Chapter 10

Interior-Point Methods for Linear Programming

We studied two pivoting algorithms for linear programming in Chapter 4. These algorithms are finite and the simplex method in particular is known to be very efficient practically. Yet, there is no known pivoting algorithm that is polynomial. There are pathological examples of linear programs for which the simplex method (or the criss-cross method) behaves badly, i.e., generates an exponential number of pivots to find an optimal solution. There is a famous example due to Klee and Minty, [28, 35]. An explicit description is given by

$$\begin{array}{llrcl} & \max & & x_n & \\ (10.1) & \text{subject to} & 0 \leq & x_1 & \leq 1, \\ & & \epsilon\, x_{j-1} \leq & x_j & \leq 1 - \epsilon\, x_{j-1}, \forall j = 2, \ldots n \end{array}$$

where $0 < \epsilon < 1/2$.

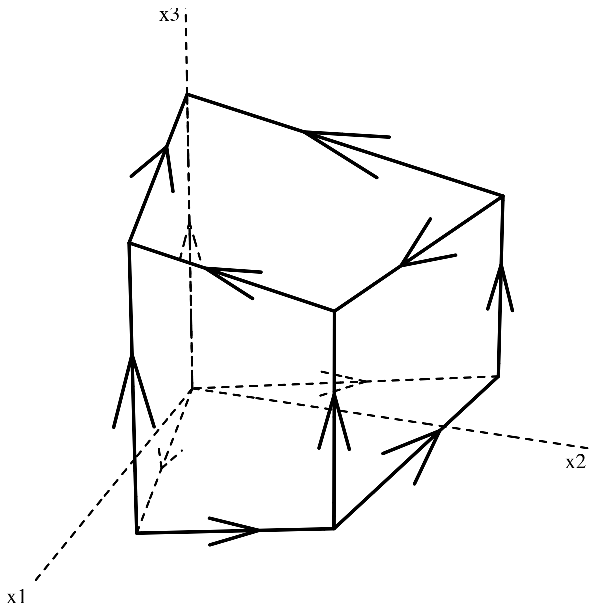


Figure 10.1: Three Dimensional Klee-Minty's Example with $\epsilon = 1/5$

As you can see in Figure 10.1, the feasible region is combinatorially a n dimensional hypercube with 2^n extreme points and the simplex method can visit all 2^n extreme points. This example is given by a system of $2n$ inequalities in n variables and in particular the binary encoding length of the problem is $O(n)$. Thus, any polynomial algorithm uses at most a polynomial number of arithmetic operations in terms of n. The orientation by the objective function has a recursive structure. Namely, the orientation restricted to the (bottom) facet associated with $\epsilon x_{n-1} \leq x_n$ is reversed on the opposite facet associated with $x_n \leq 1 - \epsilon x_{n-1}$. Each of these orientations are isomorphic to the orientation of an $(n-1)$ dimensional Klee-Minty's example. A typical exponential behavior of the simplex method goes though all vertices of the bottom facet first, moves up to the top facet, and then goes through all vertices of the facet to reach the optimal vertex.

The first polynomial algorithm for linear programming was proposed by Khachiyan [27] in 1979. The algorithm is known as the *ellipsoid method.* Although it was considered a breakthrough, the algorithm is not known to be practical because it is difficult to implement it with a reasonable arithmetic precision. The first practical polynomial algorithm (class), known as interior-point methods, was invented by Karmarker [26] in 1984, and many variations have been proposed afterwards including the primal-dual interior-point methods.

The purpose of this chapter is to give a brief description of the primal-dual interior-point methods. Our main goals are to give a basic theoretical framework and to explain key algorithmic components which were developed in the field of nonlinear continuous optimization.

10.1 Notions

For a function $f : \mathbb{R}^n \to \mathbb{R}$, the *gradient* ∇f of f is defined as the vector of its partial derivatives

$$\nabla f(x) := \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \tag{10.2}$$

Of course, for the gradient to be defined, the function must be differentiable.

The matrix of second partial derivatives of f is the *Hessian matrix* denoted by $H(x)$ or $\nabla^2 f(x)$ defined by

$$\nabla^2 f(x) = H(x) := \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ & \ddots & \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{bmatrix} \tag{10.3}$$

In this chapter, for simplicity, we assume that any function f is twice continuously differentiable, i.e. it has a second derivative that is continuous. This class of function is denoted by C^2.

For example, when $f(x) = x_1^2 - x_1x_3 + x_2^2 + 3x_2 + 2x_3^2$,

$$\nabla f(x) = \begin{bmatrix} 2x_1 - x_3 \\ 2x_2 + 3 \\ -x_1 + 4x_3 \end{bmatrix} \qquad \nabla^2 f(x) = H(x) = \begin{bmatrix} 2 & 0 & -1 \\ 0 & 2 & 0 \\ -1 & 0 & 4 \end{bmatrix}. \tag{10.4}$$

Note that when $f(x)$ is a quadratic function as for this example, the Hessian matrix is a constant matrix.

Taylor's theorem or the mean value theorem states that for a function $f \in C^2$, for any $x_0, x \in \mathbb{R}$ and $\Delta x := x - x_0$,

$$f(x) = f(x_0) + \nabla f(x_0 + \theta\Delta x)^T \Delta x, \text{ for some } 0 \le \theta \le 1 \tag{10.5}$$

$$f(x) = f(x_0) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T\, H(x_0 + \theta\Delta x)\Delta x, \text{ for some } 0 \le \theta \le 1. \tag{10.6}$$

When f is a linear (quadratic) function, the first (second, respectively) equation above is satisfied with $\theta = 0$. The *linear approximation* $\bar{f}$ and the *quadratic approximation* $\bar{\bar{f}}$ of a function f at x_0 are define by

$$\bar{f}(x) := f(x_0) + \nabla f(x_0)^T (x - x_0) \tag{10.7}$$

$$\bar{\bar{f}}(x) := f(x_0) + \nabla f(x_0)^T (x - x_0) + \frac{1}{2}(x - x_0)^T\, H(x_0)(x - x_0), \tag{10.8}$$

which are exact when f is linear and quadratic, respectively.

Let's look at a little example of one-variable function. Consider the function in one variable

$$f(x) = -x^3 + x^2 - 7x - 145. \tag{10.9}$$

Figure 10.2 depicts the behavior of this polynomial of order 3 and its linear and quadratic approximations at $x = -10$. Although these functions are quite different from the original function, at a small neighborhood of $x = -10$, all three functions are similar.

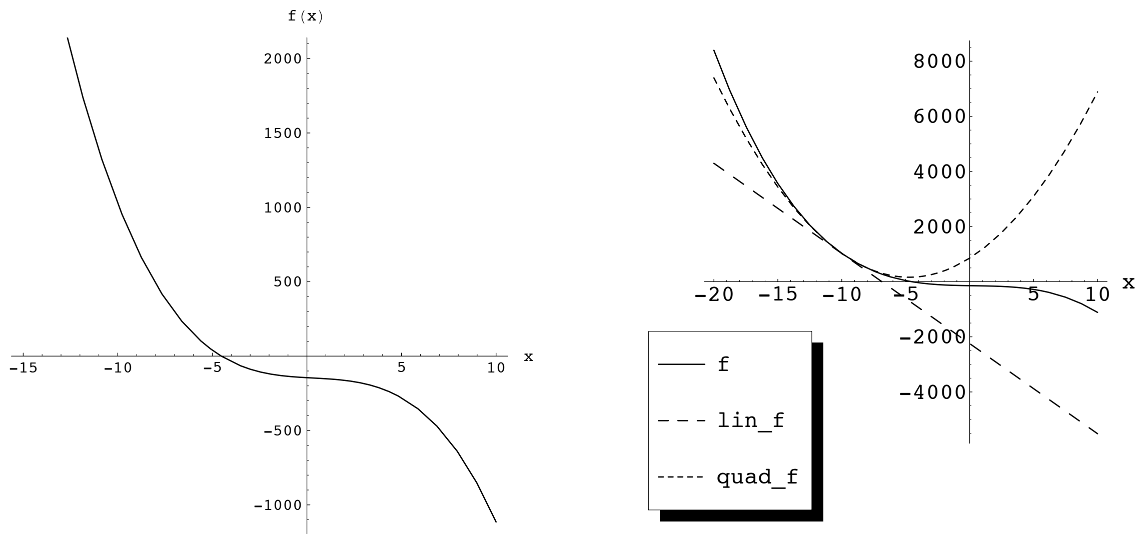


Figure 10.2: The function $-x^3 + x^2 - 7x - 145$ and its linear and quadratic approximations at $x = -10$

10.2 Newton's Method

Newton's method is to find a zero (root) of a vector-valued function $F : \mathbb{R}^n \to \mathbb{R}^n$.

Let's first look at the simplest case of one-variable function $f : \mathbb{R} \to \mathbb{R}$. The basic idea is quite simple. Newton's method starts with an initial point $x_0 \in R$, finds a root x_1 of the linearization (10.7) of f and repeats the same with the new point x_1. Figure 10.3 depicts the first two iterations applied to the function $-x^3 + x^2 - 7x - 145$ with an initial point $x_0 = -12$.

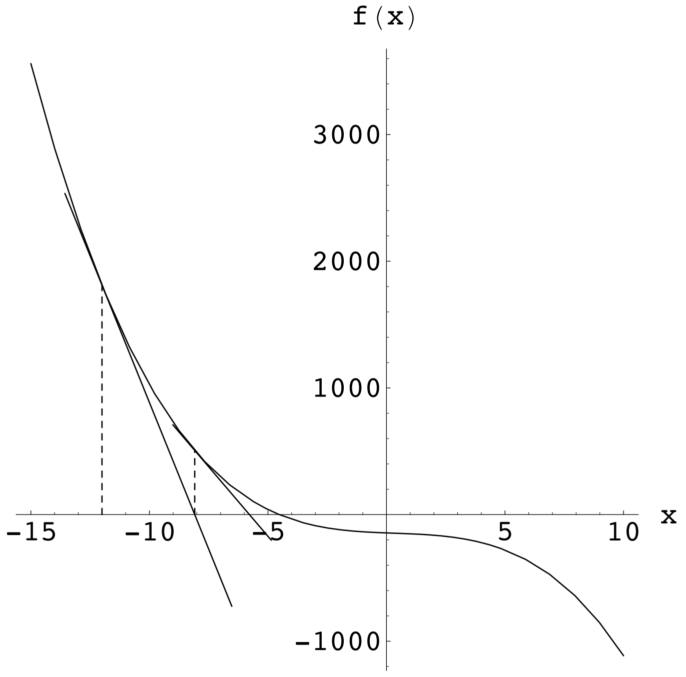


Figure 10.3: First two iterations of Newton's method applied to the function $-x^3 + x^2 - 7x - 145$ starting at $x_0 = -12$

Recall that the linear approximation of f at x_k is given by

$$\bar{f}(x) = f(x_k) + \nabla f(x_k)^T (x - x_k).$$

Thus, the key equation to find a root of $\bar{f}$ is

(10.10) $$\nabla f(x_k)^T \Delta x_k = -f(x_k), \text{ or more explicitly,}$$

(10.11) $$\Delta x_k = -\nabla f(x_k)^{-1} f(x_k).$$

Therefore, Newton's method updates x_k with

(10.12) $$x_{k+1} := x_k + \Delta x_k.$$

More generally, for a vector valued function $F : \mathbb{R}^n \to \mathbb{R}^n$ with $F(x) = (f_1(x), \ldots, f_n(x))^T$ and $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, n$, Newton's step is given by

(10.13) $$\nabla f_i(x_k)^T \Delta x_k = -f_i(x_k) \text{ for } i = 1, \ldots, n.$$

Here a step vector $\Delta x_k \in \mathbb{R}^n$ is to be determined given a current point $x_k \in \mathbb{R}^n$. A more compact expression in matrix form of these simultaneous equations is

$$J(x_k)\Delta x_k = -F(x_k), \tag{10.14}$$

where $J(x)$ is the $n \times n$ *Jacobian matrix* of F defined by

$$J(x) := \begin{bmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_n(x)^T \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} \dots \frac{\partial f_1(x)}{\partial x_n} \\ \ddots \\ \frac{\partial f_n(x)}{\partial x_1} \dots \frac{\partial f_n(x)}{\partial x_n} \end{bmatrix}. \tag{10.15}$$

In general, Newton's method might not converge to a root even if exists. For example, it is not hard to find a one-variable example where Newton's method swings between two points forever. However, under some assumptions on the function and the initial point, one can state fast conversion theorems such as the superlinear convergence. This is the case when Newton's Method is applied to "almost-linear" systems in primal-dual interior-point methods given in the next section.

Exercise 10.1 *Consider a function $F(x) = (f_1(x), \dots, f_n(x))^T$ and $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \dots, n$, which is partially linear, namely,*

$$f_i(x) = A_i x - b_i \text{ for } i = 1, \dots, k (\leq n). \tag{10.16}$$

Show that when you apply Newton's method to this class of functions F with an initial point x_0 such that $f_i(x_0) = A_i x_0 - b_i = 0$ for all $i = 1, \dots, k$, all subsequent points will satisfy these linear equations. How many iterations does Newton's method take to satisfy all linear equations, if the initial point violates some?

Exercise 10.2 *One often applies Newton's algorithm to find a global minimizer of a function $f : \mathbb{R}^n \to \mathbb{R}$ by searching for a point $\overline{x}$ such that $\nabla f(\overline{x}) = \mathbf{0}$. Let $f(x) = x_1^2 + 3x_1 - 2x_1x_2 + ax_2^2$ for some $a \in \mathbb{R}$. Explain the first Newton iteration with an initial point $x_0 \in \mathbb{R}^2$ with $(x_0)_1 = (x_0)_2 = 1$. When does it find a global minimizer? When does it fail? When it works, how many iterations does it take?*

10.3 Primal-Dual Interior-Point Methods

Consider an LP in standard form with its dual.

$$\begin{array}{lll} \max & c^T x & \\ \text{subject to} & Ax & = b \\ & x & \geq \mathbf{0}, \end{array} \tag{10.17}$$

$$\begin{array}{lllll} \min & b^T y & & & \\ \text{subject to} & A^T y & -s & = c & \\ & & s & \geq \mathbf{0}. & \end{array} \tag{10.18}$$

where $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^n$ are given.

The complementarity slackness condition says a dual pair of feasible solutions x and (y, s) are both optimal if and only if x and s are orthogonal, i.e.,

$$x^T s = \sum_{j=1}^{n} x_j s_j = 0. \tag{10.19}$$

It will be convenient to introduce a notation

$$X := diag(x_1, x_2, \dots, x_n), \qquad S := diag(s_1, s_2, \dots, s_n)$$

where $diag(a_1, a_2, \dots, a_n)$ denotes the $n \times n$ diagonal matrix of diagonal entries a_1, …, a_n. Using this notation, a dual pair of x and (y, s) are both optimal if and only if

$$x \geq \mathbf{0} \text{ and } s \geq \mathbf{0} \text{ and} \tag{10.20}$$

$$F(x, y, s) = \mathbf{0}, \tag{10.21}$$

where

$$F(x, y, s) := \begin{bmatrix} A^T y - s - c \\ Ax - b \\ XSe \end{bmatrix}. \tag{10.22}$$

Let us denote by X the set of vectors (x, y, s) such that x is primal feasible and (y, s) is dual feasible, and X^0 the relative interior of X, i.e.,

$$X := \{(x, y, s) : A^T y - s = c,\ Ax = b,\ x \geq \mathbf{0} \text{ and } s \geq \mathbf{0}\}, \tag{10.23}$$

$$X^0 := \{(x, y, s) \in X : x > \mathbf{0} \text{ and } s > \mathbf{0}\}. \tag{10.24}$$

The generic primal-dual interior point methods can be written as follows.

(10.25) **Generic Primal-Dual Interior-Point Algorithm**

- start with a point $(x, y, s) \in X^0$.
- generate a new point $(x, y, s) + \alpha \cdot (\Delta x, \Delta y, \Delta s)$ where the search direction $(\Delta x, \Delta y, \Delta s)$ is obtained by Newton's method applied to a slight modification of F, and $\alpha \leq 1$ is a step length (for example, to make sure a new point stays in X^0).
- repeat.

Of course, finding a point in X^0 is itself a nontrivial problem and may not be possible even if X is nonempty. There are other technical issues as well that need to be resolved, see Section 10.3.3. For the sequel we assume a point $(x, y, s) \in X^0$ is given.

10.3.1 Newton's Method Directly Applied to F

Here, we discuss the special case of the generic algorithm (10.25) where Newton's method is applied directly to F defined in (10.22) and the step length α is 1. This means that the basic incremental step $(\Delta x, \Delta y, \Delta s)$ is computed by solving

$$\begin{bmatrix} \mathbf{0} & A^T & -I \\ A & \mathbf{0} & \mathbf{0} \\ S & \mathbf{0} & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ -XSe \end{bmatrix}. \tag{10.26}$$

This leads to the following algorithm.

(10.27) **Naive Primal-Dual Algorithm**

- start with a point $(x, y, s) \in X^0$.
- generate a new point $(x, y, s) + (\Delta x, \Delta y, \Delta s)$ where the search direction $(\Delta x, \Delta y, \Delta s)$ is computed by solving (10.26).
- repeat.

Does this algorithm work? The problem is that even if we start with a point $(x, y, s) \in X^0$, there is no guarantee that the next point stays in X^0 or even in X. Newton's algorithm applied to F search for some root of F without respecting the nonnegativity of x and s.

Example 10.1 *Let's look at the Chataux ETH wine production problem, Example 1.1. The matrix A, the vectors c and b of this problem in standard form setting is*

$$A = \begin{bmatrix} -2 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & -2 & 0 & 1 & 0 \\ 0 & -3 & -1 & 0 & 0 & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} 4 \\ 8 \\ 6 \end{bmatrix}, \qquad c^T = \begin{bmatrix} 3 & 4 & 2 & 0 & 0 & 0 \end{bmatrix}. \tag{10.28}$$

One can easily verify that (x, y, s) as fixed below satisfifies $(x, y, s) \in X^0$.

$$x^T = \begin{bmatrix} 1 & 1 & 1 & 2 & 5 & 2 \end{bmatrix}, \qquad y^T = \begin{bmatrix} 2 & 2 & 2 \end{bmatrix}, \qquad s^T = \begin{bmatrix} 3 & 2 & 4 & 2 & 2 & 2 \end{bmatrix}. \tag{10.29}$$

At this point (x, y, s), the evaluation of F is

$$F(x, y, s)^T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 2 & 4 & 4 & 10 & 4 \end{bmatrix}. \tag{10.30}$$

In Figure 10.4, the numbers 1, 2, 3, 4, 5 *indicate the location of the initial point, the second point,.., the fifth point generated by the Naive Primal-Dual Method. The second point in x-space* $(1.39431, 1.33563, 1.10267, 1.21139, 4.40035, 0.890423)$ *is still totally positive, while the second point in s-space* $(-1.18292, -0.671268, -0.410699, 0.788611, 0.239862, 1.10958)$ *is not totally positive, The last three points in x-space are not totally positive and thus not feasible. It converges to a feasible extreme point which is not optimal.*

Such an behavior is not surprising because the naive primal-dual algorithm does not respect the positivity of x and s at all.

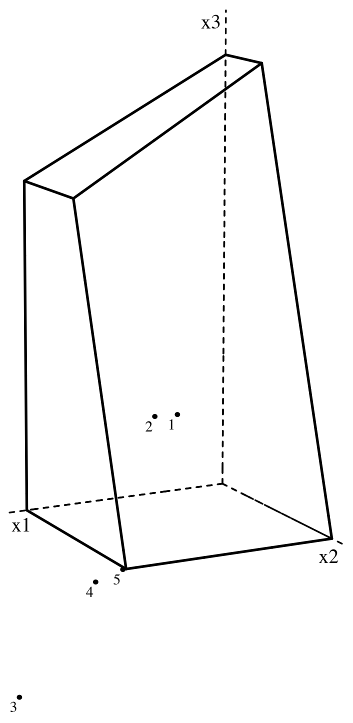


Figure 10.4: First five points generated by the naive primal-dual algorithm

10.3.2 The Central Path

As seen in the previous section, the naive primal-dual interior-point method does not work as generated points (x, y, s) do not stay in X^0 and has no guarantee to converge to an optimal point. To make the naive algorithm work, we must modify the function F in such a way that the points in X^0 satisfying $F = \mathbf{0}$ is not far from the current point. One way to do this is to parametarise F as

$$(10.31) \qquad F_\tau(x, y, s) := \begin{bmatrix} A^T y - s - c \\ Ax - b \\ XSe - \tau e \end{bmatrix}, \text{ for any } \tau \geq 0.$$

The only difference from the original F is the last part $XSe - \tau e$. The effect is that $F_\tau(x, y, s) = \mathbf{0}$ implies

$$(10.32) \qquad x_j s_j = \tau, \ \forall j = 1, \ldots, n,$$

which are relaxed complementary slackness conditions. The *central path* is the set of all feasible solutions satisfying these equations above for some τ,

$$(10.33) \qquad C := \{(x, y, s) \in X^0 : \exists \tau \geq 0 \text{ such that } x_j s_j = \tau, \ \forall j = 1, \ldots, n\}.$$

When we have a point $(x, y, s) \in X^0$, we can use the following *average distance* from the complementary slackness to control an algorithm to follow the central path approximately,

$$(10.34) \qquad \mu(x, s) := (\sum_{j=1}^{n} x_j s_j)/n.$$

This distance can be also called the *average duality gap*. A simple primal-dual algorithm applies Newton's method to $F_{\sigma\mu(x,s)}$, which results in trying to solve

$$x_j s_j = \sigma\mu(x,s),\ \forall j = 1,\dots,n, \tag{10.35}$$

for some $0 \le \sigma \le 1$.

This leads to the following algorithm.

(10.36) **Simple Primal-Dual Algorithm**(σ)

- start with a point $(x, y, s) \in X^0$.
- generate a new point $(x, y, s) + (\Delta x, \Delta y, \Delta s)$ where the search direction $(\Delta x, \Delta y, \Delta s)$ is computed by applying Newton's method to $F_{\sigma\mu(x,s)} = \mathbf{0}$.
- repeat.

When $\sigma = 0$, this algorithm coincides with the naive primal-dual algorithm. When $\sigma = 1$, this algorithm tries to find a point on C where all $x_j s_j$'s equal to the average duality gap at the current point. This is just too conservative to aim for an optimal solution, as the average distance may not decrease at all. This algorithm does not guarantee that the new point $(x, y, s) + (\Delta x, \Delta y, \Delta s)$ is feasible, but it tends to stay in X^0 because it tries to get closer to the central path.

Example 10.2 *Let's look at the Chataux ETH wine production problem, Example 1.1, again. Figure 10.5 shows the behavior of the basic primal-dual algorithm for three different settings of $\sigma = 0.6$, $\sigma = 0.4$ and $\sigma = 0.2$. The smaller σ is, the faster the conversion. Yet, when σ is too small ($\sigma = 0.2$), it may not follow the central path, as you see the 4th point generated is very close to a boundary of the feasible region. When σ is large ($\sigma = 0.8$), the algorithm is very conservative and works slowly.*

Figure 10.6 depicts the behavior of the basic primal-dual algorithm with a conservative setting of $\sigma = 0.9$. As you easily see, the first ten points stay in a neighborhood of the initial point. The later sequence of points follows closely to the central path, and eventually gets very close to the optimal solution.

Finally, the behavior of the basic primal-dual algorithm applied to the Klee-Minty's example (10.1) with $n = 3$ and $\epsilon = 1/5$ is shown in Figure 10.7. The starting interior point is $(1/2, 1/2, 1/2)$.

Exercise 10.3 *Using your favorite computer language, write an implementation of the simple primal-dual interior-point method (10.36). Using a very high level language like Maple or Mathematica, it is also easy to plot the sequence of generated points and to compare your plots with Figure 10.5, for example.*

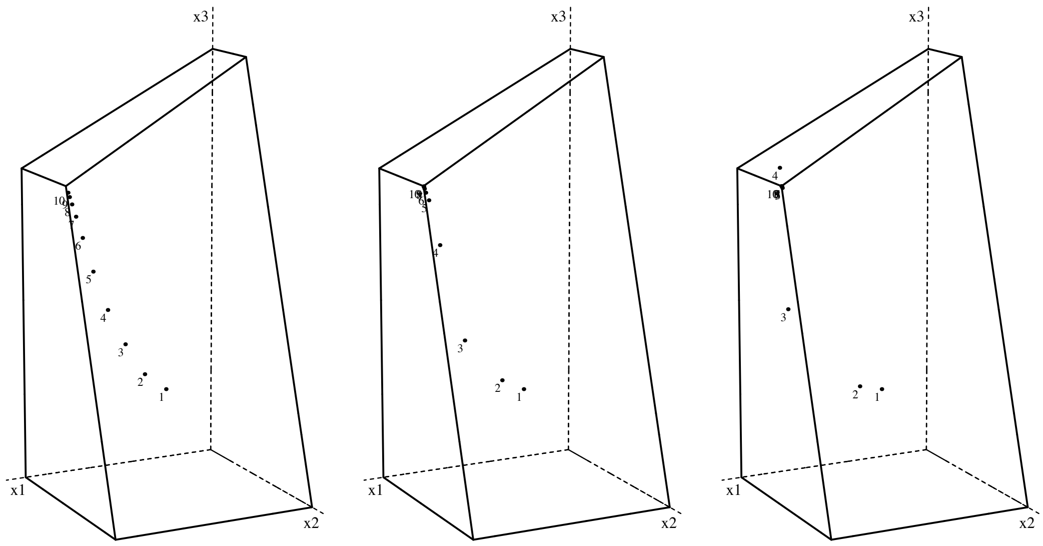


Figure 10.5: First ten points generated by the simple primal-dual algorithm with $\sigma = 0.6$ (left), $\sigma = 0.4$ (middle), $\sigma = 0.2$ (right)

10.3.3 Polynomial Complexity

The previous section describes the basic idea of following the central path and converging to an optimal solution. In order to make this idea work to design a polynomial-time algorithm for linear programming, there are still a few more basic ideas to be incorporated carefully.

For discussions below, recall that input data c, b and A are all integral. We also use L as the length of all these inputs in binary representation, or more exlicitely

$$L := L_0 + mn + m + n,$$

where L_0 is the total length of binary encodings of each components c_j, b_i and a_{ij}. One can interpret the term mn as the number of spaces (separators) needed to separate all a_{ij}'s, n as that to separate c_j's and m to separate b_i's.

Purification. This is a procedure to move from any feasible solution x to a basic feasible solution x' in such a way that the objective value at x' is at least as large as at x. This procedure is sometimes called as a *raindrop procedure* in which a raindrop falls from an interior of the feasible region until it hits a bottom extreme point. In this setting, the objective vector c points vertically downward. It is not difficult to imitate this behavior algebraically using some kind of pivot operations.

Satisfactory Neighborhood. When can we say that a generated point (x^k, y^k, s^k) is close enough to a basic optimal solution so that the purification leads to a basic optimal solution?

The key observation is the following: given two basic feasible solutions x and x' with different objective values, say $c^T x > c^T x'$, there is a lower bound of the difference $c^T x - c^T x'$ in terms of L, namely, 2^{-L}. This number in binary representation has the

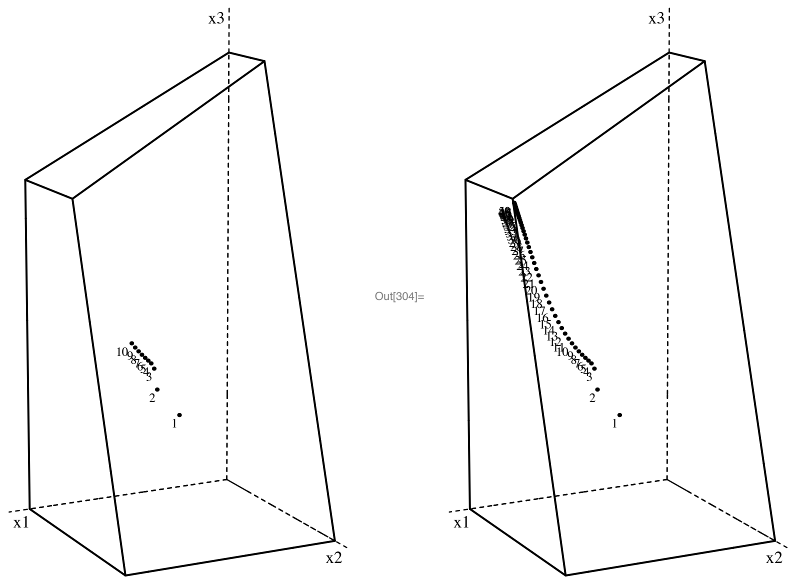


Figure 10.6: First ten and fifty points generated by the simple primal-dual algorithm with $\sigma = 0.9$

number of digits linearly bounded in L (namely, L+constant). Therefore, if $(x^k)^T s^k < 2^{-L}$, then $(c^k)^T x^k = b^T y^k - (x^k)^T s^k \geq b^T y^* - (x^k)^T s^k > z^* - 2^{-L}$, where y^* denotes any dual optimal solution and z^* denotes the optimal value. It follows that $(x^k)^T s^k < 2^{-L}$ is a correct termination criterion.

Fixing Parameter σ and Step Length α. To obtain an explicit polynomial complexity, one needs to control the step length and the parameter very carefully. The most crucial argument is to show the average duality gap $\mu(x^k, s^k)$ rapidly shrinks, such as, for any $\epsilon \in (0,1)$, if our primal-dual method generates a sequence (x^k, y^k, s^k) such that

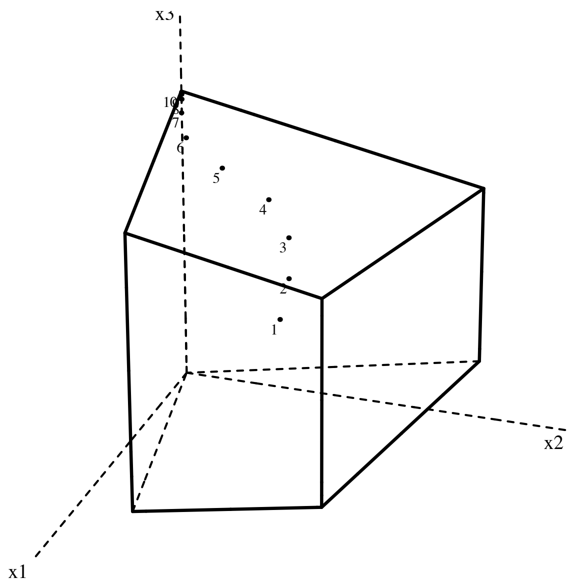
$$\mu(x^{k+1}, s^{k+1}) \leq \left(1 - \frac{\delta}{n^\omega}\right) \mu(x^k, s^k) \tag{10.37}$$

for some positive constant δ and ω, and the initial average duality gap is small, i.e., $\mu(x^0, s^0) \leq 1/\epsilon^\kappa$ for some positive constant κ, then there is an index $K = O(n^\omega |\log \epsilon|)$ such that

$$\mu(x^k, s^k) \leq \epsilon \quad \forall k \geq K. \tag{10.38}$$

By setting $\epsilon = n 2^{-L}$, we obtain the necessary number of iterations of form $O(n^\omega L)$ which is a polynomial complexity in terms of input size. Note that this complexity measures the number of iterations, and thus the actual complexity must be multiplied by the complexity of solving Newton's equations which is again polynomially bounded by the input size.

A closely related is the potential reduction algorithm which uses the same search directions as the primal-dual interior-point method but uses step sizes that approximately minimize a potential function that leads to an optimal solution. The resulting algorithm has the best known iteration complexity of $O(\sqrt{n}L)$.



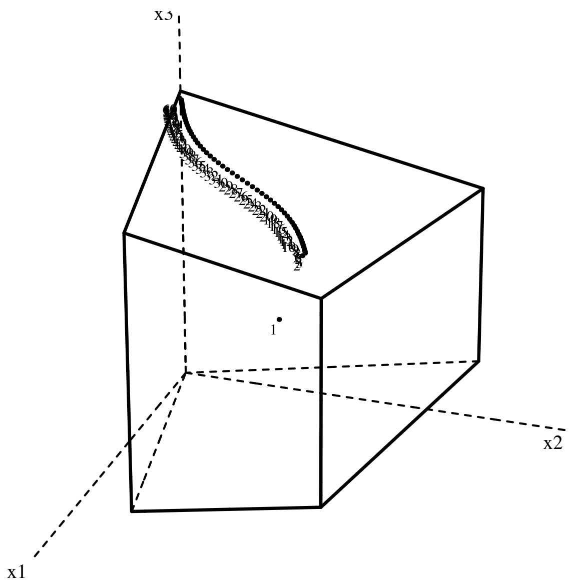


Figure 10.7: The simple primal-dual algorithm applied to a 3D Klee-Minty's Example with $\sigma = 0.4$ (left, showing first 10 points) and $\sigma = 0.9$ (right, first 60 points)

10.3.4 How Important is Polynomial Complexity in Practice?

Many researchers in optimization might argue that interior-point methods are practically efficient because of their polynomial complexity. Unfortunately, there is no correct implementation of a polynomial-time interior-point method. The reason is obvious: in order to evaluate the termination criterion $(x^k)^T s^k < 2^{-L}$, one need to use arithmetic operations of accuracy astronomically precise (i.e. L binary digits accurate) and this cannot be done in any reasonable time even for a moderate size of L, say $10,000$. More over, the rigorous algorithm requires one to solve Newton's system when some components of x^k and s^k become terribly small. Of course, using rational exact arithmetic, one can implement Newton's method correctly but it will take a long long time to solve each system.

There are a plenty of evidences that primal-dual interior-point methods are practically useful to deal with large scale problems, if an implementation simply ignores the termination criterion and replaces it with a wildly simplified version, like, $(x^k)^T s^k < 2^{-38}$. Such a method requires the number of iterations independent of L and can be very useful when the user does not worry about numerical errors too much and, in particular the goal is not proving mathematical statements. On the other hand, if one needs an LP code which is mathematically correct, an implementation of the simplex method with exact rational arithmetic is extremely powerful, as it never has to use L to control the computation. Developing an efficient implementation of interior-point methods that is guaranteed to find a (rational) exact solution remains to be hard and challenging.