the most "popular" variants of the interior-point method. According to R. Vanderbei, M. Meketon, and B. Freedman's experiment, for problems with dense constraint matrices, their primal affine scaling implementation takes about $7.3885m^{-0.0187}n^{0.1694}$ iterations to reach an optimal solution within $\epsilon = 10^{-3}$. The result was derived from the regression analysis of 137 randomly generated problems.

Although in practice the primal affine scaling algorithm performs very well, no proof shows the algorithm is a polynomial-time algorithm. Actually, N. Megiddo and M. Shub showed that the affine scaling algorithm might visit the neighborhoods of all the vertices of the Klee-Minty cube when a starting point is pushed to the boundary.

Potential push method. To avoid being trapped by the boundary behavior, a recentering method called *potential push* is introduced. The idea is to push a current solution $\mathbf{x}^k$ to a new interior solution $\hat{\mathbf{x}}^k$ which is away from the positivity walls but without increasing its objective value. Then continue the iterations from $\hat{\mathbf{x}}^k$. Figure 7.3 illustrates this concept.

In Figure 7.3, we move from $\mathbf{x}^{k-1}$ to a new solution $\mathbf{x}^k$ along the direction $\mathbf{d}_x^{k-1}$ provided by the primal affine scaling algorithm. Then we recenter $\mathbf{x}^k$ to $\hat{\mathbf{x}}^k$ by a "potential push" along the direction $\hat{\mathbf{d}}_x^k$ such that $\mathbf{x}^k$ and $\hat{\mathbf{x}}^k$ have the same objective value but $\hat{\mathbf{x}}^k$ is away from the boundary.

To achieve this goal, first we define a *potential function* $p(\mathbf{x})$, for each $\mathbf{x} > \mathbf{0}$, as

$$p(\mathbf{x}) = -\sum_{j=1}^{n} \log_e x_j \qquad (7.37)$$

The value of the potential function $p(\mathbf{x})$ becomes larger when $\mathbf{x}$ is closer to a positivity wall $x_j = 0$. Hence it creates a force to "push" $\mathbf{x}$ away from too close an approach to

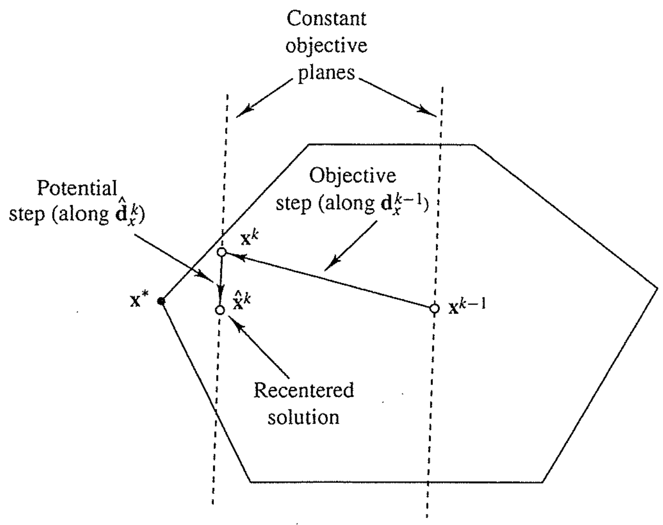
Constant objective planes
Potential step (along $\hat{\mathbf{d}}_x^k$)
Objective step (along $\mathbf{d}_x^{k-1}$)
$\mathbf{x}^k$
$\mathbf{x}^*$
$\hat{\mathbf{x}}^k$
$\mathbf{x}^{k-1}$
Recentered solution

Figure 7.3

a boundary by minimizing $p(\mathbf{x})$. With the potential function, we focus on solving the following "potential push" problem:

$$\text{Minimize} \quad p(\mathbf{x}) \tag{7.38a}$$

$$\text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \qquad \mathbf{x} > \mathbf{0} \tag{7.38b}$$

$$\mathbf{c}^T\mathbf{x} = \mathbf{c}^T\mathbf{x}^k \tag{7.38c}$$

Note that (7.38b) requires the solution of problem (7.38) to be an interior feasible solution to the original linear programming problem; (7.38c) requires it to keep the same objective value as $\mathbf{x}^k$; and minimizing $p(\mathbf{x})$ forces the solution away from the positivity walls. Therefore, we can take the optimal solution of problem (7.38) as $\hat{\mathbf{x}}^k$.

Similar to our discussions for the Phase-I problem, if we directly apply the primal affine scaling algorithm to solve the potential push problem, we have a mismatch in dimensionality, since problem (7.38) has one more constraint than the original linear programming problem. In order to implement the potential push method in a consistent framework with the primal affine scaling algorithm, we need to take care of requirement (7.38c) separately. Also notice that we do not really need to find an optimal solution to the potential push problem. Any feasible solution to problem (7.38) with improved value in $p(\mathbf{x})$ can be adopted as $\hat{\mathbf{x}}^k$.

One way to achieve this goal is to take $\mathbf{x}^k$ as an initial solution to problem (7.38), then project the negative gradient of $p(\mathbf{x})$ onto the null space of the constraint matrix $\mathbf{A}$ as a potential moving direction, say $\mathbf{p}^k$. But in order to keep the same objective value, we first project the negative gradient of the objective function $\mathbf{c}^T\mathbf{x}$ onto the null space of $\mathbf{A}$ and denote it as $\mathbf{g}$. Then, the recentering (or push) direction $\hat{\mathbf{d}}_x^k$ is taken to be the component of $\mathbf{p}^k$ which is orthogonal to $\mathbf{g}$. Finally, along this direction, we conduct a line search for an optimal step-length.

Mathematically speaking, we let $\mathbf{P} = \mathbf{I} - \mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{A}$ be the projection mapping and $\nabla p(\mathbf{x})$ be the gradient of the potential function. Then, we have

$$\mathbf{p}^k = -\mathbf{P}\left(\nabla p\left(\mathbf{x}^k\right)\right) = \left[\mathbf{I} - \mathbf{A}^T\left(\mathbf{AA}^T\right)^{-1}\mathbf{A}\right]\left(\frac{1}{\mathbf{x}^k}\right) \tag{7.39}$$

where

$$\frac{1}{\mathbf{x}^k} = \left(\frac{1}{x_1^k}, \ldots, \frac{1}{x_n^k}\right)^T$$

Similarly,

$$\mathbf{g} = -\mathbf{Pc} = -[\mathbf{I} - \mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{A}]\mathbf{c} \tag{7.40}$$

We now decompose $\mathbf{p}^k$ into two components, one along $\mathbf{g}$ and the other orthogonal to it. The first component can be expressed as $\mu\mathbf{g}$, for some $\mu > 0$, since it is along the direction of $\mathbf{g}$. Therefore the orthogonal component can be expressed as

$$\hat{\mathbf{d}}_x^k = \mathbf{p}^k - \mu\mathbf{g} \tag{7.41}$$

Moreover, the orthogonal condition requires that

$$(\hat{\mathbf{d}}_x^k)^T\mathbf{g} = 0 \tag{7.42}$$

which determines the value of μ by

$$\mu = \frac{(\mathbf{p}^k)^T\mathbf{g}}{\mathbf{g}^T\mathbf{g}} \tag{7.43}$$

and, consequently,

$$\hat{\mathbf{d}}_x^k = \mathbf{p}^k - \left[\frac{(\mathbf{p}^k)^T\mathbf{g}}{\mathbf{g}^T\mathbf{g}}\right]\mathbf{g} \tag{7.44}$$

Figure 7.4 illustrates this situation.

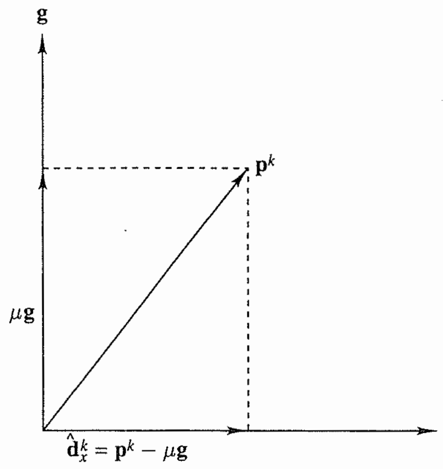

Figure 7.4

Now we focus on finding an appropriate step-length κ such that the point $\mathbf{x}^k$ is translated to a new solution $\hat{\mathbf{x}}^k = \mathbf{x}^k + \kappa\hat{\mathbf{d}}_x^k$ which has the same objective value as $\mathbf{c}^T\mathbf{x}^k$ but with a lower value in $p(\mathbf{x})$. To do so, we conduct a line search along the direction $\hat{\mathbf{d}}_x^k$. One of the easiest ways is via *binary search.* Note that the maximum value (say $\overline{\kappa}$) that κ can assume is given by

$$\overline{\kappa} = \frac{1}{\max_j \left\{-(\hat{d}_x^k)_j / x_j^k\right\}} \tag{7.45}$$

Hence we only have to search for κ in the interval $(0, \overline{\kappa})$ such that $p(\hat{\mathbf{x}}^k)$ assumes a minimum value.

Several issues are worth mentioning here:

1. When the potential push method is applied after each iteration of the primal affine scaling algorithm, since $\mathbf{P}$ needs to be evaluated only once for all iterations and a binary search is relatively inexpensive, the evaluation of the potential function required during the search becomes the most time-consuming operation associated with the potential push.
2. The purpose of applying potential push is to gain faster convergence by staying away from the boundary. If the extra speed of convergence obtained by potential push appears to be marginal, then it is not worth spending any major effort in it. Some coarse adjustments are good enough in this case. According to the authors' experience, no more than four or five searches per affine scaling iteration are needed to estimate $\hat{\mathbf{x}}^k$.
3. Recall that Karmarkar's potential function is given by (6.18), namely,

$$f(\mathbf{x}; \mathbf{c}) = n \log_e (\mathbf{c}^T\mathbf{x}) - \sum_{j=1}^{n} \log_e x_j$$

   Hence $f(\mathbf{x}; \mathbf{c}) = n \log_e (\mathbf{c}^T\mathbf{x}) - p(\mathbf{x})$, assuming that $\mathbf{c}^T\mathbf{x} > 0$. When the potential push is applied after each iteration of the primal affine scaling, we see the first term in $f(\mathbf{x}; \mathbf{c})$ is reduced by the affine scaling and the second term is reduced by the potential push. Thus the flavor of Karmarkar's approach is preserved.
4. Since the flavor of Karmarkar's potential function is preserved, it is conjectured that primal affine scaling together with potential push could result in a polynomial-time algorithm. But so far, no rigorous complexity proof has been provided.

Logarithmic barrier function method. Another way to stay away from the positivity walls is to incorporate a barrier function, with extremely high values along the boundaries $\{\mathbf{x} \in R^n \mid x_j = 0, \text{ for } some\ 1 \le j \le n\}$, into the original objective function. Minimizing this new objective function will automatically push a solution away from the positivity walls. The logarithmic barrier method considers the following nonlinear optimization problem:

$$\text{Minimize} \quad F_\mu(\mathbf{x}) = \mathbf{c}^T\mathbf{x} - \mu \sum_{j=1}^{n} \log_e x_j \tag{7.46a}$$

$$\text{subject to} \quad \mathbf{Ax} = \mathbf{b}, \qquad \mathbf{x} > \mathbf{0} \tag{7.46b}$$

where $\mu > 0$ is a scalar. If $\mathbf{x}^*(\mu)$ is an optimal solution to problem (7.46), and if $\mathbf{x}^*(\mu)$ tends to a point $\mathbf{x}^*$ as μ approaches zero, then it follows that $\mathbf{x}^*$ is an optimal solution to the original linear programming problem. Also notice that the positivity constraint $\mathbf{x} > \mathbf{0}$ is actually embedded in the definition of the logarithmic function. Hence, for any fixed $\mu > 0$, the Newton search direction $\mathbf{d}_\mu$ at a given feasible solution $\mathbf{x}$ is obtained by solving the following quadratic optimization problem:

$$\text{Minimize} \quad \frac{1}{2}\mathbf{d}^T\nabla^2 F_\mu(\mathbf{x})\mathbf{d} + ((\nabla F_\mu(\mathbf{x}))^T\mathbf{d} \tag{7.47a}$$

$$\text{subject to} \quad \mathbf{Ad} = \mathbf{0} \tag{7.47b}$$

where $\nabla F_\mu(\mathbf{x}) = \mathbf{c} - \mu\mathbf{X}^{-1}\mathbf{e}$ and $\nabla^2 F_\mu(\mathbf{x}) = \mu\mathbf{X}^{-2}$.

In other words, the Newton direction is in the null space of matrix $\mathbf{A}$ and it minimizes the quadratic approximation of $F_\mu(\mathbf{x})$. We let λ_μ denote the vector of Lagrange multipliers, then $\mathbf{d}_\mu$ and λ_μ satisfy the following system of equations:

$$\begin{pmatrix} \mu\mathbf{X}^{-2} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{d}_\mu \\ \lambda_\mu \end{pmatrix} = -\begin{pmatrix} \mathbf{c} - \mu\mathbf{X}^{-1}\mathbf{e} \\ 0 \end{pmatrix}$$

Letting $\phi_\mu = \mathbf{X}^{-1}\mu\mathbf{d}_\mu$, we have

$$\begin{pmatrix} \mathbf{I} & \mathbf{XA}^T \\ \mathbf{AX} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \phi_\mu \\ \lambda_\mu \end{pmatrix} = -\begin{pmatrix} \mathbf{Xc} - \mu\mathbf{e} \\ 0 \end{pmatrix}$$

It follows that

$$\phi_\mu = -[\mathbf{I} - \mathbf{XA}^T(\mathbf{AX}^2\mathbf{A}^T)^{-1}\mathbf{AX}](\mathbf{Xc} - \mu\mathbf{e}) \tag{7.48}$$

and

$$\mathbf{d}_\mu = -\frac{1}{\mu}\mathbf{X}[\mathbf{I} - \mathbf{XA}^T(\mathbf{AX}^2\mathbf{A}^T)^{-1}\mathbf{AX}](\mathbf{Xc} - \mu\mathbf{e}) \tag{7.49a}$$

Taking the given solution to be $\mathbf{x} = \mathbf{x}^k$ and comparing $\mathbf{d}_\mu$ with the primal affine scaling moving direction $\mathbf{d}_x^k$, we see that

$$\mathbf{d}_\mu = \frac{1}{\mu}\mathbf{d}_x^k + \mathbf{X}_k[\mathbf{I} - \mathbf{X}_k\mathbf{A}^T(\mathbf{AX}_k^2\mathbf{A}^T)^{-1}\mathbf{AX}_k]\mathbf{e} \tag{7.49b}$$

The additional component $\mathbf{X}_k[\mathbf{I} - \mathbf{X}_k\mathbf{A}^T(\mathbf{AX}_k^2\mathbf{A}^T)^{-1}\mathbf{AX}_k]\mathbf{e} = \mathbf{X}_k\mathbf{P}_k\mathbf{e}$ can be viewed as a force which pushes a solution away from the boundary. Hence some people call it a *"centering force,"* and call the logarithmic barrier method a *"primal affine scaling algorithm with centering force."*

While classical barrier function theory requires that $\mathbf{x}^k$ solves problem (7.46) explicitly before $\mu = \mu_k$ is reduced, C. Gonzaga has pointed out that there exists $\mu_0 > 0$, $0 < \rho < 1$, and $\alpha > 0$ so that choosing $\mathbf{d}_{\mu_k}$ by (7.49), $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha \mathbf{d}_{\mu_k}$, and $\mu_{k+1} = \rho\mu_k$ yields convergence to an optimal solution $\mathbf{x}^*$ to the original linear programming problem in $O(\sqrt{n}L)$ iterations. This could result in a polynomial-time affine scaling algorithm with complexity $O(n^3L)$. A simple and elegant proof is due to C. Roos and J.-Ph. Vial, similar to the one proposed by R. Monteiro and I. Adler for the primal-dual algorithm.

7.2 DUAL AFFINE SCALING ALGORITHM

Recall that the dual linear programming problem of problem (7.1) is

$$\text{Maximize} \quad \mathbf{b}^T\mathbf{w} \tag{7.50a}$$

$$\text{subject to} \quad \mathbf{A}^T\mathbf{w} + \mathbf{s} = \mathbf{c}, \qquad \mathbf{s} \geq \mathbf{0}, \quad \mathbf{w} \text{ unrestricted} \tag{7.50b}$$

Similar to the dual simplex method, the dual affine scaling algorithm starts with a dual feasible solution and takes steps towards optimality by progressively increasing the objective function while the dual feasibility is maintained in the process.

Notice that problem (7.50) contains both unrestricted variables $\mathbf{w} \in R^m$ and nonnegative variables $\mathbf{s} \in R^n$. In this case, $(\mathbf{w}; \mathbf{s})$ is defined to be an interior feasible solution if $\mathbf{A}^T\mathbf{w} + \mathbf{s} = \mathbf{c}$ and $\mathbf{s} > \mathbf{0}$. Also note that for $\mathbf{w}$-variables, there is no meaning of "centering" since they are unrestricted. But for $\mathbf{s}$-variables, we can treat them as we treat the $\mathbf{x}$-variables in the primal problem.

7.2.1 Basic Ideas of Dual Affine Scaling

The dual affine scaling algorithm also consists of three key parts, namely, starting with an interior dual feasible solution, moving to a better interior solution, and stopping with an optimal dual solution. We shall discuss the starting mechanisms and stopping rules in later sections. In this section we focus on the iterates.

Given that at the kth iteration, we have an interior dual solution $(\mathbf{w}^k; \mathbf{s}^k)$ such that $\mathbf{A}^T\mathbf{w}^k + \mathbf{s}^k = \mathbf{c}$ and $\mathbf{s}^k > \mathbf{0}$. Our objective is to find a good moving direction $(\mathbf{d}_w^k; \mathbf{d}_s^k)$ together with an appropriate step-length $\beta_k > 0$ such that a new interior solution $(\mathbf{w}^{k+1}; \mathbf{s}^{k+1})$ is generated by

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \beta_k \mathbf{d}_w^k \tag{7.51a}$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k + \beta_k \mathbf{d}_s^k \tag{7.51b}$$

which satisfies that

$$\mathbf{A}^T\mathbf{w}^{k+1} + \mathbf{s}^{k+1} = \mathbf{c} \tag{7.52a}$$

$$\mathbf{s}^{k+1} > \mathbf{0} \tag{7.52b}$$

and

$$\mathbf{b}^T\mathbf{w}^{k+1} \geq \mathbf{b}^T\mathbf{w}^k \tag{7.52c}$$

Plugging (7.51) into (7.52a) and remembering that $\mathbf{A}^T\mathbf{w}^k + \mathbf{s}^k = \mathbf{c}$, we have a requirement for the moving direction, namely,

$$\mathbf{A}^T\mathbf{d}_w^k + \mathbf{d}_s^k = \mathbf{0} \tag{7.53a}$$

In order to get better objective value, we plug (7.51a) into (7.52c), which results in another requirement for the moving direction:

$$\mathbf{b}^T\mathbf{d}_w^k \geq 0 \tag{7.53b}$$

To take care of (7.52b), the affine scaling method is applied. The basic idea is to recenter $\mathbf{s}^k$ at $\mathbf{e} = (1, 1, \ldots, 1)^T \in R^n$ in the transformed space such that the distance to each positivity wall is known. In this way, any movement within unit distance certainly preserves the positivity requirement.

Similar to what we did in the primal affine scaling algorithm, we define an affine scaling matrix $\mathbf{S}_k = \operatorname{diag}(\mathbf{s}^k)$ which is a diagonal matrix with s_i^k as its ith diagonal element. In this way, $\mathbf{S}_k^{-1}\mathbf{s}^k = \mathbf{e}$ and every $\mathbf{s}$-variable is transformed (or scaled) into a new variable $\mathbf{u} \geq \mathbf{0}$ such that

$$\mathbf{u} = \mathbf{S}_k^{-1}\mathbf{s} \tag{7.54a}$$

and

$$\mathbf{s} = \mathbf{S}_k\mathbf{u} \tag{7.54b}$$

Moreover, if $\mathbf{d}_u^k$ is a direction of cost improvement in the transformed space, then its corresponding direction in the original space is given by

$$\mathbf{d}_s^k = \mathbf{S}_k\mathbf{d}_u^k. \tag{7.54c}$$

Now we can study the iterates of the dual affine scaling algorithm in the transformed (or scaled) space. In order to synthesize a good moving direction in the transformed space, requirement (7.53a) implies that

$$\mathbf{A}^T\mathbf{d}_w^k + \mathbf{d}_s^k = \mathbf{0} \Rightarrow \mathbf{A}^T\mathbf{d}_w^k + \mathbf{S}_k\mathbf{d}_u^k = 0$$
$$\Rightarrow \mathbf{S}_k^{-1}\mathbf{A}^T\mathbf{d}_w^k + \mathbf{d}_u^k = 0 \Rightarrow \mathbf{S}_k^{-1}\mathbf{A}^T\mathbf{d}_w^k = -\mathbf{d}_u^k$$

Multiplying both sides by $\mathbf{A}\mathbf{S}_k^{-1}$ we get

$$\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T\mathbf{d}_w^k = -\mathbf{A}\mathbf{S}_k^{-1}\mathbf{d}_u^k$$

Assuming that $\mathbf{A}$ is of full row rank, we obtain

$$\mathbf{d}_w^k = -(\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{S}_k^{-1}\mathbf{d}_u^k \tag{7.55a}$$

By defining $\mathbf{Q}_k = (\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{S}_k^{-1}$, (7.55a) is simplified as

$$\mathbf{d}_w^k = -\mathbf{Q}_k\mathbf{d}_u^k \tag{7.55b}$$

The above equation says that $\mathbf{d}_w^k$ is actually determined by $\mathbf{d}_u^k$ in the transformed space. If we can find an appropriate direction $\mathbf{d}_u^k$ such that (7.53b) is satisfied, then we can achieve our goal. To do so, we simply let

$$\mathbf{d}_u^k = -\mathbf{Q}_k^T\mathbf{b} \tag{7.56a}$$

then we have

$$\mathbf{b}^T\mathbf{d}_w^k = \mathbf{b}^T\mathbf{Q}_k\mathbf{d}_u^k = \mathbf{b}^T\mathbf{Q}_k\mathbf{Q}_k^T\mathbf{b} = ||\mathbf{b}^T\mathbf{Q}_k||^2 \geq 0$$

Combining (7.56a) and (7.55b), we see that

$$\mathbf{d}_w^k = (\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{b} \tag{7.56b}$$

Consequently, from (7.53a), we have the following moving direction in the original space:

$$\mathbf{d}_s^k = -\mathbf{A}^T\mathbf{d}_w^k = -\mathbf{A}^T(\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{b} \tag{7.56c}$$

Once the moving direction $(\mathbf{d}_w^k; \mathbf{d}_s^k)$ is known, the step-length β_k is dictated by the positivity requirement of $\mathbf{s}^{k+1}$ as in the primal affine scaling algorithm, namely,

1. If $\mathbf{d}_s^k = \mathbf{0}$, then the dual problem has a constant objective value in its feasible domain and $(\mathbf{w}^k; \mathbf{s}^k)$ is dual optimal.
2. If $\mathbf{d}_s^k \geq \mathbf{0}$ (but $\neq \mathbf{0}$), then problem (7.50) is unbounded.
3. Otherwise,

$$\beta_k = \min_i \left\{ \frac{\alpha s_i^k}{-\left(d_s^k\right)_i} \;\middle|\; (d_s^k)_i < 0 \right\} \qquad \text{where} \quad 0 < \alpha < 1$$

Note that, similar to the way we defined dual estimates in the primal affine scaling algorithm, if we define

$$\mathbf{x}^k = -\mathbf{S}_k^{-2}\mathbf{d}_s^k \tag{7.57}$$

then $\mathbf{A}\mathbf{x}^k = \mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T\mathbf{d}_w^k = \mathbf{b}$. Therefore, $\mathbf{x}^k$ can be viewed as a "primal estimate" in the dual affine scaling algorithm. Once the primal estimate satisfies that $\mathbf{x}^k \geq \mathbf{0}$, then it becomes a primal feasible solution with a duality gap $\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k$. Moreover, if $\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k = 0$, then $(\mathbf{w}^k; \mathbf{s}^k)$ must be dual optimal and $\mathbf{x}^k$ primal optimal. This information can be used to define stopping rules for the dual affine scaling algorithm.

7.2.2 Dual Affine Scaling Algorithm

Based on the basic ideas discussed in the previous section, we outline a dual affine scaling algorithm here.

Step 1 (initialization): Set $k = 0$ and find a starting solution $(\mathbf{w}^0; \mathbf{s}^0)$ such that $\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 = \mathbf{c}$ and $\mathbf{s}^0 > \mathbf{0}$. (Details will be discussed later.)

Step 2 (obtain directions of translation): Let $\mathbf{S}_k = \text{diag}\,(\mathbf{s}^k)$ and compute

$$\mathbf{d}_w^k = (\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{b} \quad \text{and} \quad \mathbf{d}_s^k = -\mathbf{A}^T\mathbf{d}_w^k$$

Step 3 (check for unboundedness): If $\mathbf{d}_s^k = \mathbf{0}$, then STOP. $(\mathbf{w}^k; \mathbf{s}^k)$ is dual optimal. If $\mathbf{d}_s^k \geq \mathbf{0}$, then also STOP. The dual problem (7.50) is unbounded.

Step 4 (computation of the primal estimate): Compute the primal estimate as:

$$\mathbf{x}^k = -\mathbf{S}_k^{-2}\mathbf{d}_s^k$$

Step 5 (check for optimality): If $\mathbf{x}^k \geq \mathbf{0}$ and $\mathbf{c}^T\mathbf{x}_k - \mathbf{b}^T\mathbf{w}_k \leq \epsilon$, where ϵ is a preassigned small positive number, then STOP. $(\mathbf{w}^k; \mathbf{s}^k)$ is dual optimal and $\mathbf{x}^k$ is primal optimal. Otherwise, go to the next step.

Step 6 (computation of step-length): Compute the step-length

$$\beta_k = \min_i \left\{ \frac{\alpha s_i^k}{-(d_s^k)_i} \,\middle|\, (d_s^k)_i < 0 \right\} \qquad \text{where } 0 < \alpha < 1$$

Step 7 (move to a new solution): Update dual variables $(\mathbf{w}; \mathbf{s})$ by

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \beta_k \mathbf{d}_w^k$$
$$\mathbf{s}^{k+1} = \mathbf{s}^k + \beta_k \mathbf{d}_s^k$$

Set $k \leftarrow k+1$ and go to Step 2.

Now we present an example to illustrate the dual affine scaling algorithm.

Example 7.2

Consider the dual problem of Example 7.1 and solve it by using the dual affine scaling algorithm.

First note the dual problem is

$$\text{Maximize} \quad 15w_1 + 15w_2$$

$$\text{subject to} \quad \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{bmatrix} = \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$s_1, s_2, s_3, s_4 \geq 0$$

It is easy to verify that $\mathbf{w}^0 = [-3 \quad -3]^T$ and $\mathbf{s}^0 = [1 \quad 1 \quad 3 \quad 3]^T$ constitute an initial interior feasible solution. Hence,

$$\mathbf{S}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \qquad \mathbf{d}_w^0 = (\mathbf{A}\mathbf{S}_0^{-2}\mathbf{A}^T)^{-1}\mathbf{b} = \begin{bmatrix} 23.53211 \\ 34.67890 \end{bmatrix}$$

and

$$\mathbf{d}_s^0 = -\mathbf{A}^T\mathbf{d}_w^0 = \begin{bmatrix} -23.53211 \\ -11.14679 \\ -23.53211 \\ -34.67890 \end{bmatrix}$$

Then

$$\mathbf{x}^0 = -\mathbf{S}_0^{-2}\mathbf{d}_s^0 = (23.53211 \quad 11.14679 \quad 2.61467 \quad 3.85321)^T$$

Although $\mathbf{x}^0 \geq \mathbf{0}$, the duality gap is $\mathbf{c}^T\mathbf{x}^0 - \mathbf{b}^T\mathbf{w}^0 = 54.08257$, which is far bigger than zero. Hence the current solution is not optimal yet.

To calculate the step-length, we choose $\alpha = 0.99$. Consequently,

$$\beta_0 = \frac{0.99 \times 1}{23.53211} = 0.04207$$

Updating dual variables, we have

$$\mathbf{w}^1 = (-3 \quad -3)^T + 0.04207 \times (23.53211 \quad 34.67890)^T = (-2.01000 \quad -1.54105)^T \text{and}$$

$$\mathbf{s}^1 = (1 \quad 1 \quad 3 \quad 3)^T + 0.04207 \times (23.53211 \quad -11.146789 \quad -23.53211 \quad -34.67890)^T$$

$$= (0.01000 \quad 0.53105 \quad 2.01000 \quad 1.54105)^T$$

So far, we have finished one iteration of the dual affine scaling algorithm. Iterating again, we obtain

$$\mathbf{w}^2 = (-2.00962 \quad -1.10149)^T$$

$$\mathbf{s}^2 = [0.009624 \quad 0.00531 \quad 2.00962 \quad 1.01494]^T$$

$$\mathbf{x}^2 = [29.80444 \quad 14.80452 \quad 0.00001 \quad 0.19548]^T$$

This time, $\mathbf{x}^2 > \mathbf{0}$ and the duality gap has drastically reduced to

$$\mathbf{c}^T\mathbf{x}^2 - \mathbf{b}^T\mathbf{w}^2 = -44.80435 - (-45.36840) = 0.56404$$

which is clearly closer to zero. The reader may carry out more iterations and verify that the optimal value is assumed at $\mathbf{w}^* = (-2 \quad -1)^T$ and $\mathbf{s}^* = (0 \quad 0 \quad 2 \quad 1)^T$ with an optimal objective value of -45. The corresponding primal solution $\mathbf{x}^*$ is located at $(30 \quad 15 \quad 0 \quad 0)^T$.

7.2.3 Implementing the Dual Affine Scaling Algorithm

In this section we introduce two methods, the "big-M method" and "upper bound method," to find an initial dual feasible interior solution for the dual affine scaling algorithm. Then we discuss the stopping rules and report some computational experience regarding dual affine scaling.

Starting the dual affine scaling algorithm. The problem here is to find $(\mathbf{w}^0; \mathbf{s}^0)$ such that $\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 = \mathbf{c}$ and $\mathbf{s}^0 > \mathbf{0}$. Note that, in a special case, if $\mathbf{c} > \mathbf{0}$, then we can immediately choose $\mathbf{w}^0 = \mathbf{0}$ and $\mathbf{s}^0 = \mathbf{c}$ as an initial interior feasible solution for the dual affine scaling algorithm. Unfortunately, this special case does not happen every time, and we have to depend upon other methods to start the dual affine scaling algorithm.

Big-M Method. One of the most widely used methods to start the dual affine scaling is the big-M method. In this method, we add one more artificial variable, say w^a, and a large positive number M. Then consider the following "big-M" linear programming problem:

$$\begin{aligned} &\text{Maximize} \quad \mathbf{b}^T\mathbf{w} + Mw^a \\ &\text{subject to} \quad \mathbf{A}^T\mathbf{w} + \mathbf{p}w^a + \mathbf{s} = \mathbf{c} \\ &\qquad\qquad \mathbf{w}, w^a \text{ unrestricted} \quad \text{and} \quad \mathbf{s} \geq \mathbf{0} \end{aligned} \tag{7.58}$$

where $\mathbf{p} \in R^n$ is a column vector whose ith component, $i = 1, \dots, n$, is defined by

$$p_i = \begin{cases} 1 & \text{if } c_i \leq 0 \\ 0 & \text{if } c_i > 0 \end{cases}$$

Now, we define $\bar{c} = \max_i |c_i|$, set $\theta > 1$, and choose $\mathbf{w} = \mathbf{0}$, $w^a = -\theta\bar{c}$, and $\mathbf{s} = \mathbf{c} + \theta\bar{c}\mathbf{p}$. It is clearly seen that $(\mathbf{0}; -\theta\bar{c}; \mathbf{c} + \theta\bar{c}\mathbf{p})^T$ is feasible to the big-M problem (7.58) with $\mathbf{c} + \theta\bar{c}\mathbf{p} > \mathbf{0}$. Hence we have found an initial interior feasible solution to the big-M problem to start the dual affine scaling algorithm.

Note that w^a starts with $-\theta\bar{c} < 0$ and is forced to increase in the iterative process, since M is a large positive number. At some point of time, we expect to see that w^a becomes nonnegative unless the original problem (7.50) is infeasible. When w^a approaches or even crosses zero at the kth iteration, we can take $\hat{\mathbf{w}} = \mathbf{w}^k$ and $\hat{\mathbf{s}} = \mathbf{s}^k + \mathbf{p}w^a$ to start the dual affine scaling algorithm for the original dual linear programming problem (7.50). If w^a does not approach or cross zero, then it can be shown that the original problem (7.50) is infeasible. Showing this is left for the reader as an exercise.

Also note that both θ and M are responsible for the quantity of Mw^a. Their values could be "tweaked" simultaneously for numerical stability and robustness.

Upper Bound or Artificial Constraint Method. In this method, we assume that for a sufficiently large positive number M, one of the optimal solutions to the original primal linear programming problem (7.1) falls in the ball of $\mathbf{S}(0; M)$, and we consider a corresponding "upper-bounded" linear programming problem:

$$\begin{aligned} &\text{Minimize} \quad \mathbf{c}^T\mathbf{x} \\ &\text{subject to} \quad \mathbf{Ax} = \mathbf{b} \quad \text{and} \quad \mathbf{0} \leq \mathbf{x} \leq \mathbf{u} \end{aligned}$$

where $\mathbf{u} = [M \quad M \quad \cdots \quad M]^T \in R^n$. The additional upper-bound constraints are artificially added to create a dual problem with a trivial initial interior solution. Actually, the dual of the upper-bounded problem is given by

$$\begin{aligned} &\text{Maximize} \quad \mathbf{b}^T\mathbf{w} - \mathbf{u}^T\mathbf{v} \\ &\text{subject to} \quad \mathbf{A}^T\mathbf{w} + \mathbf{s} - \mathbf{v} = \mathbf{c}, \qquad \mathbf{s} \geq \mathbf{0}, \quad \mathbf{v} \geq \mathbf{0}, \quad \text{and} \quad \mathbf{w} \text{ unrestricted} \end{aligned}$$

Vector $\mathbf{v}$ is sometimes called the vector of *surplus variables*. Remembering the definition of $\bar{c}$ and θ in the previous section, we see that $\mathbf{w}^0 = \mathbf{0}$, $\mathbf{v}^0 = \theta\bar{c}\mathbf{e} > \mathbf{0}$, and $\mathbf{s}^0 = \mathbf{c} + \theta\bar{c}\mathbf{e} > \mathbf{0}$ form an interior feasible solution to the dual upper-bound problem. Subsequently, the dual affine scaling algorithm can be applied.

The success of this method depends upon the choice of M. It has to be sufficiently large to include at least one optimal solution to problem (7.50). If the original linear programming problem is unbounded, the choice of M becomes a real problem.

Stopping rules for dual affine scaling. For the dual affine scaling algorithm, we still use the K-K-T conditions for optimality test. Note that the dual feasibility is maintained by the algorithm throughout the entire iterative procedure. Hence we only need to check the primal feasibility and complementary slackness.

Combining (7.56c) and (7.57), we see that the primal estimate is given by

$$\mathbf{x}^k = -\mathbf{S}_k^{-2}\mathbf{d}_s^k = \mathbf{S}_k^{-2}\mathbf{A}^T(\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{b} \tag{7.59}$$

It is easy to see that the explicit constraints $\mathbf{A}\mathbf{x} = \mathbf{b}$ are automatically satisfied for any $\mathbf{x}^k$ which is defined according to formula (7.59). Therefore, if $\mathbf{x}^k \geq \mathbf{0}$, then it must be primal feasible. Also note that, if we convert problem (7.50) into a standard-form linear programming problem and apply the primal affine scaling to it, the associated dual estimates eventually result in formula (7.59).

Once we have reached dual feasibility at $\mathbf{w}^k$ and primal feasibility at $\mathbf{x}^k$, then the complementary slackness is provided by $\sigma_c = \mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k$. When σ_c is smaller than a given threshold, we can terminate the dual affine scaling algorithm.

Experiences with dual affine scaling. In light of the fact that the dual affine scaling algorithm is equivalent to the primal affine scaling algorithm applied to the dual problem, similar properties of convergence of the dual affine scaling can be established as we did for the primal affine scaling algorithm. The computational effort in each iteration of the dual affine scaling is about the same as in the primal affine scaling. To be more specific, the computational bottleneck of the primal affine scaling is to invert the matrix $\mathbf{A}\mathbf{X}_k^2\mathbf{A}^T$, and the bottleneck of dual affine scaling is to invert the matrix $\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T$. But these two matrices have exactly the same structure, although they use different scaling. Any numerical method, for example, Cholesky factorization, that improves the computational efficiency of one algorithm definitely improves the performance of the other one.

Based on the authors' experience, we have observed the following characteristics of the dual affine scaling algorithm:

1. For a variety of practical applications, we have noted a general tendency that the dual affine scaling algorithm converges faster than the primal affine scaling algorithm. However, the major drawback of the dual affine scaling algorithm is that it does not give good estimates of the primal variables.
2. The problem of losing feasibility in the primal affine scaling algorithm is not a serious problem for dual affine scaling. Actually, since the dual feasibility is maintained by choosing appropriate $\mathbf{d}_s^k = -\mathbf{A}^T\mathbf{d}_w^k$, one could approximate the inverse matrix of $\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T$ in computing $\mathbf{d}_w^k$ and still obtain a feasible direction $\mathbf{d}_s^k$. Hence the dual method is less sensitive to numerical truncation and round-off errors.

3. The dual affine scaling algorithm is still sensitive to dual degeneracy, but less sensitive to primal degeneracy.
4. The dual affine scaling algorithm improves its dual objective function in a very fast fashion. However, attaining primal feasibility is quite slow.

7.2.4 Improving Computational Complexity

Like the primal affine scaling algorithm, there is no theoretic proof showing the dual affine scaling is a polynomial-time algorithm. The philosophy of "staying away from the boundary" to gain faster convergence also applies here. In this section, we introduce the *power series method* and *logarithmic barrier function method* to improve the performance of the dual affine scaling algorithm.

Power series method. In applying the primal affine scaling algorithm, if we take the step-length α_k to be infinitesimally small, then the locus of $\mathbf{x}^k$ can be viewed as a *continuous curve* extending from the starting point $\mathbf{x}^0$ to the optimal solution $\mathbf{x}^*$. As a matter of fact, in the limit, we may pose the following equation:

$$\frac{d\mathbf{x}(\alpha)}{d\alpha} = \lim_{\alpha_k \to 0} \frac{\mathbf{x}^{k+1} - \mathbf{x}^k}{\alpha_k} = \mathbf{X}_k \mathbf{d}_y^k = -\mathbf{X}_k^2 \left(\mathbf{c} - \mathbf{A}^T \mathbf{w}^k\right)$$

as a first-order differential equation and attempt to find a solution function which describes the continuous curve. This smooth curve is called a *continuous trajectory*, and the moving direction $\mathbf{d}_x^k = \mathbf{X}_k \mathbf{d}_y^k$ at each iteration is simply the tangential direction (or first-order approximation) of this curve at an interior point of P.

As we can see from Figure 7.5, the first-order approximation deviates from the continuous trajectory easily. A higher-order approximation may stay closer to the continuous trajectory that leads to an optimal solution. The basic idea of the power series method is to find higher-order approximations of the continuous trajectory in terms of truncated power series.

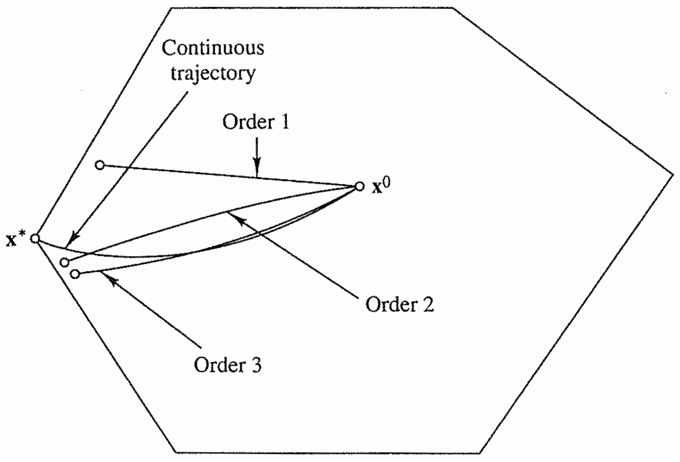

Figure 7.5

The same idea applies to the dual affine scaling algorithm. As a matter of fact, a continuous version of the dual affine scaling algorithm may be obtained by setting $\beta_k \to 0$ and solving Equation (7.51) as a system of ordinary differential equations. These equations will specify a vector field on the interior of the feasible domain. Our objective here is to generate higher-order approximations of the continuous trajectories by means of truncated power series.

Combining (7.56b) and (7.56c), we first write a system of differential equations corresponding to (7.51) as follows:

$$\frac{d\mathbf{w}(\beta)}{d\beta} = [\mathbf{A}\mathbf{S}(\beta)^{-2}\mathbf{A}^T]^{-1}\mathbf{b} \tag{7.60a}$$

$$\frac{d\mathbf{s}(\beta)}{d\beta} = -\mathbf{A}^T\frac{d\mathbf{w}(\beta)}{d\beta} \tag{7.60b}$$

with the initial conditions

$$\mathbf{w}(0) = \mathbf{w}^0 \quad \text{and} \quad \mathbf{s}(0) = \mathbf{s}^0 \tag{7.60c}$$

where $\mathbf{S}(\beta) = \text{diag}\,(\mathbf{s}(\beta))$ with $\mathbf{s}(\beta) = \mathbf{s}^0 + \beta\mathbf{d}_s > \mathbf{0}$.

In order to find the solution functions $\mathbf{w}(\beta)$ and $\mathbf{s}(\beta)$ which trace out a continuous trajectory, we may consider expanding them in power series at the current solution $\mathbf{w}(0) \equiv \mathbf{w}^0$ and $\mathbf{s}(0) \equiv \mathbf{s}^0$ such that

$$\mathbf{w}(\beta) = \mathbf{w}^0 + \sum_{j=1}^{\infty}\beta^j\left(\frac{1}{j!}\right)\left[\frac{d^j\mathbf{w}(\beta)}{d\beta^j}\right]_{\beta=0} = \sum_{j=0}^{\infty}\beta^j\left(\frac{1}{j!}\right)\left[\frac{d^j\mathbf{w}(\beta)}{d\beta^j}\right]_{\beta=0} \tag{7.61a}$$

and

$$\mathbf{s}(\beta) = \mathbf{s}^0 + \sum_{j=1}^{\infty}\beta^j\left(\frac{1}{j!}\right)\left[\frac{d^j\mathbf{s}(\beta)}{d\beta^j}\right]_{\beta=0} = \sum_{j=0}^{\infty}\beta^j\left(\frac{1}{j!}\right)\left[\frac{d^j\mathbf{s}(\beta)}{d\beta^j}\right]_{\beta=0} \tag{7.61b}$$

If we denote

$$f^{<i>} = \left(\frac{1}{i!}\right)\left[\frac{d^i f(\beta)}{d\beta^i}\right]_{\beta=0}$$

for a function $f(\beta)$, then (7.61) becomes

$$\mathbf{w}(\beta) = \sum_{j=0}^{\infty}\beta^j\mathbf{w}^{<j>} \tag{7.62a}$$

and

$$\mathbf{s}(\beta) = \sum_{j=0}^{\infty}\beta^j\mathbf{s}^{<j>} \tag{7.62b}$$

Equation (7.62) can be truncated at any desirable order to get an approximation of the continuous trajectory. Of course, higher-order truncation depicts the continuous trajectory more closely but at higher computational expense. In general, to obtain a kth

($k \geq 1$) order approximation of $\mathbf{w}(\beta)$ and $\mathbf{s}(\beta)$, we need to compute $\mathbf{w}^{<j>}$ and $\mathbf{s}^{<j>}$ for $j = 1, \ldots, k$. But Equation (7.60b) implies that

$$\mathbf{s}^{<j>} = -\mathbf{A}^T \mathbf{w}^{<j>}, \qquad \text{for } j = 1, 2, \ldots \tag{7.63}$$

Hence the key is to compute $\mathbf{w}^{<j>}$ for $j \geq 1$.

We start with Equation (7.60a) and denote $\mathbf{M}(\beta) = \mathbf{AS}(\beta)^{-2}\mathbf{A}^T$. Then we have

$$\mathbf{w}^{<1>} = \left[\mathbf{M}^{<0>}\right]^{-1} \mathbf{b} \tag{7.64}$$

where $\mathbf{M}^{<0>} = \mathbf{M}(0) = \mathbf{AS}(0)^{-2}\mathbf{A}^T$, in which $\mathbf{S}(0)^{-2}$ is the diagonal matrix with $1/(s_i^0)^2$ being its ith diagonal element, and

$$\mathbf{M}(\beta)\frac{d\mathbf{w}(\beta)}{d\beta} = \mathbf{b} \tag{7.65a}$$

Taking kth derivative on both sides, we have

$$\sum_{j=0}^{k} \left(\frac{k!}{j!(k-j)!}\right) \left[\frac{d^{(k-j)}\mathbf{M}(\beta)}{d\beta^{(k-j)}}\right] \left[\frac{d^{(j+1)}\mathbf{w}(\beta)}{d\beta^{(j+1)}}\right] = \mathbf{0} \tag{7.65b}$$

In other words, we have

$$\sum_{j=0}^{k} (j+1)\mathbf{M}^{<k-j>}\mathbf{w}^{<j+1>} = \mathbf{0} \tag{7.65c}$$

which further implies that

$$\mathbf{w}^{<k+1>} = -\left(\frac{1}{(k+1)}\right)[\mathbf{M}^{<0>}]^{-1}\left[\sum_{j=1}^{k} j\mathbf{M}^{<k-j+1>}\mathbf{w}^{<j>}\right], \qquad \text{for } k \geq 1 \tag{7.66}$$

Hence our focus is shifted to find $\mathbf{M}^{<k-j+1>}\mathbf{w}^{<j>}$ for $j = 1, 2, \ldots, k$ and compute $\mathbf{w}^{<k+1>}$ in a recursive fashion.

Remembering that $\mathbf{M}(\beta) = \mathbf{AS}(\beta)^{-2}\mathbf{A}^T$, we let $\mathbf{Y}(\beta) = \mathbf{S}(\beta)^{-2}$ be a diagonal matrix with $1/(s_i(\beta))^2$ as its ith diagonal element. Then, we have

$$\mathbf{M}^{<k-j+1>}\mathbf{w}^{<j>} = \mathbf{AY}^{<k-j+1>}\mathbf{A}^T\mathbf{w}^{<j>} = -\mathbf{AY}^{<k-j+1>}\mathbf{s}^{<j>} \tag{7.67}$$

and

$$\mathbf{w}^{<k+1>} = \left(\frac{1}{(k+1)}\right)[\mathbf{M}^{<0>}]^{-1}\left[\mathbf{A}\sum_{j=1}^{k} j\mathbf{Y}^{<k-j+1>}\mathbf{s}^{<j>}\right], \qquad \text{for } k \geq 1 \tag{7.68}$$

In order to compute $\mathbf{Y}^{<j>}$, we further define $\mathbf{Z}(\beta) = \mathbf{S}(\beta)^2$ to be a diagonal matrix with $(s_i(\beta))^2$ as its ith diagonal element. In this way, we see that

$$\mathbf{Z}(\beta)\mathbf{Y}(\beta) = \mathbf{I}.$$

Taking kth derivative results in

$$\sum_{j=0}^{k} \mathbf{Z}^{<k-j>}\mathbf{Y}^{<j>} = \mathbf{0}, \qquad \forall\, k \geq 1$$

Consequently,

$$\mathbf{Y}^{<k>} = -[\mathbf{Z}^{<0>}]^{-1}\left[\sum_{j=0}^{k-1} \mathbf{Z}^{<k-j>}\mathbf{Y}^{<j>}\right], \qquad \forall\, k \geq 1, \tag{7.69}$$

where $\mathbf{Y}^{<0>} = \mathbf{Y}(0)$, which is a diagonal matrix with $1/(s_i^0)^2$ being its ith diagonal element, and $\mathbf{Z}^{<0>} = \mathbf{Z}(0)$, which is a diagonal matrix with $(s_i^0)^2$ being its ith diagonal element.

Now, if we know $\mathbf{Z}^{<k-j>}$, then $\mathbf{Y}^{<k>}$ can be obtained by Equation (7.69). But this is an easy job, since $\mathbf{Z}(\beta) = \mathbf{S}(\beta)^2$. Taking kth derivative on both sides, we have

$$\mathbf{Z}^{<k>} = \sum_{j=0}^{k} \mathbf{S}^{<k-j>}\mathbf{S}^{<j>} \tag{7.70}$$

where $\mathbf{S}^{<j>}$ for each j is a diagonal matrix which takes the ith component of $\mathbf{s}^{<j>}$, i.e., $s_i^{<j>}$, as its ith diagonal element. In particular, $\mathbf{S}^{<0>} = \mathbf{S}(0)$, which is the diagonal matrix with s_i^0 as its ith diagonal element.

Summarizing what we have derived so far, we can start with the initial conditions

$$\mathbf{w}^{<0>} = \mathbf{w}(0) = \mathbf{w}^0; \qquad \mathbf{s}^{<0>} = \mathbf{s}(0) = \mathbf{s}^0$$

Proceed with

$$\mathbf{w}^{<1>} = [\mathbf{M}(0)]^{-1}\mathbf{b} = [\mathbf{A}\mathbf{S}_0^{-2}\mathbf{A}^T]^{-1}\mathbf{b},$$

and

$$\mathbf{s}^{<1>} = -\mathbf{A}^T\mathbf{w}^{<1>}$$

Remembering that $\mathbf{Z}^{<0>} = \mathbf{Z}(0) = \text{diag}\,((\mathbf{s}^0)^2)$ and $\mathbf{Y}^{<0>} = \mathbf{Y}(0) = [\mathbf{Z}(0)]^{-1}$, from Equation (7.70), we can derive $\mathbf{Z}^{<1>}$. Then, $\mathbf{Y}^{<1>}$ can be derived from Equation (7.69).

Now, recursively, we can compute $\mathbf{w}^{<2>}$ by Equation (7.68); compute $\mathbf{s}^{<2>}$ by

$$\mathbf{s}^{<k>} = -\mathbf{A}^T\mathbf{w}^{<k>} \tag{7.71}$$

for $k = 2$; compute $\mathbf{Z}^{<2>}$ by Equation (7.70); and compute $\mathbf{Y}^{<2>}$ by Equation (7.69). Proceeding with this recursive procedure, we can approximate $\mathbf{w}(\beta)$ and $\mathbf{s}(\beta)$ by a power series up to the desirable order.

Notice that the first-order power series approximation results in the same moving direction as the dual affine scaling algorithm at a current solution. In order to get higher-order approximation, additional matrix multiplications and additions are needed. However, there is only one matrix inversion $\mathbf{A}\mathbf{S}_0^{-2}\mathbf{A}^T$ involved, which is needed anyway by the first-order approximation. Since matrix inversion dominates the computational complexity of matrix multiplication and addition, it might be cost-effective to incorporate higher-order approximation. According to the authors' experience, we found the following characteristics of power series method:

1. Higher-order power series approximation becomes more insensitive to degeneracy.
2. Compared to the dual affine scaling algorithm, a power series approximation of order four or five seems to be able to cut the total number of iterations by half.
3. The power series method is more suitable for dense problems.

As far as the computational complexity is concerned, although it is conjectured that the power series method might result in a polynomial-time algorithm, no formal proof has yet been given.

Logarithmic barrier function method. Similar to that of the primal affine scaling algorithm, we can incorporate a barrier function, with extremely high values along the boundaries $\{(\mathbf{w}; \mathbf{s}) \mid s_j = 0, \text{ for } \textit{some } 1 \leq j \leq n\}$, into the original objective function. Now, consider the following nonlinear programming problem:

$$\text{Maximize} \quad F_\mu(\mathbf{w}, \mu) = \mathbf{b}^T\mathbf{w} + \mu \sum_{j=1}^{n} \log_e(c_j - \mathbf{A}_j^T\mathbf{w}) \tag{7.72a}$$

$$\text{subject to} \quad \mathbf{A}^T\mathbf{w} < \mathbf{c} \tag{7.72b}$$

where $\mu > 0$ is a scalar and $\mathbf{A}_j^T$ is the transpose of the jth column vector of matrix $\mathbf{A}$. Note that if $\mathbf{w}^*(\mu)$ is an optimal solution to problem (7.72), and if $\mathbf{w}^*(\mu)$ tends to a point $\mathbf{w}^*$ as μ approaches zero, then it follows that $\mathbf{w}^*$ is an optimal solution to the original dual linear programming problem.

The Lagrangian of problem (7.72) becomes

$$L(\mathbf{w}, \lambda) = \mathbf{b}^T\mathbf{w} + \mu \sum_{j=1}^{n} \log_e(c_j - \mathbf{A}_j^T\mathbf{w}) + \lambda^T(\mathbf{c} - \mathbf{A}^T\mathbf{w})$$

where λ is a vector of Lagrangian multipliers. Since $c_j - \mathbf{A}_j^T\mathbf{w} > 0$, the complementary slackness condition requires that $\lambda = \mathbf{0}$, and the associated K-K-T conditions become

$$\mathbf{b} - \mu\mathbf{A}\mathbf{S}^{-1}\mathbf{e} = \mathbf{0}, \quad \text{and} \quad \mathbf{s} > \mathbf{0}$$

Assuming that $\mathbf{w}^k$ and $\mathbf{s}^k = \mathbf{c} - \mathbf{A}^T\mathbf{w}^k > \mathbf{0}$ form a current interior dual feasible solution, we take one Newton step of the K-K-T conditions. This results in a moving direction

$$\Delta\mathbf{w} = \frac{1}{\mu}(\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{b} - (\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{S}_k^{-1}\mathbf{e} \tag{7.73}$$

Compare to $\mathbf{d}_w^k$ in (7.56b), we see that $-(\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{S}_k^{-1}\mathbf{e}$ is an additional term in the logarithmic barrier method to push a solution away from the boundary. Therefore, sometimes the logarithmic barrier function method is called *dual affine scaling with centering force.*

By appropriately choosing the barrier parameter μ and step-length at each iteration, C. Roos and J.-Ph. Vial provided a very simple and elegant polynomiality proof of the dual affine scaling with logarithmic barrier function. Their algorithm terminates in at most $O(\sqrt{n})$ iterations. Earlier, J. Renegar had derived a polynomial-time dual algorithm based upon the methods of centers and Newton's method for linear programming problems.

Instead of using (7.72), J. Renegar considers the following function:

$$f(\mathbf{w}, \beta) = t \log_e(\mathbf{b}^T\mathbf{w} - \beta) + \sum_{j=1}^{n} \log_e(c_j - \mathbf{A}_j^T\mathbf{w}) \tag{7.74}$$

where β is an underestimate for the minimum value of the dual objective function (like the idea used by Todd and Burrell) and t is allowed to be a free variable. A straightforward calculation of one Newton step at a current solution $(\mathbf{w}^k; \mathbf{s}^k)$ results in a moving direction

$$\mathbf{d}_w^k(\beta) = \gamma(\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{b} - (\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{S}_k^{-1}\mathbf{e} \tag{7.75}$$

where

$$\gamma = \frac{\mathbf{b}^T(\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{S}_k^{-1}\mathbf{e} + \mathbf{b}^T\mathbf{w}^k - \beta}{(\mathbf{b}^T\mathbf{w}^k - \beta)^2/t + \mathbf{b}^T(\mathbf{A}\mathbf{S}_k^{-2}\mathbf{A}^T)^{-1}\mathbf{b}}$$

By carefully choosing values of t and a sequence of better estimations $\{\beta^k\}$, J. Renegar showed that his dual method converges in $O(\sqrt{n}L)$ iterations and results in a polynomial-time algorithm of a total complexity $O(n^{3.5}L)$ arithmetic operations. Subsequently, P. M. Vaidya improved the complexity to $O(n^3L)$ arithmetic operations. The relationship between Renegar's method and the logarithmic barrier function method can be clearly seen by comparing (7.73) and (7.75).

7.3 THE PRIMAL-DUAL ALGORITHM

As in the simplex approach, in addition to primal affine scaling and dual affine scaling, there is a primal-dual algorithm. The primal-dual interior-point algorithm is based on the logarithmic barrier function approach. The idea of using the logarithmic barrier function method for convex programming problems can be traced back to K. R. Frisch in 1955. After Karmarkar's algorithm was introduced in 1984, the logarithmic barrier function method was reconsidered for solving linear programming problems. P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright used this method to develop a projected Newton barrier method and showed an equivalence to Karmarkar's projective scaling algorithm in 1985. N. Megiddo provided a theoretical analysis for the logarithmic barrier method and proposed a primal-dual framework in 1986. Using this framework, M. Kojima, S. Mizuno, and A. Yoshise presented a polynomial-time primal-dual algorithm for linear programming problems in 1987. Their algorithm was shown to converge in at most $O(nL)$ iterations with a requirement of $O(n^3)$ arithmetic operations per iteration. Hence the total complexity is $O(n^4L)$ arithmetic operations. Later, R. C. Monteiro and I. Adler refined the primal-dual algorithm to converge in at most $O(\sqrt{n}L)$ iterations with $O(n^{2.5})$ arithmetic operations required per iteration, resulting in a total of $O(n^3L)$ arithmetic operations.

7.3.1 Basic Ideas of the Primal-Dual Algorithm

Consider a standard-form linear program:

$$\begin{aligned} &\text{Minimize} \quad \mathbf{c}^T\mathbf{x} \\ &\text{subject to} \quad \mathbf{Ax} = \mathbf{b}, \qquad \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{P}$$

and its dual:

$$\begin{aligned} &\text{Maximize} \quad \mathbf{b}^T\mathbf{w} \\ &\text{subject to} \quad \mathbf{A}^T\mathbf{w} + \mathbf{s} = \mathbf{c}, \qquad \mathbf{s} \geq \mathbf{0}, \quad \mathbf{w} \text{ unrestricted} \end{aligned} \tag{D}$$

We impose the following assumptions for the primal-dual algorithm:

(A1) The set $\mathbf{S} \equiv \{\mathbf{x} \in R^n \mid \mathbf{Ax} = \mathbf{b}, \mathbf{x} > \mathbf{0}\}$ is nonempty.
(A2) The set $\mathbf{T} \equiv \{(\mathbf{w}; \mathbf{s}) \in R^m \times R^n \mid \mathbf{A}^T\mathbf{w} + \mathbf{s} = \mathbf{c}, \mathbf{s} > \mathbf{0}\}$ is nonempty.
(A3) The constraint matrix $\mathbf{A}$ has full row rank.

Under these assumptions, it is clearly seen from the duality theorem that problems (P) and (D) have optimal solutions with a common value. Moreover, the sets of the optimal solutions of (P) and (D) are bounded.

Note that, for $\mathbf{x} > \mathbf{0}$ in (P), we may apply the logarithmic barrier function technique, and consider the following family of nonlinear programming problems (P_μ):

$$\begin{aligned} &\text{Minimize} \quad \mathbf{c}^T\mathbf{x} - \mu \sum_{j=1}^{n} \log_e x_j \\ &\text{subject to} \quad \mathbf{Ax} = \mathbf{b}, \qquad \mathbf{x} > \mathbf{0} \end{aligned}$$

where $\mu > 0$ is a *barrier* or *penalty* parameter.

As $\mu \to 0$, we would expect the optimal solutions of problem (P_μ) to converge to an optimal solution of the original linear programming problem (P). In order to prove it, first observe that the objective function of problem (P_μ) is a strictly convex function, hence we know (P_μ) has at most one global minimum. The convex programming theory further implies that the global minimum, if it exists, is completely characterized by the Kuhn-Tucker conditions:

$$\mathbf{Ax} = \mathbf{b}, \qquad \mathbf{x} > \mathbf{0} \qquad \text{(primal feasibility)} \tag{7.76a}$$

$$\mathbf{A}^T\mathbf{w} + \mathbf{s} = \mathbf{c}, \qquad \mathbf{s} > \mathbf{0} \qquad \text{(dual feasibility)} \tag{7.76b}$$

$$\mathbf{XSe} - \mu\mathbf{e} = \mathbf{0} \qquad \text{(complementary slackness)} \tag{7.76c}$$

where $\mathbf{X}$ and $\mathbf{S}$ are diagonal matrices using the components of vectors $\mathbf{x}$ and $\mathbf{s}$ as diagonal elements, respectively.

Under assumptions (A1) and (A2) and assuming that (P) has a bounded feasible region, we see problem (P_μ) is indeed feasible and assumes a unique minimum at $\mathbf{x}(\mu)$, for each $\mu > 0$. Consequently, the system (7.76) has a unique solution $(\mathbf{x}; \mathbf{w}; \mathbf{s}) \in R^n \times R^m \times R^n$. Hence we have the following lemma:

Lemma 7.5. Under the assumptions (A1) and (A2), both problem (P_μ) and system (7.76) have a unique solution.

Observe that system (7.76) also provides the necessary and sufficient conditions (the K-K-T conditions) for $(\mathbf{w}(\mu); \mathbf{s}(\mu))$ being a maximum solution of the following program (D_μ):

$$\text{Maximize} \quad \mathbf{b}^T\mathbf{w} + \mu \sum_{j=1}^{n} \log_e s_j$$

$$\text{subject to} \quad \mathbf{A}^T\mathbf{w} + \mathbf{s} = \mathbf{c}, \qquad \mathbf{s} > \mathbf{0}, \quad \mathbf{w} \text{ unrestricted}$$

Note that Equation (7.76c) can be written componentwise as

$$x_j s_j = \mu, \qquad \text{for } j = 1, \ldots, n \tag{7.76c$'$}$$

Therefore, when the assumption (A3) is imposed, $\mathbf{x}$ uniquely determines $\mathbf{w}$ from Equations (7.76c′) and (7.76b). We let $(\mathbf{x}(\mu); \mathbf{w}(\mu); \mathbf{s}(\mu))$ denote the unique solution to system (7.76) for each $\mu > 0$. Obviously, we see $\mathbf{x}(\mu) \in \mathbf{S}$ and $(\mathbf{w}(\mu); \mathbf{s}(\mu)) \in \mathbf{T}$. Moreover, the duality gap becomes

$$\begin{aligned} g(\mu) &= \mathbf{c}^T\mathbf{x}(\mu) - \mathbf{b}^T\mathbf{w}(\mu) \\ &= (\mathbf{c}^T - \mathbf{w}(\mu)^T\mathbf{A})\mathbf{x}(\mu) \\ &= \mathbf{s}(\mu)^T\mathbf{x}(\mu) = n\mu \end{aligned} \tag{7.77}$$

Therefore, as $\mu \to 0$, the duality gap $g(\mu)$ converges to zero. This implies that $\mathbf{x}(\mu)$ and $\mathbf{w}(\mu)$ indeed converge to the optimal solutions of problems (P) and (D), respectively. Hence we have the following result:

Lemma 7.6. Under the assumptions (A1)–(A3), as $\mu \to 0$, $\mathbf{x}(\mu)$ converges to the optimal solution of program (P) and $(\mathbf{w}(\mu); \mathbf{s}(\mu))$ converges to the optimal solution of program (D).

For $\mu > 0$, we let Γ denote the curve, or path, consisting of the solutions of system (7.76), i.e.,

$$\Gamma = \{(\mathbf{x}(\mu); \mathbf{w}(\mu); \mathbf{s}(\mu)) \mid (\mathbf{x}(\mu); \mathbf{w}(\mu); \mathbf{s}(\mu)) \quad \text{solves (7.76) for some } \mu > 0\} \tag{7.78}$$

As $\mu \to 0$, the path Γ leads to a pair of primal optimal solution $\mathbf{x}^*$ and dual optimal solution $(\mathbf{w}^*; \mathbf{s}^*)$. Thus following the path Γ serves as a theoretical model for a class of primal-dual interior-point methods for linear programming. For this reason, people may classify the primal-dual approach as a *path-following* approach.

Given an initial point $(\mathbf{x}^0; \mathbf{w}^0; \mathbf{s}^0) \in \mathbf{S} \times \mathbf{T}$, the primal-dual algorithm generates a sequence of points $\{(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k) \in \mathbf{S} \times \mathbf{T}\}$ by appropriately choosing a moving direction

$(\mathbf{d}_x^k; \mathbf{d}_w^k; \mathbf{d}_s^k)$ and step-length β^k at each iteration. To measure a "deviation" from the curve Γ at each $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k)$, we introduce the following notations, for $k = 0, 1, 2, \ldots,$

$$\phi_i^k = x_i^k s_i^k, \qquad \text{for } i = 1, 2, \ldots, n \tag{7.79a}$$

$$\phi_{\text{ave}}^k = \frac{1}{n}\sum_{i=1}^{n} \phi_i^k \tag{7.79b}$$

$$\phi_{\min}^k = \min\{\phi_i^k; \quad i = 1, 2, \ldots, n\} \tag{7.79c}$$

$$\theta^k = \frac{\phi_{\text{ave}}^k}{\phi_{\min}^k} \tag{7.79d}$$

Obviously, we see that $\theta^k \geq 1$ and $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k) \in \Gamma$ if and only if $\theta^k = 1$. We shall see in later sections, when the deviation θ^0 at the initial point $(\mathbf{x}^0; \mathbf{w}^0; \mathbf{s}^0) \in \mathbf{S} \times \mathbf{T}$ is large, the primal-dual algorithm reduces not only the duality gap but also the deviation. With suitably chosen parameters, the sequence of points $\{(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k) \in \mathbf{S} \times \mathbf{T}\}$ generated by the primal-dual algorithm satisfy the inequalities

$$\mathbf{c}^T\mathbf{x}^{k+1} - \mathbf{b}^T\mathbf{w}^{k+1} = (1 - 2/(n\theta^k))(\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k) \tag{7.80a}$$

$$\theta^{k+1} - 2 \leq (1 - 1/(n+1))(\theta^k - 2), \qquad \text{if } 2 < \theta^k \tag{7.80b}$$

$$\theta^{k+1} \leq 2, \qquad \text{if } \theta^k \leq 2 \tag{7.80c}$$

The first inequality (7.80a) ensures that the duality gap decreases monotonically. With the remaining two inequalities we see the deviation θ^k becomes smaller than 3 in at most $O(n \log \theta^0)$ iterations, and then the duality gap converges to 0 linearly with the convergence rate at least $(1 - 2/(3n))$.

7.3.2 Direction and Step-Length of Movement

We are now in a position to develop the key steps of the primal-dual algorithm. Let us begin by synthesizing a direction of translation (moving direction) $(\mathbf{d}_x^k; \mathbf{d}_w^k; \mathbf{d}_s^k)$ at a current point $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k)$ such that the translation is made along the curve Γ to a new point $(\mathbf{x}^{k+1}; \mathbf{w}^{k+1}; \mathbf{s}^{k+1})$. This task is taken care of by applying the famous Newton's method to the system of equations (7.76a)–(7.76c).

Newton direction. Newton's method is one of the most commonly used techniques for finding a root of a system of nonlinear equations via successively approximating the system by linear equations. To be more specific, suppose that $F(\mathbf{z})$ is a nonlinear mapping from R^n to R^n and we need to find a $\mathbf{z}^* \in R^n$ such that $F(\mathbf{z}^*) = \mathbf{0}$. By using the multivariable Taylor series expansion (say at $\mathbf{z} = \mathbf{z}^k$), we obtain a linear approximation:

$$F(\mathbf{z}^k + \Delta z) \approx F(\mathbf{z}^k) + \mathbf{J}(\mathbf{z}^k)\Delta z \tag{7.81}$$

where $\mathbf{J}(\mathbf{z}^k)$ is the Jacobian matrix whose (i, j)th element is given by

$$\left[\frac{\partial F_i(\mathbf{z})}{\partial z_j}\right]_{\mathbf{z}=\mathbf{z}^k}$$

and Δz is a translation vector. As the left-hand side of (7.81) evaluates at a root of $F(\mathbf{z}) = \mathbf{0}$, we have a linear system

$$\mathbf{J}(\mathbf{z}^k)\Delta\mathbf{z} = -F(\mathbf{z}^k) \tag{7.82}$$

A solution vector of equation (7.82) provides one Newton iterate from $\mathbf{z}^k$ to $\mathbf{z}^{k+1} = \mathbf{z}^k + \mathbf{d}_z^k$ with a *Newton direction* $\mathbf{d}_z^k$ and a unit step-length. When $\mathbf{J}(\mathbf{z}^*)$ is nonsingular and the starting point $\mathbf{z}^0$ is "close enough" to $\mathbf{z}^*$, Newton's method converges quadratically to $\mathbf{z}^*$. But this spectacular convergence rate is only a "local" behavior. For a general nonlinear mapping $F(\mathbf{z})$, if $\mathbf{z}^0$ is not close enough to $\mathbf{z}^*$, the Newton iteration may diverge hopelessly.

Let us focus on the nonlinear system (7.76a–c). Assume that we are at a point $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k)$ for some $\mu^k > 0$, such that $\mathbf{x}^k, \mathbf{s}^k > \mathbf{0}$. The Newton direction $(\mathbf{d}_x^k; \mathbf{d}_w^k; \mathbf{d}_s^k)$ is determined by the following system of linear equations:

$$\begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}^T & \mathbf{I} \\ \mathbf{S}_k & \mathbf{0} & \mathbf{X}_k \end{bmatrix} \begin{bmatrix} \mathbf{d}_x^k \\ \mathbf{d}_w^k \\ \mathbf{d}_s^k \end{bmatrix} = - \begin{bmatrix} \mathbf{A}\mathbf{x}^k - \mathbf{b} \\ \mathbf{A}^T\mathbf{w}^k + \mathbf{s}^k - \mathbf{c} \\ \mathbf{X}_k\mathbf{S}_k\mathbf{e} - \mu^k\mathbf{e} \end{bmatrix} \tag{7.83}$$

where $\mathbf{X}_k$ and $\mathbf{S}_k$ are the diagonal matrices formed by $\mathbf{x}^k$ and $\mathbf{s}^k$, respectively. Multiplying it out, we have

$$\mathbf{A}\mathbf{d}_x^k = \mathbf{t}^k \tag{7.84a}$$

$$\mathbf{A}^T\mathbf{d}_w^k + \mathbf{d}_s^k = \mathbf{u}^k \tag{7.84b}$$

$$\mathbf{S}_k\mathbf{d}_x^k + \mathbf{X}_k\mathbf{d}_s^k = \mathbf{v}^k \tag{7.84c}$$

where

$$\mathbf{t}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k, \qquad \mathbf{u}^k = \mathbf{c} - \mathbf{A}^T\mathbf{w}^k - \mathbf{s}^k, \quad \text{and} \quad \mathbf{v}^k = \mu^k\mathbf{e} - \mathbf{X}_k\mathbf{S}_k\mathbf{e} \tag{7.85}$$

Notice that if $\mathbf{x}^k \in \mathbf{S}$ and $(\mathbf{w}^k; \mathbf{s}^k) \in \mathbf{T}$, then $\mathbf{t}^k = 0$ and $\mathbf{u}^k = 0$ correspondingly.

To solve system (7.83), we multiply both sides of Equation (7.84b) by $\mathbf{A}\mathbf{X}_k\mathbf{S}_k{}^{-1}$. Then, we have

$$\mathbf{A}\mathbf{X}_k\mathbf{S}_k^{-1}\mathbf{A}^T\mathbf{d}_w^k = \mathbf{A}\mathbf{X}_k\mathbf{S}_k^{-1}\mathbf{u}^k - \mathbf{A}\mathbf{X}_k\mathbf{S}_k^{-1}\mathbf{d}_s^k \tag{7.86}$$

Now from Equation (7.84c), we have

$$\mathbf{d}_s^k = \mathbf{X}_k^{-1}\mathbf{v}^k - \mathbf{X}_k^{-1}\mathbf{S}_k\mathbf{d}_x^k. \tag{7.87}$$

Following (7.85), we denote $\mathbf{X}_k^{-1}\mathbf{v}^k = \mu^k\mathbf{X}_k^{-1}\mathbf{e} - \mathbf{S}_k\mathbf{e}$ as $\mathbf{p}^k$. Using Equation (7.84a) in the above equation would produce

$$\mathbf{A}\mathbf{X}_k\mathbf{S}_k^{-1}\mathbf{d}_s^k = \mathbf{A}\mathbf{X}_k\mathbf{S}_k^{-1}\mathbf{p}^k - \mathbf{t}^k \tag{7.88}$$

Substituting Equation (7.88) back into Equation (7.86) yields

$$\mathbf{d}_w^k = [\mathbf{A}\mathbf{X}_k\mathbf{S}_k^{-1}\mathbf{A}^T]^{-1}\left(\mathbf{A}\mathbf{X}_k\mathbf{S}_k^{-1}(\mathbf{u}^k - \mathbf{p}^k) + \mathbf{t}^k\right) \tag{7.89a}$$

where $\mathbf{X}_k\mathbf{S}_k^{-1}$ is a positive definite diagonal matrix.

Once $\mathbf{d}_w^k$ is obtained, $\mathbf{d}_x^k$ and $\mathbf{d}_s^k$ can be readily computed by

$$\mathbf{d}_s^k = \mathbf{u}^k - \mathbf{A}^T\mathbf{d}_w^k \tag{7.89b}$$

and

$$\mathbf{d}_x^k = \mathbf{X}_k\mathbf{S}_k^{-1}[\mathbf{p}^k - \mathbf{d}_s^k] \tag{7.89c}$$

Again, for $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k) \in \mathbf{S} \times \mathbf{T}$, Equations (7.89a)–(7.89c) are simplified as

$$\mathbf{d}_w^k = -[\mathbf{A}\hat{\mathbf{D}}_k^2\mathbf{A}^T]^{-1}\mathbf{A}\mathbf{S}_k^{-1}\mathbf{v}^k \tag{7.90a}$$

$$\mathbf{d}_s^k = -\mathbf{A}^T\mathbf{d}_w^k \tag{7.90b}$$

$$\mathbf{d}_x^k = \mathbf{S}_k^{-1}[\mathbf{v}^k - \mathbf{X}_k\mathbf{d}_s^k] \tag{7.90c}$$

where $\hat{\mathbf{D}}_k^2 = \mathbf{X}_k\mathbf{S}_k^{-1}$ and $\hat{\mathbf{D}}_k = \text{diag}\left(\sqrt{\mathbf{x}^k/\mathbf{s}^k}\right)$.

It is important to note that $\mathbf{d}_x^k$, $\mathbf{d}_w^k$, and $\mathbf{d}_s^k$ in (7.90) are closely related. If we denote vector

$$\mathbf{r}^k(\mu) = \mathbf{X}_k^{-1}\hat{\mathbf{D}}_k\mathbf{v}^k(\mu) \tag{7.91a}$$

and matrix

$$\mathbf{Q} = \hat{\mathbf{D}}_k\mathbf{A}^T(\mathbf{A}\hat{\mathbf{D}}_k^2\mathbf{A}^T)^{-1}\mathbf{A}\hat{\mathbf{D}}_k$$

then $(\mathbf{d}_x^k; \mathbf{d}_w^k; \mathbf{d}_s^k)$ can be rewritten as

$$\mathbf{d}_x^k = \hat{\mathbf{D}}_k(\mathbf{I} - \mathbf{Q})\mathbf{r}^k(\mu) \tag{7.90a$'$}$$

$$\mathbf{d}_w^k = -(\mathbf{A}\hat{\mathbf{D}}_k^2\mathbf{A}^T)^{-1}\mathbf{A}\hat{\mathbf{D}}_k\mathbf{r}^k(\mu) \tag{7.90b$'$}$$

$$\mathbf{d}_s^k = \hat{\mathbf{D}}_k^{-1}\mathbf{Q}\mathbf{r}^k(\mu) \tag{7.90c$'$}$$

Since matrix $\mathbf{Q}$ is the orthogonal projection matrix onto the column space of matrix $\hat{\mathbf{D}}_k\mathbf{A}^T$, we see that

$$\hat{\mathbf{D}}_k^{-1}\mathbf{d}_x^k + \hat{\mathbf{D}}_k\mathbf{d}_s^k = \mathbf{r}^k(\mu), \tag{7.91b}$$

$$(\mathbf{d}_x^k)^T\mathbf{d}_s^k = (\hat{\mathbf{D}}_k^{-1}\mathbf{d}_x^k)^T\hat{\mathbf{D}}_k\mathbf{d}_s^k = 0 \tag{7.91c}$$

After obtaining a Newton direction at the kth iteration, the primal-dual algorithm iterates to a new point according to the following translation:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \beta^k\mathbf{d}_x^k$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \beta^k\mathbf{d}_w^k$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k + \beta^k\mathbf{d}_s^k$$

with an appropriately chosen step-length β^k at the kth iteration such that $\mathbf{x}^{k+1} \in \mathbf{S}$ and $(\mathbf{w}^{k+1}; \mathbf{s}^{k+1}) \in \mathbf{T}$.

Step-length and penalty parameter. When $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k) \in \mathbf{S} \times \mathbf{T}$, the primal-dual algorithm needs two parameters σ and τ, such that $0 \leq \tau < \sigma < 1$, to control the penalty (or barrier) parameter μ^k and the step-length β^k at the kth iteration.

For the penalty parameter, remembering the notations defined in (7.79), since we want to reduce the duality gap, $n\phi^k_{\text{ave}}$, we may choose the penalty parameter to be a smaller number by setting

$$\mu^k = \sigma\phi^k_{\text{ave}} \tag{7.92}$$

In this way, definition (7.85) implies that $\mathbf{v}^k \leq \mathbf{0}$.

As to the step-length β^k, the choice is closely related to the complementary slackness. Note that Equations (7.84c) and (7.85) imply that $x_i^k d^k_{s_i} + s_i^k d^k_{x_i} = \mu^k - \phi_i^k$. Hence the complementary slackness varies quadratically in terms of the step-length β, since

$$\begin{aligned} \phi_i^k(\beta) &\equiv \left(x_i^k + \beta d^k_{x_i}\right)\left(s_i^k + \beta d^k_{s_i}\right) \\ &= \phi_i^k + \beta\left(\mu^k - \phi_i^k\right) + \beta^2\left(d^k_{x_i} d^k_{s_i}\right), \qquad i = 1, 2, \ldots, n \end{aligned} \tag{7.93a}$$

Moreover, since $(\mathbf{d}^k_x)^T \mathbf{d}^k_s = 0$, we see the average complementary slackness, and hence the duality gap, changes linearly in β, i.e.,

$$\phi^k_{\text{ave}}(\beta) = \left(\mathbf{x}^k + \beta\mathbf{d}^k_x\right)^T\left(\mathbf{s}^k + \beta\mathbf{d}^k_s\right)/n = \phi^k_{\text{ave}} + \beta\left(\mu^k - \phi^k_{\text{ave}}\right) \tag{7.93b}$$

Ignoring the quadratic term in (7.93a) and lowering the value $\mu^k = \sigma\phi^k_{\text{ave}}$ by a factor $\tau < \sigma$, we can define a linear function

$$\psi^k(\beta) = \phi^k_{\min} + \beta(\tau\phi^k_{\text{ave}} - \phi^k_{\min}) \tag{7.94}$$

The function $\phi_i^k(\beta)$ can be either convex or concave depending upon the value of $d^k_{x_i} d^k_{s_i}$. For the convex piece, since $d^k_{x_i} d^k_{s_i} \geq 0$, the curve of $\phi_i^k(\beta)$ lies above the curve of $\psi^k(\beta)$ for $0 \leq \beta \leq 1$. However, a concave piece of $\phi_i^k(\beta)$ may intersect $\psi^k(\beta)$ as shown in Figure 7.6. In order to control the deviation parameter θ^k while reducing the complementary slackness, we choose

$$\begin{aligned} \alpha^k = \max\ \Big\{\overline{\beta}\ \big|\ \phi_i^k(\beta) \geq \psi^k(\beta) \quad &\text{for all} \quad \beta \in (0, \overline{\beta}), \\ &0 < \overline{\beta} < 1, \quad \text{and} \quad i = 1, \ldots, n\Big\} \end{aligned} \tag{7.95}$$

Then the step-length β^k at the kth iteration is defined by

$$\beta^k = \min\left\{1, \alpha^k\right\} \tag{7.96}$$

The geometrical significance of α^k and β^k is depicted in Figure 7.6. It is clearly seen from the figure that the choice of β^k depends on the choice of $0 < \tau < 1$ to ensure the existence of α^k.

Note that when $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k) \in \mathbf{S} \times \mathbf{T}$, since $(\mathbf{d}^k_x; \mathbf{d}^k_w; \mathbf{d}^k_s)$ is a solution to (7.84) with $\mathbf{t}^k = \mathbf{0}$ and $\mathbf{u}^k = \mathbf{0}$, we know that $\mathbf{A}\mathbf{x}^{k+1} = \mathbf{b}$ and $\mathbf{A}^T\mathbf{w} + \mathbf{s} = \mathbf{c}$. Moreover, the definition of α^k in (7.95) further implies that $\mathbf{x}^{k+1} > \mathbf{0}$ and $\mathbf{s}^{k+1} > \mathbf{0}$. In other words, $(\mathbf{x}^{k+1}; \mathbf{w}^{k+1}; \mathbf{s}^{k+1}) \in \mathbf{S} \times \mathbf{T}$.

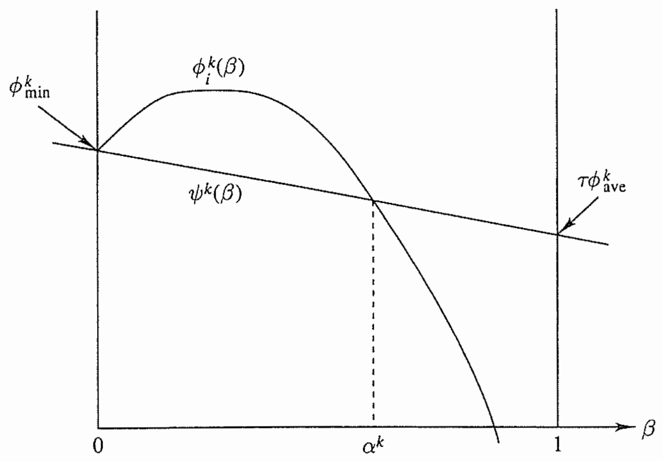

Figure 7.6

7.3.3 Primal-Dual Algorithm

We now are ready to state the primal-dual algorithm as following:

Step 1 (initialization): Set $k = 0$ and find a starting solution $(\mathbf{x}^0; \mathbf{w}^0; \mathbf{s}^0) \in \mathbf{S} \times \mathbf{T}$. Let $\epsilon > 0$ be a tolerance for duality gap and σ, τ be control parameters such that $0 \leq \tau < \sigma < 1$.

Step 2 (checking for optimality): If $\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k < \epsilon$, then stop. Otherwise, continue.

Step 3 (finding the direction of translation): Define ϕ^k_{ave} and $\phi^k_{\min}$ by (7.79). Let $\mu^k = \sigma\phi^k_{\text{ave}}$ and $\mathbf{v}^k = \mu^k\mathbf{e} - \mathbf{X}_k\mathbf{S}_k\mathbf{e}$. Compute $\mathbf{d}^k_x$, $\mathbf{d}^k_w$, $\mathbf{d}^k_s$ according to (7.90).

Step 4 (calculating step-length): Compute α^k by (7.95) and β^k by (7.96).

Step 5 (moving to a new solution): Let

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \beta^k\mathbf{d}^k_x$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \beta^k\mathbf{d}^k_w$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k + \beta^k\mathbf{d}^k_s$$

Set $k \leftarrow k + 1$ and go to Step 2.

7.3.4 Polynomial-Time Termination

Unlike the pure primal affine scaling and the pure dual affine scaling, the primal-dual algorithm is a polynomial-time algorithm. The well-chosen step-length β^k at each iteration leads to the nice convergence results:

Theorem 7.3. If the step-length $\beta^k < 1$ at the kth iteration, then

$$\beta^k \geq \frac{4(\sigma - \tau)}{n(1 - 2\sigma + \theta^k\sigma^2)} \geq \frac{4(\sigma - \tau)}{n(1 + \sigma^2)\theta^k} \tag{7.97a}$$

$$\mathbf{c}^T\mathbf{x}^{k+1} - \mathbf{b}^T\mathbf{w}^{k+1} = (1-(1-\sigma)\beta^k)(\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k) \tag{7.97b}$$

$$\theta^{k+1} - \sigma/\tau \le (1-\nu)(\theta^k - \sigma/\tau) \qquad \text{if } \sigma/\tau < \theta^k \tag{7.97c}$$

$$\theta^{k+1} \le \sigma/\tau \qquad \text{if } \theta^k \le \sigma/\tau \tag{7.97d}$$

where

$$\nu = \frac{4(\sigma-\tau)\tau}{n(1+\sigma^2)+4(\sigma-\tau)\tau}$$

On the other hand, if $\beta^k = 1$, then

$$\mathbf{c}^T\mathbf{x}^{k+1} - \mathbf{b}^T\mathbf{w}^{k+1} = \sigma(\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k) \tag{7.98a}$$

$$\theta^{k+1} \le \sigma/\tau \tag{7.98b}$$

Proof. Let us define

$$\eta^k(\beta) = \phi^k_{\min} + \beta\left(\mu^k - \phi^k_{\min}\right) - \left\|\mathbf{r}^k(\mu^k)\right\|^2\left(\beta^2/4\right) \tag{7.99}$$

where $\mathbf{r}^k(\mu^k)$ is defined by (7.91a). What we want to prove first is that

$$\phi^k_i(\beta) \ge \eta^k(\beta) \qquad \text{for } \beta \in [0,1] \quad \text{and} \quad i = 1, \dots, n \tag{7.100}$$

Note that the first two terms of Equation (7.93a) are bounded below by the first two terms of Equation (7.99) for all $\beta \in [0,1]$. Hence we only have to evaluate the last quadratic term. By (7.91b) and (7.91c), we know $\mathbf{r}^k(\mu^k)$ is the orthogonal sum of vectors $\hat{\mathbf{D}}_k^{-1}\mathbf{d}^k_x$ and $\hat{\mathbf{D}}_k\mathbf{d}^k_s$. Therefore,

$$(\hat{\mathbf{D}}_k^{-1}\mathbf{d}^k_x)_j(\hat{\mathbf{D}}_k\mathbf{d}^k_s)_j = (r^k_j(\mu^k) - (\hat{\mathbf{D}}_k\mathbf{d}^k_s)_j)(\hat{\mathbf{D}}_k\mathbf{d}^k_s)_j \le |r^k_j|^2/4$$

Moreover, we see that

$$\begin{aligned}(\hat{\mathbf{D}}_k^{-1}\mathbf{d}^k_x)_i(\hat{\mathbf{D}}_k\mathbf{d}^k_s)_i &= (\hat{\mathbf{D}}_k^{-1}\mathbf{d}^k_x)^T(\hat{\mathbf{D}}_k\mathbf{d}^k_s) - \sum_{j\ne i}(\hat{\mathbf{D}}_k^{-1}\mathbf{d}^k_x)_j(\hat{\mathbf{D}}_k\mathbf{d}^k_s)_j \\ &\ge 0 - \|\mathbf{r}^k(\mu^k)\|^2/4\end{aligned}$$

Hence we conclude that (7.100) holds.

Now consider the case of $\beta^k < 1$. We let $\gamma = \max\{\beta \mid \eta^k(\beta) \ge \psi^k(\beta)\}$, where $\psi^k(\beta)$ is defined by (7.94) with $\tau = \sigma$. Since $\mu^k = \sigma\phi^k_{\text{ave}}$, with (7.96), we know $\beta^k = \alpha^k \ge \gamma$. Moreover, putting (7.94) and (7.99) together, by computing a positive solution of the equation

$$\eta^k(\beta) = \psi^k(\beta) \tag{7.101}$$

we get

$$\beta^k = \alpha^k \ge \gamma = \frac{4(\sigma-\tau)\phi^k_{\text{ave}}}{\|\mathbf{r}^k(\mu^k)\|^2} \tag{7.102}$$

On the other hand, by (7.91), we have

$$
\begin{aligned}
\left\|\mathbf{r}^k(\mu^k)\right\|^2 &= \sum_{i=1}^{n}[(\phi_i^k)^{1/2} - \sigma\phi_{\text{ave}}^k(\phi_i^k)^{-1/2}]^2 \\
&= n\phi_{\text{ave}}^k - 2\sigma n\phi_{\text{ave}}^k + \sigma^2(\phi_{\text{ave}}^k)^2\sum_{i=1}^{n}(\phi_i^k)^{-1} \\
&\le n\phi_{\text{ave}}^k(1 - 2\sigma + \sigma^2\phi_{\text{ave}}^k/\phi_{\min}^k) \\
&= n\phi_{\text{ave}}^k(1 - 2\sigma + \sigma^2\theta^k) \\
&\le n\phi_{\text{ave}}^k(1+\sigma^2)\theta^k, \qquad \text{since } \theta^k \ge 1
\end{aligned} \tag{7.103}
$$

Combining the last two inequalities in (7.103) with (7.102), we obtain the inequality (7.97a).

Notice that $x_i^k d_{s_i}^k + s_i^k d_{x_i}^k = \mu^k - \phi_i^k$ and $(\mathbf{d}_x^k)^T(\mathbf{d}_s^k) = 0$. We further have

$$
\begin{aligned}
\phi_{\text{ave}}^{k+1} &= (1/n)\sum_{i=1}^{n}\phi_i^{k+1} = (1/n)(\mathbf{x}^{k+1})^T(\mathbf{s}^{k+1}) \\
&= \phi_{\text{ave}}^k + \beta^k(\mu^k - \phi_{\text{ave}}^k)
\end{aligned} \tag{7.104}
$$

Since $\mu^k = \sigma\phi_{\text{ave}}^k = \sigma(\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k)/n$, Equation (7.97b) follows immediately. Recalling the definitions (7.94), (7.95), and (7.96), we see that

$$
\phi_{\min}^{k+1} = \psi^k(\beta^k) = \phi_{\min}^k + \beta^k(\tau\phi_{\text{ave}}^k - \phi_{\min}^k) \tag{7.105}
$$

Together with (7.104), we have

$$
\begin{aligned}
\theta^{k+1} - \frac{\sigma}{\tau} &= \frac{\phi_{\text{ave}}^k + \beta^k\left(\sigma\phi_{\text{ave}}^k - \phi_{\text{ave}}^k\right)}{\phi_{\min}^k + \beta^k\left(\tau\phi_{\text{ave}}^k - \phi_{\min}^k\right)} - \frac{\sigma}{\tau} \\
&= \frac{\left(1-\beta^k\right)\left(\tau\phi_{\text{ave}}^k - \sigma\phi_{\min}^k\right)}{[\phi_{\min}^k + \beta^k\left(\tau\phi_{\text{ave}}^k - \phi_{\min}^k\right)]\tau} \\
&= \left[1 - \frac{\tau\theta^k\beta^k}{1 - \left(1 - \tau\theta^k\right)\beta^k}\right]\left(\theta^k - \frac{\sigma}{\tau}\right)
\end{aligned} \tag{7.106}
$$

When $\theta^k \le \sigma/\tau$, the right-hand side of (7.106) becomes nonpositive. Consequently, so does the left-hand side, and $\theta^{k+1} \le \sigma/\tau$. This proves (7.97d).

On the other hand, when $\theta^k > \frac{\sigma}{\tau}$, (7.106) implies that

$$
\theta^{k+1} - \frac{\sigma}{\tau} \le \left[1 - \frac{\tau\theta^k\beta^k}{1 + \tau\theta^k\beta^k}\right]\left(\theta^k - \frac{\sigma}{\tau}\right) \tag{7.107}
$$

Note that (7.97a) implies

$$
\theta^k\beta^k \ge \frac{4(\sigma - \tau)}{n(1+\sigma^2)} \tag{7.108}
$$

Substituting this inequality into (7.107), we have the result (7.97c).

Finally, let us prove (7.98a) and (7.98b) in case $\beta^k = 1$. It is not difficult to see that (7.98a) can be derived in the same way as we derive (7.97b). As to (7.98b), we first note that as $\beta^k = 1$,

$$\phi_i^k(\beta^k) \geq \psi^k(\beta^k) \qquad \text{for } i = 1, 2, \ldots, n \tag{7.109}$$

Hence,

$$\theta^{k+1} = \frac{\phi_{\text{ave}}^{k+1}}{\phi_{\text{min}}^{k+1}} \leq \frac{\phi_{\text{ave}}^k\left(\beta^k\right)}{\psi^k\left(\beta^k\right)} = \frac{\sigma}{\tau} \tag{7.110}$$

and the proof is complete.

In view of Theorem 7.3, if k^* is the earliest iteration at which $\theta^{k^*} \leq \sigma/\tau$, then

$$\frac{\sigma}{\tau} < \theta^k \leq (1-\nu)^k\left(\theta^0 - \frac{\sigma}{\tau}\right) + \frac{\sigma}{\tau}, \qquad \forall\, k < k^*$$

and

$$\theta^k \leq \frac{\sigma}{\tau}, \qquad \forall\, k \geq k^*$$

If such a k^* does not exist, then $\theta^k > \sigma/\tau$, $\forall\, k$ and

$$\frac{\sigma}{\tau} < \theta^k \leq (1-\nu)^k\left(\theta^0 - \frac{\sigma}{\tau}\right) + \frac{\sigma}{\tau}, \qquad \forall\, k$$

Notice that $0 < \nu < 1$, in either case we have

$$\theta^k \leq \max\left\{\sigma/\tau, \theta^0\right\}, \qquad \forall\, k$$

Then it is clear to see that θ^k gets smaller than $(\sigma/\tau) + 1$ in at most $O(n \log_e \theta^0)$ (say, $\hat{k}$) iterations. Consequently, it follows from Equation (7.97a) that

$$(1-\sigma)\beta^k \geq \frac{4(1-\sigma)(\sigma-\tau)}{n(1+\sigma^2)(\frac{\sigma}{\tau}+1)}, \qquad \forall\, k \geq \hat{k} \tag{7.111}$$

By the inequality (7.97b), the duality gap $\mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{w}^k$ attains the given accuracy ϵ and the iteration stops in at most

$$O\left(n \log_e\left(\frac{\mathbf{c}^T\mathbf{x}^0 - \mathbf{b}^T\mathbf{w}^0}{\epsilon}\right)\right)$$

additional iterations. Hence the primal-dual algorithm terminates in at most

$$O(n\ \log_e \theta^0) + O\left(n\ \log_e\left(\frac{\mathbf{c}^T\mathbf{x}^0 - \mathbf{b}^T\mathbf{w}^0}{\epsilon}\right)\right)$$

iterations.

There are various ways of setting the control parameters σ and τ such that $0 \leq \tau < \sigma < 1$. As a special case, we let $\sigma = 1/2$ and $\tau = 1/4$, then

$$\beta^k \geq \frac{4}{n\theta^k}$$

and (7.80) follows. Also notice that at each iteration of the primal-dual algorithm, the computational bottleneck is the inversion of the matrix $\mathbf{A}\hat{\mathbf{D}}_k^2\mathbf{A}^T$. A direct implementation requires $O(n^3)$ elementary operations for matrix inversion and results in an $O(n^4L)$ complexity for the primal-dual algorithm. Definitely, this complexity can be reduced by better implementation techniques.

7.3.5 Starting the Primal-Dual Algorithm

In order to apply the primal-dual algorithm, we start with an arbitrary point $(\mathbf{x}^0; \mathbf{w}^0; \mathbf{s}^0) \in R^{n+m+n}$ such that $\mathbf{x}^0 > \mathbf{0}$ and $\mathbf{s}^0 > \mathbf{0}$.

In case $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$ and $\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 = \mathbf{c}$, we know $\mathbf{x}^0 \in \mathbf{S}$ and $(\mathbf{w}^0; \mathbf{s}^0) \in \mathbf{T}$ and we have a starting solution for the primal-dual algorithm. Otherwise, consider the following pair of artificial primal and dual linear programs:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^T\mathbf{x} + \pi x_{n+1} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + (\mathbf{b} - \mathbf{A}\mathbf{x}^0)x_{n+1} = \mathbf{b} \\ & (\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 - \mathbf{c})^T\mathbf{x} + x_{n+2} = \lambda \\ & (\mathbf{x}; x_{n+1}; x_{n+2}) \geq \mathbf{0} \end{aligned} \tag{AP}$$

where x_{n+1} and x_{n+2} are two artificial variables and π and λ are sufficiently large positive numbers to be specified later;

$$\begin{aligned} \text{Maximize} \quad & \mathbf{b}^T\mathbf{w} + \lambda w_{m+1} \\ \text{subject to} \quad & \mathbf{A}^T\mathbf{w} + (\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 - \mathbf{c})w_{m+1} + \mathbf{s} = \mathbf{c} \\ & (\mathbf{b} - \mathbf{A}\mathbf{x}^0)^T\mathbf{w} + s_{n+1} = \pi \\ & w_{m+1} + s_{n+2} = 0 \\ & (\mathbf{s}; s_{n+1}; s_{n+2}) \geq \mathbf{0} \end{aligned} \tag{AD}$$

where w_{m+1}, s_{n+1} and s_{n+2} are artificial variables.

Notice that if we choose π and λ such that

$$\pi > (\mathbf{b} - \mathbf{A}\mathbf{x}^0)^T\mathbf{w}^0 \tag{7.112a}$$

$$\lambda > \left(\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 - \mathbf{c}\right)^T \mathbf{x}^0 \tag{7.112b}$$

then $(\mathbf{x}^0, x_{n+1}^0, x_{n+2}^0)$ and $(\mathbf{w}^0, w_{m+1}^0; \mathbf{s}^0, s_{n+1}^0, s_{n+2}^0)$ are feasible solutions to the artificial problems (AP) and (AD), respectively, where

$$\begin{aligned} x_{n+1}^0 &= 1 \\ x_{n+2}^0 &= \lambda - (\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 - \mathbf{c})^T\mathbf{x}^0 \\ w_{m+1}^0 &= -1 \end{aligned}$$

$$s_{n+1}^0 = \pi - (\mathbf{b} - \mathbf{A}\mathbf{x}^0)^T\mathbf{w}^0$$

$$s_{n+2}^0 = 1$$

In this case, the primal-dual algorithm can be applied to the artificial problems (AP) and (AD) with a known starting solution. Actually, the optimal solutions of (AP) and (DP) are closely related to those of the original problems (P) and (D). The following theorem describes this relationship:

Theorem 7.4. Let $\mathbf{x}^*$ and $(\mathbf{w}^*; \mathbf{s}^*)$ be optimal solutions of the original problems (P) and (D). In addition to (7.112a) and (7.112b), suppose that

$$\lambda > (\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 - \mathbf{c})^T\mathbf{x}^* \tag{7.112c}$$

and

$$\pi > (\mathbf{b} - \mathbf{A}\mathbf{x}^0)^T\mathbf{w}^* \tag{7.112d}$$

Then the following two statements are true:

(i) A feasible solution $(\overline{\mathbf{x}}, \overline{x}_{n+1}, \overline{x}_{n+2})$ of (AP) is a minimizer if and only if $\overline{\mathbf{x}}$ solves (P) and $\overline{x}_{n+1} = 0$.

(ii) A feasible solution $(\overline{\mathbf{w}}, \overline{w}_{m+1}; \overline{\mathbf{s}}, \overline{s}_{n+1}, \overline{s}_{n+2})$ of (AD) is a maximizer if and only if $(\overline{\mathbf{w}}; \overline{\mathbf{s}})$ solves (D) and $\overline{w}_{m+1} = 0$.

Proof. Since $\mathbf{x}^*$ is feasible to (P), if we further define that $x_{n+1}^* = 0$ and $x_{n+2}^* = \lambda - (\mathbf{A}^T\mathbf{w}^0 + \mathbf{s}^0 - \mathbf{c})^T\mathbf{x}^*$, then $(\mathbf{x}^*, x_{n+1}^*, x_{n+2}^*)$ is feasible to (AP). Suppose that $(\mathbf{x}, x_{n+1}, x_{n+2})$ is feasible to (AP) with $x_{n+1} > 0$, then

$$\mathbf{c}^T\mathbf{x}^* + \pi x_{n+1}^* = \mathbf{w}^{*T}\mathbf{b} = \mathbf{w}^{*T}(\mathbf{A}\mathbf{x} + (\mathbf{b} - \mathbf{A}\mathbf{x}^0)x_{n+1})$$

Note that $\mathbf{A}^T\mathbf{w}^* + \mathbf{s}^* = \mathbf{c}$, $x_{n+1} > 0$, and (7.112d). We see that

$$\mathbf{c}^T\mathbf{x}^* + \pi x_{n+1}^* < (\mathbf{c} - \mathbf{s}^*)^T\mathbf{x} + \pi x_{n+1} \le \mathbf{c}^T\mathbf{x} + \pi x_{n+1}$$

since $\mathbf{s}^{*T}\mathbf{x} \ge 0$. This means that $(\mathbf{x}, x_{n+1}, x_{n+2})$ cannot be an optimal solution to (AP) unless $x_{n+1} = 0$. Furthermore, through the property of continuity, we know $(\mathbf{x}^*, x_{n+1}^*, x_{n+2}^*)$ is an optimal solution to (AP). Therefore, if a feasible solution $(\overline{\mathbf{x}}, \overline{x}_{n+1}, \overline{x}_{n+2})$ of (AP) is optimal, then $\overline{x}_{n+1} = 0$ and $\mathbf{c}^T\overline{\mathbf{x}} = \mathbf{c}^T\mathbf{x}^*$. Because $\overline{\mathbf{x}}$ satisfies all the constraints of (P), it must be an optimal solution to (P).

Conversely, if $(\overline{\mathbf{x}}, 0, \overline{x}_{n+2})$ is a feasible solution of (AP) and $\overline{\mathbf{x}}$ is an optimal solution of (P), then the objective value $\mathbf{c}^T\overline{\mathbf{x}} + \pi\overline{x}_{n+1}$ coincides with the minimal value $\mathbf{c}^T\mathbf{x}^* + \pi x_{n+1}^*$. Hence it is an optimal solution of (AP). This concludes the proof of part (i). Similarly, we can prove part (ii).

7.3.6 Practical Implementation

In the real implementation of the primal-dual algorithm, it is a very difficult task to keep $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k) \in \mathbf{S} \times \mathbf{T}$ due to numerical problems. Also the choice of the control parameters greatly affects the performance of the algorithm. Much effort has been devoted to designing a version of the primal-dual algorithm for practical implementation. In this section, we introduce one version of the primal-dual algorithm that allows us to start with an arbitrary point $(\mathbf{x}^0; \mathbf{w}^0; \mathbf{s}^0)$ with $\mathbf{x}^0, \mathbf{s}^0 > \mathbf{0}$. This version produces a sequence of iterates $\{(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k)\}$, with $\mathbf{x}^k, \mathbf{s}^k > \mathbf{0}$, which leads to an optimal solution, although they no longer stay on the curve of $\mathbf{S} \times \mathbf{T}$. It is important to know that, at this moment, there is no rigorous convergence proof for this version of the primal-dual algorithm, but it is widely used in many commercial packages.

Moving direction. The basic idea of this version follows from the analysis of Section 3.2. Assume that we are at a point $(\mathbf{x}^k; \mathbf{w}^k; \mathbf{s}^k)$ for some $\mu^k > 0$, such that $\mathbf{x}^k, \mathbf{s}^k > \mathbf{0}$. The Newton direction $(\mathbf{d}_x^k; \mathbf{d}_w^k; \mathbf{d}_s^k)$ is determined by Equations (7.89a)–(7.89c). Combining (7.89a), (7.89b), and (7.89c), we have

$$\mathbf{d}_x^k = \mu^k \hat{\mathbf{D}}_k \mathbf{P}_k \hat{\mathbf{D}}_k \mathbf{X}_k^{-1}\mathbf{e} - \hat{\mathbf{D}}_k \mathbf{P}_k \hat{\mathbf{D}}_k \mathbf{c} + \hat{\mathbf{D}}_k^2 \mathbf{A}^T (\mathbf{A}\hat{\mathbf{D}}_k^2 \mathbf{A}^T)^{-1} \mathbf{t}^k \tag{7.113a}$$

where $\hat{\mathbf{D}}_k^2 = \mathbf{X}_k \mathbf{S}_k^{-1}$ and $\mathbf{P}_k = \mathbf{I} - \hat{\mathbf{D}}_k \mathbf{A}^T (\mathbf{A}\hat{\mathbf{D}}_k^2 \mathbf{A}^T)^{-1} \mathbf{A}\hat{\mathbf{D}}_k$, which is the projection matrix onto the null space of matrix $\mathbf{A}\hat{\mathbf{D}}_k$.

If we further define that

$$\mathbf{d}_{x_{ctr}}^k = \mu^k \hat{\mathbf{D}}_k \mathbf{P}_k \hat{\mathbf{D}}_k \mathbf{X}_k^{-1}\mathbf{e}, \qquad \mathbf{d}_{x_{obj}}^k = -\hat{\mathbf{D}}_k \mathbf{P}_k \hat{\mathbf{D}}_k \mathbf{c} \quad \text{and} \quad \mathbf{d}_{x_{feas}}^k = \hat{\mathbf{D}}_k^2 \mathbf{A}^T (\mathbf{A}\hat{\mathbf{D}}_k^2 \mathbf{A}^T)^{-1} \mathbf{t}^k$$

then (7.113a) becomes

$$\mathbf{d}_x^k = \mathbf{d}_{x_{ctr}}^k + \mathbf{d}_{x_{obj}}^k + \mathbf{d}_{x_{feas}}^k \tag{7.113b}$$

The first term of (7.113b) is usually called the *centering direction*, since in light of the potential push, it is nothing but the projection of the push vector $(1/\mathbf{x}^k)$ which helps the algorithm stay away from the walls of the primal polytope. The second term is called the *objective reduction direction*, since it is the projected negative gradient of the primal objective function which leads to a reduction in the primal objective function. The third term is called the *feasibility direction*, since $\mathbf{t}^k$ is a measure of primal feasibility. Also note that $\mathbf{A}\mathbf{d}_{x_{ctr}}^k = \mathbf{0}$, and $\mathbf{A}\mathbf{d}_{x_{obj}}^k = \mathbf{0}$. Hence these two directions are in the null space of matrix $\mathbf{A}$, and the primal feasibility is solely affected by $\mathbf{d}_{x_{feas}}^k$.

In practice, if we start with an arbitrary point $(\mathbf{x}^0; \mathbf{w}^0; \mathbf{s}^0)$ with $\mathbf{x}^0, \mathbf{s}^0 > \mathbf{0}$, the value of $\mathbf{t}^0$ might be very large, since $\mathbf{x}^0$ could be far from being feasible. At this point, the main effort of the algorithm will be in finding a feasible point near the central trajectory. Once a feasible solution is found (say, at the kth iteration) the algorithm will try to keep $\mathbf{t}^{k'} = \mathbf{0}$ for all $k' \geq k$, except for the case that feasibility is lost due to numerical truncation or round-off errors. In this way, $\mathbf{d}_{x_{feas}}^k$ will eventually vanish from the picture.

In a similar fashion one can carry out the analysis of moving directions on the dual side, i.e, $\mathbf{d}_w^k$ and $\mathbf{d}_s^k$. It is left as an exercise for the reader.

Step-length. Once the moving direction is obtained, we are ready to move to a new point $(\mathbf{x}^{k+1}; \mathbf{w}^{k+1}; \mathbf{s}^{k+1})$ with $\mathbf{x}^{k+1} > \mathbf{0}$ and $\mathbf{s}^{k+1} > \mathbf{0}$. To do so, we let

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \beta_P \mathbf{d}_x^k \tag{7.114a}$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k + \beta_D \mathbf{d}_w^k \tag{7.114b}$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k + \beta_D \mathbf{d}_s^k \tag{7.114c}$$

where β_P and β_D are the step-lengths in the primal and dual spaces, respectively. The nonnegativity requirements of $\mathbf{x}^{k+1}$ and $\mathbf{s}^{k+1}$ dictate the choice of the step-lengths β_P and β_D. One simple way, as we did before, is to take

$$\beta_P = \frac{1}{\max\left\{1, -d_{x_i}^k/\alpha x_i^k\right\}} \tag{7.115a}$$

and

$$\beta_D = \frac{1}{\max\left\{1, -d_{s_i}^k/\alpha s_i^k\right\}} \tag{7.115b}$$

where $\alpha < 1$, $(d_x^k)_i$ is the ith component of $\mathbf{d}_x^k$, x_i^k is the ith component of $\mathbf{x}^k$, $(d_s^k)_i$ is the ith component of $\mathbf{d}_s^k$, and s_i^k is the ith component of $\mathbf{s}^k$.

Adjusting Penalty Parameters and Stopping Rules. Notice that the moving direction at the kth iteration is determined by the value of the penalty parameter μ^k. Strictly speaking, the translation described above has to be carried out several times for a fixed value of μ^k, so that the Newton steps actually converge to the central trajectory corresponding to that μ^k. However, it is apparent that doing so would be an "overkill." Recall that at optimality μ^k has to be brought to zero to satisfy the complementary slackness. Therefore, in practical implementations, the value of μ^k is reduced from iteration to iteration and only one Newton step is carried out for a given value of μ^k.

The way in which μ^k can be reduced at each iteration is suggested by the algorithm itself. From Equation (7.76c) we see that $\mu = \mathbf{s}^T\mathbf{x}/n$. Plugging in the values of $\mathbf{x}^k$ and $\mathbf{s}^k$ gives us a reasonably good measure of the penalty parameter for the current point. According to our experience, sometimes, a lower value of μ^k, say, $\sigma[(\mathbf{s}^k)^T\mathbf{x}^k]/n$ with $\sigma < 1$, could accelerate the convergence of the algorithm. There have been other similar ideas reported by various authors on the choice of μ^k. Nevertheless, the above simple rule seems to work well for a variety of practical problems solved by the authors.

As far as the stopping rules are concerned, we may check the primal feasibility, dual feasibility, and complementary slackness. Notice that the primal feasibility is measured by $\mathbf{t}^k$, dual feasibility by $\mathbf{u}^k$, and complementary slackness by $\mathbf{v}^k$ as defined by (7.85).

Step-by-Step Procedure. As a summary of our discussion, we now provide a step-by-step procedure for the implementation of the new version of the primal-dual interior-point algorithm.

Step 1 (starting the algorithm): Set $k = 0$. Choose an arbitrary $(\mathbf{x}^0; \mathbf{w}^0; \mathbf{s}^0)$ with $\mathbf{x}^0 > \mathbf{0}$ and $\mathbf{s}^0 > \mathbf{0}$, and choose sufficiently small positive numbers ϵ_1, ϵ_2, and ϵ_3.

Step 2 (intermediate computations): Compute

$$\mu^k = \frac{\left(\mathbf{x}^k\right)^T \mathbf{s}^k}{n}$$

$\mathbf{t}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$, $\mathbf{u}^k = \mathbf{c} - \mathbf{A}^T\mathbf{w}^k - \mathbf{s}^k$, $\mathbf{v}^k = \mu^k\mathbf{e} - \mathbf{X}_k\mathbf{S}_k\mathbf{e}$, $\mathbf{p}^k = \mathbf{X}_k^{-1}\mathbf{v}^k$, and $\hat{\mathbf{D}}_k^2 = \mathbf{X}_k\mathbf{S}_k^{-1}$, where $\mathbf{X}_k$ and $\mathbf{S}_k$ are diagonal matrices whose diagonal entries are x_i^k and s_i^k, respectively.

Step 3 (checking for optimality): If

$$\mu^k < \epsilon_1, \qquad \frac{||\mathbf{t}||}{||\mathbf{b}||+1} < \epsilon_2, \qquad \text{and} \qquad \frac{||\mathbf{u}||}{||\mathbf{c}||+1} < \epsilon_3$$

then STOP. The solution is optimal. Otherwise go to the next step.

[*Note:* $||\mathbf{u}||$ and $||\mathbf{c}||$ are computed only when the dual constraints are violated. If $\mathbf{u} \geq \mathbf{0}$, then there is no need to compute this measure of optimality.]

Step 4 (calculating directions of translation): Compute

$$\mathbf{d}_w^k = \left(\mathbf{A}\hat{\mathbf{D}}_k^2\mathbf{A}^T\right)^{-1}\left(\mathbf{A}\hat{\mathbf{D}}_k^2\left(\mathbf{u}^k - \mathbf{p}^k\right) + \mathbf{t}^k\right)$$

$$\mathbf{d}_s^k = \mathbf{u}^k - \mathbf{A}^T\mathbf{d}_w^k$$

$$\mathbf{d}_x^k = \hat{\mathbf{D}}_k^2\left(\mathbf{p}^k - \mathbf{d}_s^k\right)$$

Step 5 (checking for unboundedness): If

$$\mathbf{t}^k = \mathbf{0}, \quad \mathbf{d}_x^k > \mathbf{0}, \quad \text{and} \quad \mathbf{c}^T\mathbf{d}_x^k < 0$$

then the primal problem (P) is unbounded. If

$$\mathbf{u}^k = \mathbf{0}, \quad \mathbf{d}_s^k > \mathbf{0}, \quad \text{and} \quad \mathbf{b}^T\mathbf{d}_w^k > 0$$

then the dual problem (D) is unbounded. If either of these cases happens, STOP. Otherwise go to the next step.

Step 6 (finding step-lengths): Compute the primal and dual step-lengths

$$\beta_P = \frac{1}{\max\left\{1, -d_{x_i}^k/\alpha x_i^k\right\}}$$

and

$$\beta_D = \frac{1}{\max\left\{1, -d_{s_i}^k/\alpha s_i^k\right\}}$$

where $\alpha < 1$ (say, 0.99).

Step 7 (moving to a new point): Update the solution vectors

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \beta_P\mathbf{d}_x^k$$

$$\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \beta_D\mathbf{d}_w^k$$

$$\mathbf{s}^{k+1} \leftarrow \mathbf{s}^k + \beta_D\mathbf{d}_s^k$$

Set $k \leftarrow k+1$ and go to Step 2.

Now we present a numerical example to illustrate the algorithm.

Example 7.3

Consider the same problem as in Example 7.1 and Example 7.2. We begin with an arbitrary assignment of $\mathbf{x}^0 = [1 \quad 1 \quad 1 \quad 1]^T$, $\mathbf{w}^0 = [0 \quad 0]^T$, $\mathbf{s}^0 = [1 \quad 1 \quad 1 \quad 1]^T$. With this information, we see that $\mathbf{X}_0$, $\mathbf{S}_0$ and $\hat{\mathbf{D}}_0^2$ are all equal to the identity matrix $\mathbf{I}$, and $\mu^0 = 1$.

We now compute

$$\mathbf{t}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0 = [14 \quad 13]^T, \qquad \mathbf{u}^0 = \mathbf{c} - \mathbf{A}^T\mathbf{w}^0 - \mathbf{s}^0 = [-3 \quad 0 \quad -1 \quad -1]^T$$

$$\mathbf{v}^0 = \mu^0\mathbf{e} - \mathbf{X}_0\mathbf{S}_0\mathbf{e} = [0 \quad 0 \quad 0 \quad 0]^T, \qquad \mathbf{p}^0 = \mathbf{X}_0^{-1}\mathbf{v}^0 = [0 \quad 0 \quad 0 \quad 0]^T$$

Therefore,

$$\mathbf{d}_w^0 = \left(\mathbf{A}\hat{\mathbf{D}}_0^2\mathbf{A}^T\right)^{-1}\left[\mathbf{A}\hat{\mathbf{D}}_0^2\left(\mathbf{u}^0 - \mathbf{p}^0\right) + \mathbf{t}^0\right] = \begin{bmatrix} 0.4 & 0.2 \\ 0.2 & 0.6 \end{bmatrix}\begin{bmatrix} 10 \\ 12 \end{bmatrix} = \begin{bmatrix} 6.4 \\ 0.2 \end{bmatrix}$$

$$\mathbf{d}_s^0 = \mathbf{u}^0 - \mathbf{A}^T\mathbf{d}_w^0 = [-9.4 \quad -2.8 \quad -7.4 \quad -10.2]^T$$

$$\mathbf{d}_x^0 = \hat{\mathbf{D}}_0^2(\mathbf{p}^0 - \mathbf{d}_s^0) = [9.4 \quad 2.8 \quad 7.4 \quad 10.2]^T$$

Although $\mathbf{d}_x^0 > \mathbf{0}$ and $\mathbf{c}^T\mathbf{d}_x^0 < 0$, we see from $\mathbf{t}^0$ that the primal is still infeasible at this moment. Hence we proceed further.

We choose $\alpha = 0.99$. Using the formula to compute the step-lengths, we find that $\beta_P = 1.0$ and $\beta_D = 1/10.30303 = 0.097059$. Therefore the updated solution becomes

$$\mathbf{x}^1 = [1 \quad 1 \quad 1 \quad 1]^T + 1.0 \times [9.4 \quad 2.8 \quad 7.4 \quad 10.2]^T$$
$$= [10.4 \quad 3.8 \quad 8.4 \quad 11.2]^T$$

$$\mathbf{s}^1 = [1 \quad 1 \quad 1 \quad 1]^T + 0.0.097059 \times [-9.4 \quad -2.8 \quad -7.4 \quad -10.2]^T$$
$$= [0.08765 \quad 0.72824 \quad 0.28176 \quad 0.00999]^T$$

and

$$\mathbf{w}^1 = [0 \quad 0] + 0.0.097059 \times [6.4 \quad 9.2]^T = [0.62118 \quad 0.89294]^T$$

The new solution $\mathbf{x}^1$ is already primal feasible, which is in tune with our previous discussions. The reader is urged to carry out more iterations to see that an optimal solution with

$$\mathbf{x}^* = [30 \quad 15 \quad 0 \quad 0]^T, \qquad \mathbf{w}^* = [-2 \quad -1]^T, \quad \text{and} \quad \mathbf{s}^* = [0 \quad 0 \quad 2 \quad 1]^T$$

is finally reached.

7.3.7 Accelerating via Power Series Method

As we discussed before, ideally it takes several Newton steps for a given penalty parameter to get onto the central trajectory, although we found that in most cases it is adequate to carry out only one Newton step for each penalty parameter. In order to track the continuous central trajectories more closely, we may consider using the power-series approximation method as we did for the dual affine scaling algorithm.

To simplify the discussion, we choose the smaller one between β_P and β_D as a common step-length β for both the primal and dual iterations and focus on a current point, say $(\mathbf{x}^0; \mathbf{w}^0; \mathbf{s}^0)$. In the limiting case of $\beta \to 0$, (7.84) can be rewritten in the following continuous version:

$$\mathbf{A}\frac{d\mathbf{x}(\beta)}{d\beta} = \mathbf{t}(\beta) \tag{7.116a}$$

$$\mathbf{A}^T\frac{d\mathbf{w}(\beta)}{d\beta} + \frac{d\mathbf{s}(\beta)}{d\beta} = \mathbf{u}(\beta) \tag{7.116b}$$

$$\mathbf{S}(\beta)\frac{d\mathbf{x}(\beta)}{d\beta} + \mathbf{X}(\beta)\frac{d\mathbf{s}(\beta)}{d\beta} = \mathbf{v}(\beta) \tag{7.116c}$$

such that $\mathbf{x}(0) = \mathbf{x}^0$, $\mathbf{w}(0) = \mathbf{w}^0$, and $\mathbf{s}(0) = \mathbf{s}^0$, where $\mathbf{t}(\beta) = \mathbf{b} - \mathbf{A}\mathbf{x}(\beta)$, $\mathbf{u}(\beta) = \mathbf{c} - \mathbf{A}^T\mathbf{w}(\beta) - \mathbf{s}(\beta)$, $\mathbf{v}(\beta) = \mu\mathbf{e} - \mathbf{X}(\beta)\mathbf{S}(\beta)\mathbf{e}$, and $\mathbf{X}(\beta)$ and $\mathbf{S}(\beta)$ are the diagonal matrices whose diagonal elements are $x_j(\beta)$ and $s_j(\beta)$, respectively.

Now, what we have to do is to find a solution to the system depicted by Equation (7.116) in the form of a truncated power series. This can be carried out exactly as we did for the dual affine scaling algorithm. The only difference is that, in addition to the expansions of $\mathbf{w}(\beta)$ and $\mathbf{s}(\beta)$, we need to consider the expansions of $\mathbf{x}(\beta)$, $\mathbf{t}(\beta)$, $\mathbf{u}(\beta)$, and $\mathbf{v}(\beta)$ around $\beta = 0$ as well. Owing to the similarity in procedure, the algebraic simplifications are left for the readers as an exercise.

Based on our experience, we note the following characteristics of the primal-dual interior point algorithm:

1. The algorithm is essentially a one-phase method.
2. The computational burden per iteration is more or less the same as that of the primal or the dual affine scaling algorithm.
3. The improvement in convergence rate obtained by performing the power series enhancement to the primal-dual algorithm is not as significant as we obtained in the dual affine scaling algorithm.
4. Owing to its "self-correcting nature" (at least, in the case of restoring feasibility that might have been lost due to numerical errors of computers), the primal-dual algorithm is found to be numerically robust.

7.4 CONCLUDING REMARKS

In this chapter we have studied the basic concepts of affine scaling including the primal, dual, and primal-dual algorithms. Many extensions have been made to enhance the basic affine scaling algorithms. However, it is important to understand that the research work in this area is still ongoing. Different barrier functions including the entropy and inverse functions have been proposed. Unfortunately, no polynomial convergence result has been achieved at this moment. A unified treatment will definitely help the development of the interior-point methods for linear programming. The idea of using interior-point methods to solve quadratic and convex programming problems with linear constraints has also been explored by many researchers. We shall study these interesting topics in later chapters.

REFERENCES FOR FURTHER READING

7.1 Adler, I., Karmarkar, N., Resende, M. G. C., and Veiga, G., "An implementation of Karmarkar's algorithm for linear programming," *Mathematical Programming* 44, 297–335 (1989).

7.2 Adler, I., and Resende, M. G. C., "Limiting behavior of the affine scaling continuous trajectories for linear programming problems," *Mathematical Programming* 50, 29–51 (1991).

7.3 Barnes, E. R., "A variation of Karmarkar's algorithm for solving linear programming problems," *Mathematical Programming* 36, 174–182 (1986).

7.4 Cavalier, T. M., and Soyster, A. L., "Some computational experience and a modification of the Karmarkar algorithm," presented at the 12th Symposium on Mathematical Programming, Cambridge, MA (1985).

7.5 Dikin, I. I., "Iterative solution of problems of linear and quadratic programming" (in Russian), *Doklady Akademiia Nauk USSR* 174, 747–748, (English translation) *Soviet Mathematics Doklady* 8, 674–675 (1967).

7.6 Frisch, K. R., "The logarithmic potential method of convex programming," Technical Report, University Institute of Economics, Oslo, Norway (1955).

7.7 Freund, R. M., "Polynomial-time algorithms for linear programming based only on primal affine scaling and projected gradients of a potential function," *Mathematical Programming* 51, 203–222 (1991).

7.8 Gill, P. E., Murray, W., Saunders, M. A., Tomlin, J. A., and Wright, M. H., "On projected barrier methods for linear programming and an equivalence to Karmarkar's projective method," *Mathematical Programming* 36, 183–209 (1986).

7.9 Gonzaga, C., "An algorithm for solving linear programming problems in $O(n^3L)$ operations," in *Progress in Mathematical Programming: Interior-Point and Related Methods*, ed. N. Megiddo, Springer-Verlag, New York, 1–28 (1989).

7.10 Gonzaga, C., "Polynomial affine algorithms for linear programming," *Mathematical Programming* 49, 7–21 (1990).

7.11 Huard, P., "Resolution of mathematical programming with nonlinear constraints by the method of centers," in *Nonlinear Programming*, ed. J. Abadie, North-Holland, Amsterdam, Holland, 207–219 (1967).

7.12 Karmarkar, N., Lagarias, J. C., Slutsman, L., and Wang, P., "Power series variants of Karmarkar-type algorithms," *AT&T Technical Journal* 68, No. 3, 20–36 (1989).

7.13 Kojima, M., Mizuno, S., and Yoshise, A., "A primal-dual interior point method for linear programming," in *Progress in Mathematical Programming: Interior-Point and Related Methods*, ed. N. Megiddo, Springer-Verlag, New York, 29–48 (1989).

7.14 Megiddo, N., "On the complexity of linear programming," in *Advances in Economical Theory*, ed. T. Bewely, Cambridge University Press, Cambridge, 225–268 (1987).

7.15 Megiddo, N., *Progress in Mathematical Programming: Interior-Point and Related Methods*, Springer-Verlag, New York (1989).

7.16 Megiddo, N., and Shub, M., "Boundary behavior of interior point algorithms in linear programming," *Mathematics of Operations Research* 14, 97–146 (1989).

7.17 Monteiro, R. C., and Adler, I., "Interior path following primal-dual algorithms. Part I: Linear programming," *Mathematical Programming* 44, 27–42 (1989).

7.18 Monteiro, R. C., Adler, I., and Resende, M. C., "A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension," *Mathematics of Operations Research* 15, 191–214 (1990).

7.19 Renegar, J., "A polynomial-time algorithm based on Newton's method for linear programming," *Mathematical Programming* 40, 59–93 (1988).

7.20 Roos, C., "A new trajectory following polynomial-time algorithm for linear programming problem," *Journal of Optimization Theory and Applications* 63, 433–458 (1989).

7.21 Roos, C., and Vial, J.-Ph., "Long steps with the logarithmic penalty barrier function in linear programming," in *Economic Decision Making: Games, Economics, and Optimization*, ed. J. Gabszevwicz, J.-F. Richard, and L. Wolsey, Elsevier Science Publisher B.V., 433–441 (1990).

7.22 Sun, J., "A convergence proof for an affine-scaling algorithm for convex quadratic programming without nondegeneracy assumptions," manuscript to appear in *Mathematical Programming* (1993).

7.23 Tseng, P., and Luo, Z. Q., "On the convergence of affine-scaling algorithm," manuscript to appear in *Mathematical Programming* 53 (1993).

7.24 Tsuchiya, T., "A study on global and local convergence of interior point algorithms for linear programming" (in Japanese), PhD thesis, Faculty of Engineering, The University of Tokyo, Tokyo, Japan (1991).

7.25 Vanderbei, R. J., "Karmarkar's algorithm and problems with free variables," *Mathematical Programming* 43, 31–44 (1989).

7.26 Vanderbei, R. J., "ALPO: Another linear program solver," Technical Memorandum, No. 11212-900522-18TM, AT&T Bell Laboratories (1990).

7.27 Vanderbei, R. J., and Lagarias, J. C., "I. I. Dikin's convergence result for the affine-scaling algorithm," *Contemporary Mathematics* 114, 109–119 (1990).

7.28 Vanderbei, R. J., Meketon, M. S., and Freedman, B. A., "A modification of Karmarkar's linear programming algorithm," *Algorithmica* 1, 395–407 (1986).

7.29 Vaidya, P. M., "An algorithm for linear programming which requires $O(((m+n)n^2+(m+n)^{1.5}n)L)$ arithmetic operations," *Mathematical Programming* 47, 175–201 (1990).

7.30 Ye, Y., "An $O(n^3L)$ potential reduction algorithm for linear programming," *Contemporary Mathematics* 114, 91–107 (1990).

7.31 Zhang, Y., Tapia, R. A., and Dennis, J. E., "On the superlinear and quadratic convergence of primal-dual interior point linear programming algorithms," *SIAM Journal on Optimization* 2, 304–324 (1992).

EXERCISES

7.1. You are given two algorithms, A and B. Algorithm A solves systems of linear equations; Algorithm B solves linear programming problems.

(a) How can you use Algorithm A to solve a linear programming problem?

(b) How can you use Algorithm B to solve a system of linear equations?

(c) Combining (a) and (b), what is your conclusion? Why?

7.2. Consider the following linear programming problem:

$$\begin{aligned} \text{Minimize} \quad & -x_1 + 1 \\ \text{subject to} \quad & x_3 - x_4 = 0 \\ & x_1 + x_2 + x_3 + x_4 = 1 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

(a) Draw a graph of its feasible domain. Notice that $(0, 0, 0.5, 0.5)$ is a vertex. Use the revised simplex method to find its moving direction at this vertex and display it on the graph.

(b) Note that $(0.01, 0.01, 0.49, 0.49)$ is an interior feasible solution which is "near" to the vertex in (a). Use Karmarkar's algorithm to find its moving direction at this solution and display it on the graph.

(c) Use the primal affine scaling algorithm to find its moving direction at $(0.01, 0.01, 0.49, 0.49)$ and display it on the graph.

(d) Use the primal affine scaling algorithm with logarithmic barrier function to find its moving direction at $(0.01, 0.01, 0.49, 0.49)$ and display it on a graph.

(e) Compare the directions obtained from (a) – (d). What kind of observations can be made? Do you have any reason to support your observations?

7.3. Focus on the same linear programming problem as in Exercise 7.2.

(a) Find its dual problem and draw a graph of the dual feasible domain.

(b) Show that $(1, -2)$ is an interior feasible solution to the dual linear program.

(c) Apply the dual affine scaling algorithm to find its moving direction at this point and display it on the graph of the dual feasible domain.

(d) Is this moving direction pointing to the dual optimal solution?

(e) Apply the dual affine scaling algorithm with logarithmic barrier function to find its moving direction at this point and display it on the graph of the dual feasible domain.

(f) Is the direction obtained in (e) better than that in (c)? Why?

7.4. Focus on the same linear programming problem again.

(a) Starting with the primal feasible solution $\mathbf{x} = (0.01, 0.01, 0.49, 0.49)$ and dual feasible solution $\mathbf{w} = (1, -2)$, apply the primal-dual algorithm as stated in Section 7.3.6 under "Step-by-Step Procedure" to find its moving directions $\mathbf{d}_x$ and $\mathbf{d}_w$.

(b) Display the moving directions on the corresponding graphs.

(c) Can you make further observations and explain why?

7.5. Given a linear programming problem with bounded feasible domain, if the problem is both primal and dual nondegenerate and $\mathbf{x}^k$ is a primal feasible solution, show that

(a) $\mathbf{AX}_k$ is of full row rank (assuming that $m < n$).

(b) The set C defined in (7.15) is a set of vertices of the polytope P of primal feasible domain.

7.6. Consider a linear programming problem with lower bounds:

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{q} \end{aligned}$$

where $\mathbf{A}$ is an $m \times n$ matrix with full row rank and $\mathbf{q} \in R^n$.

(a) Convert it into a standard form linear programming problem with exactly n variables.

(b) Find the dual linear program of (a). Show that when $\mathbf{q} = \mathbf{0}$, a regular dual program is obtained.

(c) Our objective is to design an interior-point method to solve the problem. The basic philosophy is to map a current interior solution $\mathbf{x}^k$ ($\mathbf{A}\mathbf{x}^k = \mathbf{b}$ and $\mathbf{x}^k > \mathbf{q}$) to the "center" of the first orthant of R^n (i.e., $\mathbf{e} = (1, \ldots, 1)^T$).

(i) Find such a transformation and prove it is one-to-one and onto from the set $\{\mathbf{x} \in R^n \mid \mathbf{x} \geq \mathbf{q}\}$ to the set $\{\mathbf{y} \in R^n \mid \mathbf{y} \geq \mathbf{0}\}$.

(ii) Write down the corresponding linear program in the transformed space.

(iii) In the transformed space, project the negative gradient of the objective function into the null space of the constraints. What is the moving direction?

(iv) Derive the corresponding moving direction in the original space.

(v) Apply the primal affine scaling algorithm to the converted standard linear program of (a). Compare the moving direction with the one obtained in (iii). What is your conclusion?

(vi) Continue the work of (iii): how do you choose an appropriate step-length to keep feasibility?

(vii) Give the formula for updating a current interior solution.

(viii) What is your stopping rule?

(ix) How do you find an initial interior solution?

(x) Finally, state a step-by-step procedure to solve a linear programming problem with lower bounds.

7.7. Consider the primal affine scaling with logarithmic barrier function. Define $\mathbf{P}_{\mathbf{AX}_k}$ to be the projection map onto the null space of matrix $\mathbf{AX}_k$, and show that the moving direction (7.49a) at a current solution $\mathbf{x}^k$ can be written as

$$\mathbf{d}_\mu^k = -\mathbf{X}_k\mathbf{P}_{\mathbf{AX}_k}\left(\frac{\mathbf{X}_k\mathbf{c}}{\mu^k} - \mathbf{e}\right)$$

7.8. In this problem, we try to outline a proof showing the primal affine scaling algorithm with logarithmic barrier function is a polynomial-time algorithm. This proof is due to Roos and Vial.

(a) Show that

$$\mathbf{P}_{\mathbf{AX}_k}\left(\frac{\mathbf{X}_k\mathbf{c}}{\mu^k} - \mathbf{e}\right) = \frac{\mathbf{X}_k\mathbf{z}(\mathbf{x}^k, \mu^k)}{\mu^k} - \mathbf{e}, \qquad \text{where } \mathbf{z}\left(\mathbf{x}^k, \mu^k\right) \text{ minimizes } \left\|\frac{\mathbf{X}_k\mathbf{z}}{\mu^k} - \mathbf{e}\right\|$$

with the constraints $\mathbf{A}^T\mathbf{y}+\mathbf{z} = \mathbf{c}$ and $\mathbf{y} \in R^m$. [*Hint:* Consider the first-order optimality conditions of the minimization problem.]

(b) Problem (a) indicates that the 2-norm of

$$\mathbf{P}_{\mathbf{AX}_k}\left(\frac{\mathbf{X}_k\mathbf{c}}{\mu^k}-\mathbf{e}\right)$$

can be used as a measure for the distance of a given point $\mathbf{x}^k$ to the point $\mathbf{x}^k(\mu^k)$ on the central trajectory. Let us denote this distance measure by $\delta(\mathbf{x}^k, \mu^k)$, i.e.,

$$\delta(\mathbf{x}^k, \mu^k) = \left\|\mathbf{P}_{\mathbf{AX}_k}\left(\frac{\mathbf{X}_k\mathbf{c}}{\mu^k}-\mathbf{e}\right)\right\|$$

Show that $\delta(\mathbf{x}^k(\mu^k), \mu^k) = 0$ and $\mathbf{z}(\mathbf{x}^k(\mu^k), \mu^k) = \mathbf{z}(\mu^k)$.

(c) A new solution is given by $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}^k_\mu$. Show that

$$\mathbf{x}^{k+1} = 2\mathbf{x}^k - \frac{\mathbf{X}_k^2\mathbf{z}\left(\mathbf{x}^k, \mu^k\right)}{\mu^k}$$

(d) Prove that, if $\delta(\mathbf{x}^k, \mu^k) < 1$, then $\mathbf{x}^{k+1}$ is an interior feasible solution to (P). Moreover, $\delta(\mathbf{x}^{k+1}, \mu^k) \leq \delta(\mathbf{x}^k, \mu^k)$. This implies that if we repeatedly replace $\mathbf{x}^k$ by $\mathbf{x}^{k+1}$, with fixed μ^k, then we obtain a sequence of points which converges to $\mathbf{x}^*(\mu^k)$ quadratically.

(e) Choose $0 < \theta < 1$ and let $\mu^{k+1} = (1-\theta)\mu^k$. Show that

$$\delta\left(\mathbf{x}^{k+1}, \mu^{k+1}\right) < \frac{1}{1-\theta}\left(\delta(\mathbf{x}^{k+1}, \mu^k) + \theta\sqrt{n}\right)$$

(f) Let $\delta(\mathbf{x}^k, \mu^k) \leq 1/2$ and $\theta = 1/6\sqrt{n}$. Show that $\delta(\mathbf{x}^{k+1}, \mu^{k+1}) \leq 1/2$.

(g) Notice that when $\mathbf{z}(\mathbf{x}^k, \mu^k)$ is determined, then $\mathbf{y}(\mathbf{x}^k, \mu^k)$ is also determined by $\mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{c}$. Now, if $\delta(\mathbf{x}^k, \mu^k) \leq 1$, show that $\mathbf{y}(\mathbf{x}^k, \mu^k)$ is dual feasible. Moreover,

$$\mu^k(n - \delta(\mathbf{x}^k, \mu^k)\sqrt{n}) \leq \mathbf{c}^T\mathbf{x}^k - \mathbf{b}^T\mathbf{y}(\mathbf{x}^k, \mu^k) \leq \mu^k(n + \delta(\mathbf{x}^k, \mu^k)\sqrt{n})$$

(h) Given an initial interior feasible solution $\mathbf{x}^0 > \mathbf{0}$ and a barrier parameter $\mu^0 > 0$ such that $\delta(\mathbf{x}^0, \mu^0) \leq 1/2$. Also let q be a large positive integer. We state our algorithm as follows:

```
begin
    $\theta := 1/6\sqrt{n}$, $\mathbf{x} := \mathbf{x}^0$, $\mu := \mu^0$;
    while $n\mu > e^{-q}$ do
    begin
        $\mathbf{z} := \mathbf{z}(\mathbf{x}, \mu)$;
        $\mathbf{x} := 2\mathbf{x} - \frac{\mathbf{X}^2\mathbf{z}}{\mu}$;
        $\mu := (1-\theta)\mu$;
    end
end
```

Let $q^0 = -\log_e(n\mu^0)$, and show that the algorithm terminates after at most $6(q-q^0)\sqrt{n}$ iterations. The final points $\mathbf{x}$ and $\mathbf{y}(\mathbf{x}, \mu)$ are interior solutions satisfying

$$\mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{y}(\mathbf{x}, \mu) \leq \frac{3}{2}e^{-q}$$

7.9. For the dual affine scaling algorithm, explain the meaning of "primal estimate" as defined in (7.59).

7.10. For the primal-dual algorithm, try to decompose $\mathbf{d}_w^k$ and $\mathbf{d}_s^k$ as we did for $\mathbf{d}_x^k$ in (7.113). Then analyze different components.

7.11. We take $\mathbf{x}^0 = \mathbf{e}$, $\mathbf{w}^0 = \mathbf{0}$, and $\mathbf{s}^0 = \mathbf{e}$.

(a) Show that (7.112a) becomes $\pi > 0$.

(b) Show that (7.112b) becomes $\lambda > n - \mathbf{c}^T\mathbf{e}$.

(c) What about (7.112c) and (7.112d)?

7.12. Derive the power-series expansions for $\mathbf{x}(\beta)$, $\mathbf{w}(\beta)$, $\mathbf{s}(\beta)$, $\mathbf{t}(\beta)$, $\mathbf{u}(\beta)$, and $\mathbf{v}(\beta)$ in the primal-dual algorithm.

7.13. Develop computer codes for the primal affine scaling, dual affine scaling, and primal-dual algorithms and test those problems in Exercise 3.16.