Interior-point methods for linear programming: a guided tour*

Image Technical Report 9901

François Glineur
Service de Mathématique et de Recherche Opérationnelle,
Faculté Polytechnique de Mons,
Rue de Houdain, 9, B-7000 Mons, Belgium,
Francois.Glineur@fpms.ac.be.

January 99

*This work is based on the first part of the author's engineering thesis [Gli97] and has been supported by a grant from the F.N.R.S. (Belgian National Fund for Scientific Research).

Abstract

The purpose of *mathematical programming*, a branch of *optimization*, is to minimize (or maximize) a function of several variables under a set of constraints. This is a very important problem arising in many real-world situations (e.g. cost or duration minimization).

When the function to optimize and its associated set of constraints are linear, we talk about *linear programming*[1]. The *simplex algorithm*, first developed by Dantzig in 1947, is a very efficient method to solve this class of problems [Dan63]. It has been thoroughly studied and improved since its first appearance, and is now widely used in commercial software to solve a great variety of problems (production planning, transportation, scheduling, etc.).

However, an article by Karmarkar [Kar84] introduced in 1984 a new class of methods: the so-called *interior-point methods*. Most of the ideas underlying these new methods originate from the nonlinear optimization domain. These methods offer the following advantages:

- **Theoretical efficiency**. These methods require a number of iterations that is polynomially bounded by the size of the problem (while the number of iterations needed by the simplex method may grow exponentially with that size).
- **Practical efficiency**. Computing times of these methods compare very well to those of the simplex method (and even beat them for some large-scale problems).
- **Large-scale optimization**. These methods allow the resolution of some very large problems that were previously unsolvable in a reasonable amount of time.
- **Nonlinear optimization**. It is possible to adapt these methods to nonlinear programming, in particular to convex programming (see e.g. [NN94]), which enables the resolution of some types of problems for which no efficient algorithm was previously known.

The purpose of this article is to give an overview of this rather new domain, providing a clear and understandable description of these methods, both from a theoretical and a practical point of view.

Keywords: linear programming, interior-point methods.

[1] Because of the strong connections between computer science and the term *programming*, some authors prefer to talk about *linear optimization*.

1 Introduction

1.1 Linear programming

The purpose of linear programming is to optimize a linear objective function f depending on n decision variables under a set of linear (equality or inequality) constraints, which can be mathematically stated as (using matrix notation)

$$\min_{x \in \mathbb{R}^n} f(x) = c^T x \quad \text{s.t.} \quad \begin{cases} A_e x = b_e \\ A_i x \geq b_i \end{cases} , \tag{1.1}$$

where vector x contains the n decision variables, vector c defines the objective function and pairs (A_e, b_e) and (A_i, b_i) define the m_e equality and m_i inequality constraints. Column vectors x and c have size n, column vectors b_e and b_i have size m_e and m_i and matrices A_e and A_i have dimensions $m_e \times n$ and $m_i \times n$.

Many linear programs have simpler inequality constraints, e.g. nonnegativity constraints ($x \geq 0$) or bound constraints ($l \leq x \leq u$), possibly in addition to some other ones. The linear programming *standard form* is a special case of linear program used for most theoretical developments of interior-point methods:

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{s.t.} \quad \begin{cases} Ax = b \\ x \geq 0 \end{cases} . \tag{1.2}$$

The only inequality constraints in this format are nonnegativity constraints for *all* variables, i.e. there are no *free* variables (we have thus that m_i is equal to n, A_i is the identity matrix and b_i is the null vector). It is furthermore possible to show that every linear program in the general form (1.1) admits an equivalent program in the standard form, obtainable by adding/removing variables/constraints (by *equivalent problem*, we mean that solving the transformed problem allows us to find the solution of the original one).

1.2 The simplex method

The set of all x satisfying the constraints in (1.2) is a polyhedron in $\mathbb{R}^n$. Since the objective is linear, the constant-cost sets are parallel hyperplanes orthogonal to c and the optimal solution must lie on one of the polyhedron vertices (it is also possible that a whole face of the polyhedron is optimal or that no solution exists).

The main idea behind the *simplex method* is to explore these vertices in an iterative way, moving from the current vertex to an adjacent one improving the objective function value. This is done using an algebraic characterization of vertices called *basis*. When such a move becomes impossible to make, the algorithm stops. Dantzig proved that this always happens after a finite number of moves, and that the resulting vertex is optimal [Dan63].

1.3 A first glimpse on interior-point methods

We are now able to give a first description of interior-point methods. As opposed to the simplex method which uses vertices, these methods start with a point that lies *inside* the set of feasible solutions. Using the standard form notation (1.2), we define the feasible set $\mathcal{P}$ to be the set of vectors x satisfying the constraints, i.e.

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax = b \text{ and } x \geq 0\} \ ,$$

and the associated set $\mathcal{P}^+$ to be the subset of $\mathcal{P}$ satisfying strict nonnegativity constraints

$$\mathcal{P}^+ = \{x \in \mathbb{R}^n \mid Ax = b \text{ and } x > 0\} \ .$$

$\mathcal{P}^+$ is called the *strictly feasible* set[2] and its elements are called *strictly feasible* points.

Interior-point methods are *iterative* methods that compute a sequence of iterates belonging to $\mathcal{P}^+$ and converging to an optimal solution. This is completely different from the simplex method, where an *exact* optimal solution is obtained after a finite number of steps. Interior-point iterates tend to an optimal solution but never attain it (since the optimal solutions do not belong to $\mathcal{P}^+$ but to $\mathcal{P} \setminus \mathcal{P}^+$). This apparent drawback isn't really disturbing since

- ⋄ Most of the time, an approximate solution (with e.g. 10^{-8} relative accuracy) is enough for all purposes.
- ⋄ A *rounding* procedure can convert a nearly optimal interior point into an exact optimal vertex solution (see e.g. [CRV97]).

Another significant difference occurs when an entire face of $\mathcal{P}$ is optimal: interior-point methods converge to the interior of that face while the simplex method ends on one of its vertices.

The last difference we'd like to point out at this stage is about algorithmic complexity. While the simplex method may potentially make a number of moves that grows exponentially with the problem size, interior-point methods need a number of iterations that is polynomially bounded by the problem size to attain a given accuracy. This property is with no doubt mainly responsible for the huge amount of research that has been done on the topic of interior-point methods for linear programming.

1.4 A short historical account

The purpose of this paragraph is not to be exhaustive but rather to give some important milestones in the development of interior-point methods.

1.4.1 First steps of linear programming

1930-1940 First appearance of linear programming formulations.

1939-1945 Second World War: operations research makes its debuts with military applications.

1947 Georges B. Dantzig publishes the first article about the simplex method for linear programming [Dan63].

[2] $\mathcal{P}^+$ is in fact the relative interior of $\mathcal{P}$, see [Roc70].

1970 V. Klee and G. Minty prove that the simplex method has exponential worst-case complexity [KM72].

1.4.2 First steps of interior point methods

1955 K. R. Frisch proposes a *barrier* method to solve nonlinear programs [Fri55].

1968 A. V. Fiacco and G. P. McCormick develop barrier methods for convex nonlinear programming [FM68].

1978 L. G. Khachiyan applies the *ellipsoid* method (developed by N. Shor in 1970 [Sho70]) to linear programming and proves that it is polynomial [Kha79].

It is important to note that these barrier methods are developed in the nonlinear framework. Although they are applicable to linear programming, their authors do not consider them as viable competitors to the simplex method. We also point out that the theoretical advantage of the ellipsoid method over the simplex algorithm led to some disappointment due to a very slow practical behaviour[3].

1.4.3 The interior-point revolution

1984 N. Karmarkar discovers a polynomial interior-point method that is much more efficient than the ellipsoid method. He also claims superior performance compared to the simplex method [Kar84].

1994 Y. Nesterov and A. Nemirovsky publish a monograph on polynomial interior-point methods for convex programming [NN94].

1997 Since Karmarkar's first breakthrough, more than 3000 articles have been published on the topic of interior point methods. A few textbooks have been published (see e.g. [Wri97, CRV97, Ye97]). Research is now concentrating on nonlinear programming, especially on convex programming.

Karmarkar's algorithm was not competitive with the best simplex implementations, especially on small-scale problems, but his announcement concentrated a stream of research on the topic. We also point out that Khachiyan's method is not properly speaking the first polynomial algorithm for linear programming, since Fiacco and McCormick's method has been shown *a posteriori* to be polynomial by Anstreicher [Ans90].

2 Building blocks

In this section, we are going to review the different concepts needed to get a correct understanding of interior-point methods. We start with the very well studied notion of duality for linear programming.

[3]The simplex method only shows an exponential complexity on some hand-crafted linear programs and is much faster on real-world problems, while the ellipsoid method always shows its worst-case polynomial number of iterations.

2.1 Duality

Let us state again the standard form of a linear program

$$\min_{x\in\mathbb{R}^n} c^T x \quad \text{s.t.} \quad \begin{cases} Ax = b \\ x \geq 0 \end{cases} . \tag{P}$$

Using the same data (viz. A, b and c) it is possible to describe another linear program

$$\max_{y\in\mathbb{R}^m} b^T y \quad \text{s.t.} \quad \begin{cases} A^T y \leq c \\ y \text{ is free} \end{cases} .$$

As we will see later, this program is tightly related to (P) and will be called the *dual* of (P) (which will be called *primal* program). It is readily seen that this program may also be written as

$$\max_{y\in\mathbb{R}^m, s\in\mathbb{R}^n} b^T y \quad \text{s.t.} \quad \begin{cases} A^T y + s = c \\ s \geq 0 \text{ and } y \text{ free} \end{cases} . \tag{D}$$

This extra *slack* vector s will prove useful in simplifying our notations and we will therefore mainly use this formulation of the dual. We also define the dual feasible and strictly feasible sets $\mathcal{D}$ and $\mathcal{D}^+$ similarly to $\mathcal{P}$ and $\mathcal{P}^+$

$$\begin{aligned} \mathcal{D} &= \left\{(y, s) \mid A^T y + s = c \text{ and } s \geq 0\right\} , \\ \mathcal{D}^+ &= \left\{(y, s) \mid A^T y + s = c \text{ and } s > 0\right\} . \end{aligned}$$

From now on, we will assume that matrix A has full row rank, i.e. that its rows are linearly independent[4]. Because of the equation $A^T y + s = c$, this implies a one-to-one correspondence between the y and s variables in the dual feasible set. In the following, we will thus refer to either (y, s), y or s as the dual variables.

We now state various important facts about duality:

- ◇ If x is feasible for (P) and (y, s) for (D), we have $b^T y \leq c^T x$. This means that any feasible point of (D) provides a lower bound for (P) and that any feasible point of (P) provides an upper bound for (D). This is the *weak* duality property. The nonnegative quantity $c^T x - b^T y$ is called the *duality gap* and is equal to $x^T s$.
- ◇ One of these three situations occurs for problems (P) and (D)
  1. Both problems have finite optimal solutions
  2. One problem is unbounded (i.e. its optimal value is infinite) and the other one is impossible (i.e. its feasible set is empty)
  3. Both problems are impossible
- ◇ x and (y, s) are optimal for (P) and (D) if and only if the duality gap is zero. This is the *strong* duality property. This implies that when both problems have optimal solutions, their objective values are equal. In that case, since $x^T s = 0$ and $x \geq 0$, $s \geq 0$, we have that all products $x_i s_i$ must be zero, i.e. at least one of x_i and s_i is zero for each i (this is known as *complementary slackness*).

[4] This is done without loss of generality: if a row of A is linearly depending on some other rows, we have that the associated constraint is either redundant (and can be safely ignored) or impossible to satisfy (leading to an impossible problem), depending on the value of the right-hand side vector b.

2.2 Optimality conditions

Karush-Kuhn-Tucker (KKT) conditions are necessary optimality conditions pertaining to nonlinear constrained optimization with a differentiable objective. Moreover, they are sufficient when the problem is convex, which is the case for linear programming. For problem (P) they lead to the following system

$$x \text{ is optimal for (P)} \quad \Leftrightarrow \quad \exists\,(z,t) \quad \text{s.t.} \quad \left\{ \begin{array}{rcl} Ax & = & b \\ A^T z + t & = & c \\ x_i t_i & = & 0\ \forall i \\ x \text{ and } t & \geq & 0 \end{array} \right. .$$

The second equation closely resembles the equality constraint for the dual problem (D): indeed, if we identify z with y and t with s we find

$$x \text{ is optimal for (P)} \quad \Leftrightarrow \quad \exists\,(y,s) \quad \text{s.t.} \quad \left\{ \begin{array}{rcl} Ax & = & b \\ A^T y + s & = & c \\ x_i s_i & = & 0\ \forall i \\ x \text{ and } s & \geq & 0 \end{array} \right. .$$

Finally, using the definitions of $\mathcal{P}$ and $\mathcal{D}$ and the fact that when u and v are nonnegative

$$u_i v_i = 0\ \forall i \Leftrightarrow \sum_i u_i v_i = 0 \Leftrightarrow u^T v = 0$$

we have

$$x \text{ is optimal for (P)} \quad \Leftrightarrow \quad \exists\,(y,s) \quad \text{s.t.} \quad \left\{ \begin{array}{rcl} x & \in & \mathcal{P} \\ (y,s) & \in & \mathcal{D} \\ x^T s & = & 0 \end{array} \right. .$$

This is in fact a confirmation of the strong duality theorem, revealing the deep connections between a problem and its dual: a necessary and sufficient condition for the optimality of a feasible primal solution is the existence of a feasible dual solution with zero duality gap (i.e. the same objective value).

Similarly, applying the KKT conditions to the dual problem would lead exactly to the same set of conditions, requiring the existence of a feasible primal solution with zero duality gap.

2.3 Newton method

The fact that finding the optimal solution of a linear program is completely equivalent to solving the KKT conditions may suggest the use of a general method designed to solve systems of nonlinear equations[5]. The most popular of these methods is the *Newton method*, whose principle is described in the following paragraph.

[5]Strictly speaking, the first two conditions are linear while only the $x_i s_i = 0$ equations are nonlinear. The nonnegativity constraints are not equations and cannot be handled by such a method.

Let $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ be a differentiable nonlinear mapping. The Newton method is an iterative process aiming to find an $x \in \mathbb{R}^n$ such that $F(x) = 0$. For each iterate x_k, the method computes a first-order approximation to F around x_k and sets x_{k+1} to the zero of this linear approximation. Formally, if J is the Jacobian of F (assumed to be nonsingular), we have

$$F(x_k + \Delta x_k) \approx F(x_k) + J(x_k)\Delta x_k$$

and thus we let $x_{k+1} = x_k + \Delta x_k$ where $\Delta x_k = -J(x_k)^{-1}F(x_k)$[6]. Convergence to a solution is guaranteed if the initial iterate x_0 lies in a suitable neighbourhood of one of F's zeros.

Newton method is also applicable to minimization problems in the following way: let $g : \mathbb{R}^n \mapsto \mathbb{R}$ be a function to minimize. We form a second-order approximation to $g(x)$ around x_k, namely

$$g(x_k + \Delta x_k) \approx g(x_k) + \nabla g(x_k)^T \Delta x_k + \frac{1}{2}\Delta x_k{}^T \nabla^2 g(x_k)\Delta x_k \ .$$

If the Hessian $\nabla^2 g(x_k)$ is positive definite, which happens when g is strictly convex, this approximation has a unique minimizer, which we take as next iterate. It is defined by $\Delta x_k = -\nabla^2 g(x_k)^{-1}\nabla g(x_k)$, which leads to a method that is basically equivalent to applying Newton method to the gradient-based optimality condition $\nabla g(x) = 0$.

One problem with the application of Newton method to the resolution of the KKT conditions is the nonnegativity constraints on x and s, which cannot directly be taken into account via the mapping F. One way of incorporating these constraints is to use a *barrier* term, as described in the next paragraph.

2.4 Barrier function

A barrier function $\phi : \mathbb{R}^+ \mapsto \mathbb{R}$ is simply a differentiable function such that $\lim_{x\to 0^+} \phi(x) = +\infty$. Using such a barrier, it is possible to deduce from an inequality-constrained problem a parameterized family of unconstrained problems in the following way

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{s.t.} \quad g_i(x) \geq 0 \ \forall i \tag{G}$$

$$\to \quad \min_{x\in\mathbb{R}^n} f(x) + \mu \sum_i \phi(g_i(x)) \ , \tag{G$_\mu$}$$

where $\mu \in \mathbb{R}^+$. The purpose of the added barrier term is to drive the iterates generated by an unconstrained optimization method away from the infeasible zone (where one or more g_i's are negative). Of course, we should not expect the optimal solutions to (G$_\mu$) to be equal to those of (G). In fact each value of μ gives rise to a different problem (G$_\mu$) with its own optimal solutions.

However, if we solve a sequence of problems (G$_\mu$) with μ decreasing to zero, we might expect the sequence of optimal solutions we obtain to converge to the optimum of the original problem (G), since the impact of the barrier term is less and less significant compared to

[6] Computation of Δx_k is usually done with the linear system $J(x_k)\Delta x_k = -F(x_k)$ rather than computing explicitly $J(x_k)$'s inverse.

the real objective function. The advantage of this procedure is that each optimal solution in the sequence will satisfy the strict inequality constraints $g_i(x) > 0$, leading to a feasible optimal solution to (G)[7].

The application of this technique to linear programming will lead to a fundamental notion in interior-point methods: the *central path*.

2.5 The central path

Interior-point researchers use the following barrier function, called *logarithmic barrier*:

$$\phi(x) = -\log(x) .$$

Using ϕ, let us apply a barrier term to the linear programming problem (P)

$$\min_{x\in\mathbb{R}^n} c^T x - \mu \sum_i \log(x_i) \quad \text{s.t.} \quad \begin{cases} Ax = b \\ x > 0 \end{cases} \tag{P_μ}$$

and to its dual (D) (since it is a maximization problem, we have to subtract the barrier term)

$$\max_{y\in\mathbb{R}^m} b^T y + \mu \sum_i \log(s_i) \quad \text{s.t.} \quad \begin{cases} A^T y + s = c \\ s > 0 \text{ and } y \text{ free} \end{cases} . \tag{D_μ}$$

It is possible to prove that both these problems have unique optimal solutions x_μ and (y_μ, s_μ) for all $\mu > 0$ if and only if both $\mathcal{P}^+$ and $\mathcal{D}^+$ are nonempty[8]. In that case, we call these sets of optimal solutions $\{x_\mu \mid \mu > 0\} \subset \mathcal{P}^+$ and $\{(y_\mu, s_\mu) \mid \mu > 0\} \subset \mathcal{D}^+$ respectively the primal and dual central paths. These parametric curves have the following properties:

- ◇ The primal (resp. dual) objective value $c^T x$ (resp. $b^T y$) is monotonically decreasing (resp. increasing) along the primal (resp. dual) central path when $\mu \to 0$.
- ◇ The duality gap $c^T x_\mu - b^T y_\mu$ for the primal-dual solution (x_μ, y_μ, s_μ) is equal to $n\mu$. For this reason, μ will be called the *duality measure*. When a point (x, y, s) does not lie exactly on the central path, we can compute its *estimated* duality measure using $\mu = (c^T x - b^T y)/n$.
- ◇ The limit points $x_* = \lim_{\mu\to 0} x_\mu$ and $(y_*, s_*) = \lim_{\mu\to 0}(y_\mu, s_\mu)$ exist and hence are optimal solutions to problems (P) and (D) (because we have $c^T x_* - b^T y_* = 0$). Moreover, we have that $x_* + s_* > 0$, i.e. this optimal pair is strictly complementary[9].

[7] The notion of barrier function was first investigated in [Fri55, FM68].
[8] This condition is known as the *interior-point condition*.
[9] For optimal solutions (x, s) we always have $x_i s_i = 0$, i.e. at least one of x_i and s_i is zero. In the case of a strictly complementary solution, *exactly* one of x_i and s_i is zero.

2.6 Link between central path and KKT equations

To conclude this section we establish a link between the central path and the KKT equations. Applying the general KKT conditions to either problem (P_μ) or (D_μ) we find the following necessary and sufficient conditions

$$\left\{\begin{array}{rcl} Ax & = & b \\ A^T y + s & = & c \\ x_i s_i & = & \mu \, \forall i \\ x \text{ and } s & > & 0 \end{array}\right. \quad \Leftrightarrow \quad \left\{\begin{array}{rcl} x & \in & \mathcal{P}^+ \\ (y,s) & \in & \mathcal{D}^+ \\ x_i s_i & = & \mu \, \forall i \end{array}\right. . \qquad (\mathrm{KKT}_\mu)$$

This system is very similar to the original KKT system, the only difference being the right-hand side of the third condition and the strict inequalities. This means in fact that the points on the central path obey to a slightly perturbed version of the optimality KKT conditions for (P) and (D).

We now have all the tools we need to give a description of interior-point methods for linear programming.

3 Interior-point algorithms

Since Karmarkar's breakthrough, many different interior-point methods have been developed. It is important to note there exists in fact a whole collection of methods, sharing the same basic principles but whose individual characteristics may vary a lot.

Among the criteria that are commonly used to classify the methods, we have

- **Iterate space**. A method is said to be *primal*, *dual* or *primal-dual* when its iterates belong respectively to the primal space, the dual space or the Cartesian product of these spaces.

- **Type of iterate**. A method is said to be *feasible* when its iterates are feasible, i.e. satisfy both the equality and nonnegativity constraints. In the case of an *infeasible method*, the iterates need not satisfy the equality constraints, but are still required to satisfy the nonnegativity conditions.

- **Type of algorithm**. This is the main difference between the methods. Although the denominations are not yet fully standardized, we will distinguish *path-following* algorithms, *affine-scaling* algorithms and *potential reduction* algorithms.

- **Type of step**. In order to preserve their polynomial complexity, some algorithms are obliged to take very small steps at each iteration, leading to a high total number of iterations when applied to practical problems[10]. These methods are called *short-step* methods and are mainly of theoretical interest. Therefore *long-step* methods, which are allowed to take much longer steps, have been developed and are the only methods used in practice.

[10] Please note that this is not in contradiction with the fact that this number of iterations is polynomially bounded by the size of the problem. This may simply mean that the polynomial coefficients are large.

It is not our purpose to give an exhaustive list of all the methods that have been developed up to now, but rather to present some representative algorithms, highlighting their underlying principles.

3.1 Path-following algorithms

We start with the most elegant category of methods, the path-following algorithms. As suggested by their denomination and the previous section, the main idea behind these methods is to follow the central path up to its limit point. One could imagine the following naive conceptual algorithm (at this point, we want to keep generality and do not specify whether our method is primal, dual or primal-dual)

> **Given** an initial iterate v_0 and a sequence of duality measures monotonically decreasing to zero: $\mu_1 > \mu_2 > \mu_3 > \cdots > 0$ and $\lim_{k\to 0} \mu_k = 0$.
>
> **Repeat** for $k = 0, 1, 2, \ldots$
>
> Using v_k as starting point, compute v_{k+1}, the point on the central path with a duality measure equal to μ_{k+1}.
>
> **End**

It is clear from this scheme that v_k will tend to the limit point of the central path, which is an optimal solution to our problem.

However, the determination of a point on the central path requires to solve a minimization problem like (P_μ) or the (KKT_μ) conditions, which potentially implies a lot of computational work. This is why path-following interior-point methods will only try to compute points that are *approximately* on the central path, hopefully with much less computational work, and will thus only *loosely* follow the central path. Our conceptual algorithm becomes

> **Given** an initial iterate v_0 and a sequence of duality measures monotonically decreasing to zero: $\mu_1 > \mu_2 > \mu_3 > \cdots > 0$ and $\lim_{k\to 0} \mu_k = 0$.
>
> **Repeat** for $k = 0, 1, 2, \ldots$
>
> Using v_k as starting point, compute v_{k+1}, an approximation of the point on the central path with a duality measure equal to μ_{k+1}.
>
> **End**

The main task in proving the convergence and complexity of these methods will be to assess how well we approximate our targets on the central path (i.e. how close to the central path we stay).

3.1.1 Short-step primal-dual path-following algorithm

This specific algorithm is a primal-dual feasible method, which means that all the iterates will lie in $\mathcal{P}^+ \times \mathcal{D}^+$. Let (x_k, y_k, s_k) be the current iterate with duality measure μ_k. We also suppose that this iterate is *close* to the point $(x_{\mu_k}, y_{\mu_k}, s_{\mu_k})$ on the central path. To compute the next iterate, we target $(x_{\mu_{k+1}}, y_{\mu_{k+1}}, s_{\mu_{k+1}})$, a point on the central path with a smaller duality measure μ_{k+1} (thus closer to the optimal limit point). The main two characteristics of the short-step method are

◇ The duality measure of the point we target is defined by $\mu_{k+1} = \sigma\mu_k$ where σ is a constant between 0 and 1.

◇ The next iterate will be computed applying *one single* Newton step to the perturbed primal-dual conditions (KKT_μ) with $\mu = \sigma\mu_k$ defining our target on the central path[11]

$$\left\{ \begin{array}{rcl} Ax & = & b \\ A^T y + s & = & c \\ x_i s_i & = & \sigma\mu_k \ \forall i \end{array} \right. .$$

Formally, the Newton step we take is defined by the following linear system

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S_k & 0 & X_k \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta y_k \\ \Delta s_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -X_k S_k e + \sigma\mu_k e \end{pmatrix} \tag{3.1}$$

where e stands for the all-one vector and X_k and S_k are diagonal matrices made up with vectors x_k and s_k (these notations are standard in the field of interior-point methods).

This leads to the following algorithm

Given an initial iterate $(x_0, y_0, s_0) \in \mathcal{P}^+ \times \mathcal{D}^+$ with duality measure μ_0 and a constant $0 < \sigma < 1$.

Repeat for $k = 0, 1, 2, \ldots$

Compute the Newton step $(\Delta x_k, \Delta y_k, \Delta s_k)$ using the linear system (3.1).

Let $(x_{k+1}, y_{k+1}, s_{k+1}) = (x_k, y_k, s_k) + (\Delta x_k, \Delta y_k, \Delta s_k)$ and $\mu_{k+1} = \sigma\mu_k$.

End

We now sketch a proof of the correctness of this algorithm. For our path-following strategy to work, we have to ensure that our iterates (x_k, y_k, s_k) stay close to the points $(x_{\mu_k}, y_{\mu_k}, s_{\mu_k})$ on the central path, which guide us to an optimal solution. For this purpose we define a quantity that measures the proximity between a strictly feasible iterate $(x, y, s) \in \mathcal{P}^+ \times \mathcal{D}^+$ and the central point (x_μ, y_μ, s_μ). Since the main property of this central point is $x_i s_i = \mu \ \forall i$, which is equivalent to[12] $xs = \mu e$, the following measure (see e.g. [Wri97])

$$\delta(x, s, \mu) = \frac{1}{\mu} \|xs - \mu e\| = \left\| \frac{xs}{\mu} - e \right\|$$

seems adequate: it is zero if and only if (x, y, s) is equal to (x_μ, y_μ, s_μ) and increases as we move away from this central point. It is also interesting to note that the size of a neighbourhood defined by $\delta(x, s, \mu) < R$ decreases with μ, because of the leading term $\frac{1}{\mu}$.

Another possibility of proximity measure with the same properties is

$$\delta(x, s, \mu) = \frac{1}{2} \left\| \sqrt{\frac{xs}{\mu}} - \sqrt{\frac{\mu}{xs}} \right\|$$

[11] Note that we have to ignore the nonnegativity conditions for the moment.

[12] xs denotes here the componentwise product of vectors x and s.

where the square roots are taken componentwise (see [CRV97]).

Our proof has the following steps

1. **Strict Feasibility**. Prove that strict feasibility is conserved after the Newton step: if $(x_k, y_k, s_k) \in \mathcal{P}^+ \times \mathcal{D}^+$, we have $(x_{k+1}, y_{k+1}, s_{k+1}) \in \mathcal{P}^+ \times \mathcal{D}^+$. We have to be especially careful with the strict nonnegativity constraints, since they are not taken into account by the Newton method.

2. **Duality measure**. Prove that the target duality measure is attained after the Newton step: if (x_k, y_k, s_k) has a duality measure equal to μ_k, the next iterate $(x_{k+1}, y_{k+1}, s_{k+1})$ has a duality measure equal to $\sigma\mu_k$

3. **Proximity**. Prove that proximity to the central path targets is conserved: there is a constant τ such that if $\delta(x_k, s_k, \mu_k) < \tau$, we have after the Newton step that $\delta(x_{k+1}, s_{k+1}, \mu_{k+1}) < \tau$.

Adding the additional initial assumption that $\delta(x_0, s_0, \mu_0) < \tau$, this is enough to prove that the sequence of iterates is converging to the limit point of the central path, which is a (strictly complementary) optimal solution. The last delicate question is to choose a suitable combination of constants σ and τ that allows us to prove the three statements above. For the first duality measure we presented the following values are acceptable (see [Wri97])

$$\sigma = 1 - \frac{0.4}{\sqrt{n}} \text{ and } \tau = 0.4 ,$$

where n stands for the size of vectors x and s as usual, while for the second measure we may choose (see [CRV97])

$$\sigma = 1 - \frac{1}{2\sqrt{n}} \text{ and } \tau = \frac{1}{\sqrt{2}} .$$

To conclude this description, we specify how the algorithm terminates. Given an accuracy parameter ε, we stop our computations when the duality gap falls below ε, which happens when $n\mu_k < \varepsilon$. This guarantees that $c^T x$ and $b^T y$ approximate the true optimal objective value with an error smaller than ε. We now state this algorithm in its final form:

> **Given** an initial iterate $(x_0, y_0, s_0) \in \mathcal{P}^+ \times \mathcal{D}^+$ with duality measure μ_0, an accuracy parameter ε and suitable constants $0 < \sigma < 1$ and τ such that $\delta(x_0, y_0, s_0) < \tau$.
>
> **Repeat** for $k = 0, 1, 2, \ldots$
>
> Compute the Newton step $(\Delta x_k, \Delta y_k, \Delta s_k)$ using the linear system (3.1).
>
> Let $(x_{k+1}, y_{k+1}, s_{k+1}) = (x_k, y_k, s_k) + (\Delta x_k, \Delta y_k, \Delta s_k)$ and $\mu_{k+1} = \sigma\mu_k$.
>
> **Until** $n\mu_{k+1} < \varepsilon$

Moreover, it is also possible to prove that in both cases, the solution with ε accuracy will be reached after a number of iterations N such that

$$N = O\left(\sqrt{n} \log \frac{n\mu_0}{\varepsilon}\right) . \tag{3.2}$$

This polynomial complexity bound varying like the square root of the problem size is the best attained so far for linear programming.

However, it is important to note that values of σ presented above will always be in practice nearly equal to one, which means that the duality measures will decrease very slowly. Although its complexity is polynomial, this method requires a large number of iterations and is not very efficient from a practical point of view.

3.1.2 Dual short-step path-following methods

This second short-step method is very similar to the previous one but its iterates lie in the dual space $\mathcal{D}^+$. We keep the general principle of following the dual central path and targeting points (y_{μ_k}, s_{μ_k}) on it but we have to make the following adjustments[13]

- ⋄ We cannot deduce the Newton step from the (KKT_μ) conditions any more, since they involve both primal and dual variables. We apply instead a single minimizing Newton step to the (D_μ) barrier problem, which gives the following linear system

$$\begin{pmatrix} A^T & I \\ AS_k^{-2}A^T & 0 \end{pmatrix} \begin{pmatrix} \Delta y_k \\ \Delta s_k \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{b}{\sigma\mu_k} - AS_k^{-1}e \end{pmatrix} . \tag{3.3}$$

- ⋄ We have to modify our measure of proximity: we now define $\delta(s, \mu)$ with

$$\delta(s,\mu) = \min_x \{\delta(x,s,\mu) \mid Ax = b\} = \frac{1}{\mu} \min_x \{\|xs - \mu e\| \mid Ax = b\}$$

  (we have that this measure is zero if and only if $s = s_\mu$).

Our algorithm simply becomes

> **Given** an initial iterate $(y_0, s_0) \in \mathcal{D}^+$ with duality measure μ_0, an accuracy parameter ε and suitable constants $0 < \sigma < 1$ and τ such that $\delta(y_0, s_0) < \tau$.
>
> **Repeat** for $k = 0, 1, 2, \ldots$
>
> Compute the Newton step $(\Delta y_k, \Delta s_k)$ using the linear system (3.3).
>
> Let $(y_{k+1}, s_{k+1}) = (y_k, s_k) + (\Delta y_k, \Delta s_k)$ and $\mu_{k+1} = \sigma\mu_k$.
>
> **Until** $n\mu_{k+1} < \varepsilon$

In this case we may for example choose

$$\sigma = 1 - \frac{1}{3\sqrt{n}} \text{ and } \tau = \frac{1}{\sqrt{2}} ,$$

which leads to the same complexity bound (3.2) for the total number of iterations.

[13] It is of course also possible to design a primal short-step path-following method in a completely similar fashion.

3.1.3 Primal-dual long-step path-following methods

The long-step primal-dual method we are going to describe now is an attempt to overcome the main limitation of the short-step methods: their very small step size. As presented above, the fundamental reason for this slow progress is the value of σ that has to be chosen nearly equal to one in order to prove the polynomial complexity of the method.

A simple idea to accelerate the method would simply be to decrease the duality measure more aggressively, i.e. still using $\mu_{k+1} = \sigma\mu_k$ but with a lower σ. However, this apparently small change breaks down the good properties we were able to prove for the short-step algorithms. Indeed, if our target on the central path is too far from our current iterate, we may have that

- ◇ The Newton step computed by (3.1) is no longer feasible. The reason for that is easy to understand. The Newton method is asked to solve the (KKT_μ) system, which is made of two linear equations and one mildly nonlinear equation. Because of this third equation, the linear system we solve is only an approximation of the real set of equations, and the further we are from the solution we target, the less accurate this approximation is. When our target is located too far, the linear approximation becomes so bad that barrier term doesn't play its role and the Newton step jumps out of the feasible region by violating the nonnegativity constraints[14] $x > 0$ and $s > 0$.

  Since the iterates of an interior-point method must always satisfy the strict nonnegativity conditions, we have to take a so-called *damped* Newton step, i.e. reduce it with a factor $\alpha_k < 1$ in order to make it stay within the strictly feasible region $\mathcal{P}^+ \times \mathcal{D}^+$:

  $$(x_{k+1}, y_{k+1}, s_{k+1}) = (x_k, y_k, s_k) + \alpha_k(\Delta x_k, \Delta y_k, \Delta s_k) \ .$$

- ◇ This damping of the Newton step cancels the property that the duality measure we target is attained. It is indeed possible to show that the duality measure after a damped Newton step becomes $(1 - \alpha_k(1 - \sigma))\mu_k$, which varies linearly between μ_k and $\sigma\mu_k$ when α decreases from 1 to 0.

  There is unfortunately no way to circumvent this drawback, and we have to accept that our iterates never exactly achieve the targeted duality measures, unless a full Newton step is taken.

- ◇ We cannot guarantee that a single Newton step will keep the proximity to the central path in the sense of $\delta(x, s, \mu) < \tau$, for the same reasons as above (nonlinearity). In the long-step strategy we describe, we take several Newton steps with the same target duality measure until proximity to the central path is restored. Then we may choose another target and decrease μ.

Our long-step method may be described in the following way:

> **Given** an initial iterate $(x_0, y_0, s_0) \in \mathcal{P}^+ \times \mathcal{D}^+$, an initial duality measure μ_0, an accuracy parameter ε and suitable constants $0 < \sigma < 1$ and τ such that $\delta(x_0, y_0, s_0) < \tau$.

[14] Note that since the first two conditions $Ax = b$ and $A^T y + s = c$ are linear, they are always fulfilled after the Newton step.

Repeat for $k = 0, 1, 2, \ldots$

Compute the Newton step $(\Delta x_k, \Delta y_k, \Delta s_k)$ using the linear system (3.1).

Let $(x_{k+1}, y_{k+1}, s_{k+1}) = (x_k, y_k, s_k) + \alpha_k(\Delta x_k, \Delta y_k, \Delta s_k)$ with a step length α_k chosen such that $(x_{k+1}, y_{k+1}, s_{k+1}) \in \mathcal{P}^+ \times \mathcal{D}^+$.

If $\delta(x_{k+1}, s_{k+1}, \sigma\mu_k) < \tau$ **Then** let $\mu_{k+1} = \sigma\mu_k$ **Else** let $\mu_{k+1} = \mu_k$.

Until $n\mu_{k+1} < \varepsilon$

As opposed to the complexity analysis of the short-step method, we may choose here whatever value we want for the constant σ, in particular values much smaller than 1. It is the choice of τ and α_k that makes the method polynomial. The main task is here to analyze the number of iterations that is needed to restore proximity to the central path. Taking for σ a constant independent of n (like .5, .1 or .01), it is possible to prove that suitable choices of τ and α_k lead to the following number of iterations

$$N = O\left(n \log \frac{n\mu_0}{\varepsilon}\right) .$$

Let us point out an odd fact: although this method takes longer steps and is practically more efficient than the short-step methods, its theoretical complexity is worse than the short-step complexity (3.2).

3.2 Affine-scaling algorithms

The intensive stream of research on the topic of interior-point methods for linear programming was triggered by Karmarkar's seminal article [Kar84]. His method used projective transformations and was not described in terms of central path or Newton method. Later, researchers simplified this algorithm, removing the need for projective transformations, and obtained a class of methods called affine-scaling algorithms. It was later discovered that these methods had been previously proposed by Dikin in Russia, 17 years before Karmarkar [Dik67].

Affine-scaling algorithms do not explicitly follow the central path and do not even refer to it. The basic idea underlying these methods is the following: consider for example the primal problem (P)

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{s.t.} \quad \begin{cases} Ax = b \\ x \geq 0 \end{cases} . \tag{P}$$

This problem is hard to solve because of the nonnegativity constraints, which give the feasible region a polyhedral shape. Let us consider the current iterate x_k and replace the polyhedral feasible region by an inscribed ellipsoid centered at x_k. The idea is to minimize the objective on this ellipsoid, which should be easier than on a polyhedron, and take this minimum as next iterate.

How do we construct an ellipsoid that is centered at x_k and inscribed into the feasible region ? Consider a positive diagonal matrix D. It is easy to show that problem ($\mathrm{P_D}$)

$$\min_{w \in \mathbb{R}^n} (Dc)^T w \quad \text{s.t.} \quad \begin{cases} ADw = b \\ w \geq 0 \end{cases} \tag{$\mathrm{P_D}$}$$

is equivalent to (P), the x variable being simply scaled by $x = Dw$ (this scaling operation is responsible for the denomination of the method). Choosing a special diagonal matrix $D = X_k$, which maps the current iterate x_k to e, we obtain the following problem

$$\min_{w \in \mathbb{R}^n} (X_k c)^T w \quad \text{s.t.} \quad \begin{cases} AX_k w = b \\ w \geq 0 \end{cases} .$$

We are now able to restrict the feasible region defined by $w \geq 0$ to a unit ball centered at e, according to the inclusion $\{w \mid w \geq 0\} \subset \{w \mid \|w - e\| \leq 1\}$. Our problem becomes

$$\min_{w \in \mathbb{R}^n} (X_k c)^T w \quad \text{s.t.} \quad \begin{cases} AX_k w = b \\ \|w - e\| \leq 1 \end{cases} ,$$

i.e. the minimization of a linear objective over the intersection of a unit ball and an affine subspace, whose solution can be easily computed analytically via a linear system. Back in the original space, this is equivalent to

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{s.t.} \quad \begin{cases} Ax = b \\ \left\|X_k^{-1} x - e\right\| \leq 1 \end{cases} ,$$

whose feasible region is an ellipsoid centered at x_k. This ellipsoid is called the *Dikin* ellipsoid and lies entirely inside $\mathcal{P}$. The minimum over this ellipsoid is given by $x_k + \Delta x_k$, where[15]

$$\Delta x_k = -\frac{X_k P_{AX_k} X_k c}{\|P_{AX_k} X_k c\|} . \tag{3.4}$$

Because our ellipsoid lies entirely within the feasible region, the step Δx_k is feasible and the next iterate $x_k + \Delta x_k$ is expected to be closer to the optimal solution than x_k.

3.2.1 Short- and long-step primal affine-scaling algorithms

Introducing a constant μ to reduce the step size, we may state our algorithm as

> **Given** an initial iterate $x_0 \in \mathcal{P}^+$ and a constant $0 < \mu < 1$.
>
> **Repeat** for $k = 0, 1, 2, \ldots$
>
> Compute the affine scaling step Δ_k with (3.4) and let $x_{k+1} = x_k + \mu \Delta_k$.
>
> **End**

This scheme is known as the short-step primal affine-scaling algorithm. Convergence to a primal solution has been proved for $\mu = \frac{1}{8}$, but we still do not know whether this method has polynomial complexity[16]. It is of course possible to design a dual and even a primal-dual variant of this method (all we have to do is to define the corresponding Dikin ellipsoids).

---

[15] P_Q denotes the projection matrix onto $\mathrm{Ker}(Q)$, which can be written as $P_Q = I - Q^T(QQ^T)^{-1}Q$ when Q has maximal rank.

[16] When certain nondegeneracy conditions hold, convergence has been proved for $0 < \mu < 1$.

It is also possible to make the algorithm more efficient by taking longer steps, i.e. moving outside of the Dikin ellipsoid. Keeping the same direction as for the short-step method, the maximum step we can take without leaving the primal feasible region is given by

$$\Delta x_k = -\frac{X_k P_{AX_k} X_k c}{\max\left[P_{AX_k} X_k c\right]} , \tag{3.5}$$

which leads to the following algorithm:

Given an initial iterate x_0 and a constant $0 < \lambda < 1$.

Repeat for $k = 0, 1, 2, \ldots$

Compute the affine scaling step Δ_k with (3.5) and let $x_{k+1} = x_k + \lambda\Delta_k$.

End

The constant λ decides which fraction of the way to the boundary of the feasible region we move[17]. Global convergence has been proved when $0 < \lambda \leq 2/3$ but a surprising counterexample has been found with $\lambda = 0.999$ (see [Mas93]). Finally, as for the short-step method, we do not know whether this method has polynomial complexity.

3.2.2 Link with path-following algorithms

There is an interesting and unexpected link between affine-scaling methods and path-following algorithms. Taking for example the definition (3.3) of the dual Newton step in the path-following framework and letting σ tend to zero, i.e. letting the target duality measure tend to zero, we find that the resulting limit direction is exactly equal to the dual affine-scaling direction ! This surprising fact, which is also valid for their primal counterparts, gives us some insight about both methods:

- ◇ The affine-scaling method can be seen as a kind of Newton method that is targeting the limit point of the central path, i.e. that tries to jump directly to an optimal solution without following the central path.

- ◇ It is possible to decompose the dual Newton step into two parts:

$$\Delta x_k = \frac{1}{\sigma\mu_k}\Delta^a x_k + \Delta^c x_k ,$$

where

$$\begin{pmatrix} A^T & I \\ AS_k^{-2}A^T & 0 \end{pmatrix}\begin{pmatrix} \Delta^a y_k \\ \Delta^a s_k \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix} \text{ and } \begin{pmatrix} A^T & I \\ AS_k^{-2}A^T & 0 \end{pmatrix}\begin{pmatrix} \Delta^c y_k \\ \Delta^c s_k \end{pmatrix} = \begin{pmatrix} 0 \\ -AS_k^{-1}e \end{pmatrix} .$$

  - $\Delta^a x_k$ is called the affine-scaling component. It has the same direction as the affine-scaling method and is only seeking optimality.
  - $\Delta^c x_k$ is called the centering component. It is targeting a point on the central path with the same duality measure as the current iterate, i.e. only tries to improve proximity to the central path.

It is possible to show that most interior-point methods follow in fact directions that are combinations of these two basic directions.

[17] This constant has to be strictly less than 1 since we want to stay in the interior of the feasible region.

3.3 Potential reduction algorithms

Instead of targeting a decreasing sequence of duality measures, the method of Karmarkar made use of a potential function to monitor the progress of its iterates. A potential function is a way to measure the worth of an iterate. Its main two properties are the following:

- ◇ It should tend to $-\infty$ if and only if the iterates tend to optimality.
- ◇ It should tend to $+\infty$ when the iterates tend to the boundary of the feasible region without tending to an optimal solution[18].

The main goal of a potential reduction algorithm is simply to reduce the potential function by a fixed amount δ at each step, hence its name. Convergence follows directly from the first property above.

3.3.1 Primal-dual potential reduction algorithm

We are going to describe the application of this strategy in the primal-dual case. The Tanabe-Todd-Ye primal-dual potential function is defined on the strictly feasible primal-dual space $\mathcal{P}^+ \times \mathcal{D}^+$ by

$$\Phi_\rho(x, s) = \rho \log x^T s - \sum_i \log x_i s_i \ ,$$

where ρ is a constant required to be greater than n. We may rewrite it as

$$\Phi_\rho(x, s) = (\rho - n) \log x^T s - \sum_i \log \frac{x_i s_i}{x^T s/n} + n \log n$$

and note the following

- ◇ The first term makes the potential tend to $-\infty$ when (x, s) tends to optimality, since we have then the duality gap $x^T s$ tending to 0.
- ◇ The second term measures *centrality* of the iterate. A perfectly centered iterate will have all its products $x_i s_i$ equal to their average value $x^T s/n$, making the second term equal to zero. As soon these products become different, this term increases, and tends to $+\infty$ if one of the products $x_i s_i$ tends to zero without $x^T s$ tending also to zero (which means exactly that we approach the boundary of the feasible region without tending to an optimal solution).

The search direction for this method is not new: it is the same as for the path-following algorithm, defined with a target duality measure $n\mu_k/\rho$ (i.e. with $\sigma = n/\rho$). However, in this case, μ_k will not follow a predefined decreasing sequence, but will have to be recomputed after each step (since this algorithm cannot guarantee that the duality measure targeted by the Newton step will be attained). The algorithm proceeds as follows:

[18] We cannot of course simply prevent the method from approaching the boundary of the feasible region, since our optimal solution lies on it.

Given an initial iterate $(x_0, y_0, s_0) \in \mathcal{P}^+ \times \mathcal{D}^+$ with duality measure μ_0 and a constant $\rho > n$. Define $\sigma = n/\rho$.

Repeat for $k = 0, 1, 2, \ldots$

Compute the Newton step $(\Delta x_k, \Delta y_k, \Delta s_k)$ using the linear system (3.1).

Let $(x_{k+1}, y_{k+1}, s_{k+1}) = (x_k, y_k, s_k) + \alpha_k(\Delta x_k, \Delta y_k, \Delta s_k)$ where α_k is defined by

$$\begin{aligned} \alpha_k \quad = \quad & \arg\min_{\alpha} \Phi_\rho(x_k + \alpha \Delta x_k, s_k + \alpha \Delta s_k) \\ & \text{s.t.} \quad (x_k, y_k, s_k) + \alpha(\Delta x_k, \Delta y_k, \Delta s_k) \in \mathcal{P}^+ \times \mathcal{D}^+ \, . \end{aligned}$$

Evaluate μ_{k+1} with $(x_{k+1}^T s_{k+1})/n$.

Until $n\mu_{k+1} < \varepsilon$

The principle of this method is thus to minimize the potential function along the search direction at each iteration. The main task in analyzing the complexity of this method is to prove that this step will provide at least a fixed reduction of Φ_ρ at each iteration. Using $\rho = n + \sqrt{n}$, it is possible to prove that $\Phi_\rho(x_{k+1}, s_{k+1}) \leq \Phi_\rho(x_k, s_k) - \delta$ with $\delta = 0.16$ (see e.g. [Ans96]), leading to a total number of iterations equal to

$$N = O\left(\sqrt{n} \log \frac{n\mu_0}{\varepsilon}\right) \, ,$$

matching the best complexity results for the path-following methods.

It is in general too costly for a practical algorithm to minimize exactly the potential function along the search direction, since Φ_ρ is a highly nonlinear function. We may use instead one the following strategies

- ◇ Define a quadratic approximation of Φ_ρ along the search direction and take its minimizer as next iterate.
- ◇ Take a fixed percentage (e.g. 95 %) of the maximum step along the search direction staying inside of the feasible region.

We note however that polynomial complexity is no more guaranteed in these cases.

4 Enhancements

In the following, we present various enhancements that are needed to make the theoretical methods of the previous section work in practice.

4.1 Infeasible algorithms

All the algorithms we have described up to now are feasible methods, which means they need a strictly feasible iterate as starting point. However, such a point is not always available:

- ◇ For some problems, a *natural* strictly feasible point is not directly available and finding one may be as difficult as solving the whole linear program.

- ◇ Some problems have no strictly feasible points although they are perfectly valid and have finite optimal solutions. This situation happens in fact if and only if the optimal solution set is not bounded[19].

We can think of two different strategies to handle such cases: embed the problem into a larger one that admits a strictly feasible starting point (this will be developed in the next paragraph) or modify the algorithm to make it work with infeasible iterates. We are now going to give an overview of this second strategy.

We recall that the iterates of an infeasible method do not satisfy the equality constraints $Ax = b$ and $A^T y + s = c$ but are required to be nonnegative, i.e. $x > 0$ and $s > 0$. The main idea is simply to ask the Newton method to make the iterates feasible. This amounts to a simple modification of the linear system (3.1), which becomes

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S_k & 0 & X_k \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta y_k \\ \Delta s_k \end{pmatrix} = \begin{pmatrix} c - A^T y_k - s_k \\ b - A x_k \\ -X_k S_k e + \sigma \mu_k e \end{pmatrix} . \tag{4.1}$$

The only difference with the feasible system is the right-hand side vector, which now incorporates the primal and dual residuals $b - Ax_k$ and $c - (A^T y_k + s_k)$. Newton steps will try to reduce both the duality measure and the iterate infeasibility at the same time.

Infeasible variants of both path-following and potential reduction methods have been developed using this search direction. Without going into the details, let us point out that an additional constraint on the step has to be enforced to ensure that infeasibility is reduced at least at the same pace as the duality measure (to avoid ending with an "optimal" solution that would be infeasible). The complexity results for these methods are the same as those of their feasible counterparts, although the analysis is generally much more involved.

4.2 Homogeneous self-dual embedding

As mentioned in the previous subsection, another way to handle infeasibility is to embed our problem into a larger linear program that admits a known feasible starting point. We choose a starting iterate (x_0, y_0, s_0) such that $x_0 > 0$ and $s_0 > 0$ and define the following quantities

$$\hat{b} = b - Ax_0 \qquad \hat{c} = c - A^T y_0 - s_0 \qquad \hat{g} = b^T y_0 - c^T x_0 - 1 \qquad \hat{h} = x_0^T s_0 + 1 .$$

We consider the following problem

$$\begin{array}{lrrrrrrcr} \min & & & & \hat{h}\theta & & & & \\ \text{s.t.} & & Ax & -b\tau & +\hat{b}\theta & & & = & 0 \\ & -A^T y & & +c\tau & -\hat{c}\theta & -s & & = & 0 \\ & b^T y & -c^T x & & -\hat{g}\theta & & -\kappa & = & 0 \\ & -\hat{b}^T y & +\hat{c}^T x & +\hat{g}\tau & & & & = & -\hat{h} \\ & & x \geq 0 & \tau \geq 0 & & s \geq 0 & \kappa \geq 0 & & . \end{array} \tag{HSD}$$

[19] This is the case for example when a variable that is not upper bounded by the constraints is not present in the objective.

It is easy to see that this problem admits $(x, y, s, \tau, \kappa, \theta) = (x_0, y_0, s_0, 1, 1, 1)$ as a strictly feasible starting point. Without going into too many details, we give a brief description of the new variables involved in (HSD): τ is a homogenizing variable, θ is measuring infeasibility and κ refers to the duality gap in the original problem. We also point out that the first two equalities correspond to the feasibility constraints $Ax = b$ and $A^T y + s = c$.

This program has the following interesting properties:

- ◇ This program is homogeneous, i.e. its right-hand side is the zero vector (except for the last equality that is a homogenizing constraint).
- ◇ This program is self-dual, i.e. its dual is identical to itself (this is due to the fact that the coefficient matrix is skew-symmetric).
- ◇ The optimal value of (HSD) is 0 (i.e. $\theta_* = 0$).
- ◇ Given a strictly complementary solution $(x_*, y_*, s_*, \tau_*, \kappa_*, 0)$ to (HSD) we have either $\tau_* > 0$ or $\kappa_* > 0$.
- ◇ If $\tau_* > 0$ then $(x_*/\tau_*, y_*/\tau_*, s_*/\tau_*)$ is an optimal solution to our original problem.
- ◇ If $\kappa_* > 0$ then our original problem has no finite optimal solution. Moreover, we have in this case $b^T y_* - c^T x_* > 0$ and
  - When $b^T y_* > 0$, problem (P) is infeasible.
  - When $-c^T x_* > 0$, problem (D) is infeasible.

Since we know a strictly feasible starting point, we can apply a feasible path-following method to this problem that will converge to an optimal strictly complementary solution. Using the above-mentioned properties, it is then possible to compute an optimal solution to our original problem or detect its infeasibility.

This homogeneous self-dual program has roughly twice the size of our original linear program, which may be seen as a drawback. However, it is possible to take advantage of the self-duality property and use some algorithmic devices to solve this problem at nearly the same computational cost as the original program.

4.3 Theory vs. implemented algorithms

We have already mentioned that a polynomial complexity result is not necessarily a guarantee of good practical behaviour. Short-step methods are definitely too slow because of the tiny reduction of the duality measure they allow. Long-step methods perform better but are still too slow. This is why practitioners have implemented various tricks to accelerate their practical behaviour. It is important to note that the complexity results we've mentioned so far do not apply to these modified methods, since they do not strictly follow the theory.

The infeasible primal-dual long-step path-following algorithm is by far the most commonly implemented interior-point method. The following tricks are usually added:

- ⋄ The theoretical long-step method takes several Newton steps targeting the same duality measure until proximity to the central path is restored. Practical algorithms ignore this and take only a single Newton step, like short-step methods.
- ⋄ Instead of choosing the step length recommended by the theory, practical implementations usually take a very large fraction of the maximum step that stays within the feasible region (common values are 99.5 % or 99.9 %). This modification works especially well with primal-dual methods.
- ⋄ The primal and dual steps are taken with different step lengths, i.e. we take

  $$x_{k+1} = x_k + \alpha^{\mathrm{P}} \Delta x_k \text{ and } (y_{k+1}, s_{k+1}) = (y_k, s_k) + \alpha^{\mathrm{D}} (\Delta y_k, \Delta s_k) \ .$$

  These steps are chosen according to the previous trick, for example with $(\alpha^{\mathrm{P}}, \alpha^{\mathrm{D}}) = 0.995\,(\alpha^{\mathrm{P}}_{\max}, \alpha^{\mathrm{D}}_{\max})$. This modification alone is responsible for a substantial decrease of the total number of iterations, but is not theoretically justified.

4.4 The Mehrotra predictor-corrector algorithm

The description of the methods from the previous section has underlined the fact that the constant σ, defining the target duality measure $\sigma\mu_k$, has a very important role in determining the algorithm efficiency:

- ⋄ Choosing σ nearly equal to 1 allows us to take a full Newton step, but this step is usually very short and does not make much progress towards the solution. However it has the advantage of increasing the proximity to the central path.
- ⋄ Choosing a smaller σ produces a larger Newton step making more progress towards optimality, but this step is generally infeasible and has to be damped. Moreover this kind of step usually tends to move the iterate away from the central path.

We understand that the best choice of σ may vary according to the current iterate: small if a far target is easy to attain and large otherwise. Mehrotra has designed a very efficient way to choose σ according to this principle: the predictor-corrector primal-dual infeasible algorithm [Meh92].

This algorithm first computes an affine-scaling *predictor* step $(\Delta x^{\mathrm{aff}}_k, \Delta y^{\mathrm{aff}}_k, \Delta s^{\mathrm{aff}}_k)$, i.e. solves (4.1) with $\sigma = 0$, targeting directly the optimal limit point of the central path. The maximum feasible step lengths are then computed separately using

$$\begin{aligned} \alpha^{\mathrm{aff,P}}_k &= \arg\max \left\{ \alpha \in [0,1] \mid x_k + \alpha \Delta x^{\mathrm{aff}}_k \geq 0 \right\} \ , \\ \alpha^{\mathrm{aff,D}}_k &= \arg\max \left\{ \alpha \in [0,1] \mid s_k + \alpha \Delta s^{\mathrm{aff}}_k \geq 0 \right\} \ . \end{aligned}$$

Finally, the duality measure of the resulting iterate is evaluated with

$$\mu^{\mathrm{aff}}_{k+1} = \frac{(x_k + \alpha^{\mathrm{aff,P}}_k \Delta x^{\mathrm{aff}}_k)^T (\alpha^{\mathrm{aff,D}}_s \Delta s^{\mathrm{aff}}_k)}{n} \ .$$

This quantity measures how easy it is to progress towards optimality: if it is much smaller than the current duality measure μ_k, we can choose a small σ and hope to make much progress, on the other hand if it is just a little smaller, we have to be more careful and choose σ closer to one, in order to increase proximity to the central path and be in a better position to achieve a large decrease of the duality measure on the *next* iteration. Mehrotra suggested the following heuristic, which has proved to be very efficient in practice

$$\sigma = \left(\frac{\mu_{k+1}^{\text{aff}}}{\mu_k} \right)^3 .$$

We now simply compute a *corrector* step $(\Delta x_k^{\text{corr}}, \Delta y_k^{\text{corr}}, \Delta s_k^{\text{corr}})$ using this σ and take the maximum feasible step lengths separately in the primal and dual spaces.

However, this algorithm can be improved a little further using the following fact. After a full predictor step, the pairwise product $x_i s_i$ is transformed into $(x_i + \Delta x_i^{\text{aff}})(s_i + \Delta s_i^{\text{aff}})$, which is equal to $\Delta x_i^{\text{aff}} \Delta s_i^{\text{aff}}$. Since the Newton method was trying to make $x_i s_i$ equal to zero, this last product measures the error due to the nonlinearity of the equations we are trying to solve. The idea is simply to incorporate this error term in the computation of the corrector step, using the following modification to the right-hand side in (4.1)

$$\begin{pmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S_k & 0 & X_k \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta y_k \\ \Delta s_k \end{pmatrix} = \begin{pmatrix} c - A^T y_k - s_k \\ b - A x_k \\ -X_k S_k e - \Delta X_k^{\text{aff}} \Delta S_k^{\text{aff}} e + \sigma \mu_k e \end{pmatrix} .$$

This strategy of computing a step taking into account the results of a first-order prediction gives rise to a second-order method. The complete algorithm follows:

Given an initial iterate (x_0, y_0, s_0) with duality measure μ_0 such that $x_0 > 0$ and $s_0 > 0$, an accuracy parameter ε and a constant $\rho < 1$ (e.g. 0.995 or 0.999).

Repeat for $k = 0, 1, 2, \ldots$

Compute the predictor Newton step $(\Delta x_k^{\text{aff}}, \Delta y_k^{\text{aff}}, \Delta s_k^{\text{aff}})$ using the linear system (4.1) and $\sigma = 0$.

Compute the maximal step lengths and the resulting duality measure with

$$\begin{aligned} \alpha_k^{\text{aff,P}} &= \arg\max \left\{ \alpha \in [0,1] \mid x_k + \alpha \Delta x_k^{\text{aff}} \geq 0 \right\} , \\ \alpha_k^{\text{aff,D}} &= \arg\max \left\{ \alpha \in [0,1] \mid s_k + \alpha \Delta s_k^{\text{aff}} \geq 0 \right\} , \\ \mu_{k+1}^{\text{aff}} &= \frac{(x_k + \alpha_k^{\text{aff,P}} \Delta x_k^{\text{aff}})^T (s_k + \alpha_k^{\text{aff,D}} \Delta s_k^{\text{aff}})}{n} . \end{aligned}$$

Compute the corrector Newton step $(\Delta x_k^{\text{corr}}, \Delta y_k^{\text{corr}}, \Delta s_k^{\text{corr}})$ using the modified linear system (4.4) and $\sigma = \left(\mu_{k+1}^{\text{aff}} / \mu_k\right)^3$.

Compute the maximal step lengths with

$$\begin{aligned} \alpha_k^{\text{P}} &= \arg\max \{ \alpha \in [0,1] \mid x_k + \alpha \Delta x_k^{\text{corr}} \geq 0 \} , \\ \alpha_k^{\text{D}} &= \arg\max \{ \alpha \in [0,1] \mid s_k + \alpha \Delta s_k^{\text{corr}} \geq 0 \} . \end{aligned}$$

Let $x_{k+1} = x_k + \rho\, \alpha_k^{\text{P}} \Delta x_k^{\text{corr}}$ and $(y_{k+1}, s_{k+1}) = (y_k, s_k) + \rho\, \alpha_k^{\text{D}} (\Delta y_k^{\text{corr}}, \Delta s_k^{\text{corr}})$.

Evaluate μ_{k+1} with $(x_{k+1}^T s_{k+1})/n$.

Until $n\mu_{k+1} < \varepsilon$

It is important to note that the predictor step is only used to compute σ and the right-hand side of (4.4) and is not actually taken. This has a very important effect on the computational work, since the calculation of both the predictor and the corrector step is made with the same current iterate. This implies that the coefficient matrix in the linear systems (4.4) and (4.1) is the same, the only difference being the right-hand side vector. The resolution of the second system will then reuse the factorization of the coefficient matrix and will only need a computationally cheap additional backsubstitution. This property is responsible for the great efficiency of Mehrotra's algorithm: a clever heuristic to decrease the duality measure using very little additional computational work.

5 Implementation

We mention here some important facts about the implementation of interior-point algorithms.

5.1 Linear algebra

It is important to realize that the resolution of the linear system defining the Newton step takes up most of the computing time in interior-point methods (some authors report 80–90 % of the total CPU time). It should be therefore very carefully implemented. Equations (4.1) are not usually solved in this format: some pivoting is done, leading first to the following system (where we define $D_k^2 = S_k^{-1} X_k$)

$$\begin{pmatrix} -D_k^{-2} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x_k \\ \Delta y_k \end{pmatrix} = \begin{pmatrix} c - A^T y_k - \sigma \mu_k X_k^{-1} e) \\ b - A x_k \end{pmatrix} \tag{5.1a}$$

$$\Delta s_k = -s_k + \sigma \mu_k X_k^{-1} e - D_k^{-2} \Delta x_k \; , \tag{5.1b}$$

and then to this one

$$A D_k^2 A^T \Delta y_k = b - A(x_k - D_k^2 c + D_k^2 A^T y_k + \sigma \mu_k S_k^{-1} e) \tag{5.2a}$$

$$\Delta s_k = c - A^T y_k - s_k - A^T \Delta y_k \tag{5.2b}$$

$$\Delta x_k = -x_k + \sigma \mu_k S_k^{-1} e - D_k^2 \Delta s_k \; . \tag{5.2c}$$

System (5.1a) is called the *augmented* system and can be solved with a Bunch-Partlett factorization. However, the most usual way to compute the Newton step is to solve (5.2a), called the *normal* equation, with a Cholevsky factorization, taking advantage of the fact that matrix $AD_k^2A^T$ is positive definite (see [EDAX96] for a discussion). At this stage, it is important to note that most real-world problems have very few nonzero entries in matrix A. It is thus very important to exploit this sparsity in order to reduce both computing times and storage capacity requirements. More specifically, one should try to find a reordering of the rows and columns of matrix $AD_k^2A^T$ that leads to the sparsest Cholevsky factor[20]. This permutation has to be computed only once, since the sparsity pattern of matrix $AD_k^2A^T$ is the same for all iterations.

[20] Because the problem of finding the optimal reordering is NP-hard, heuristics have been developed, e.g. the *minimum degree* and minimum local fill-in heuristics.

On a side note, let us note that the complexity of solving this linear system is $O(n^3)$ arithmetic iterations, which gives the best interior-point methods a total complexity of

$$O\left(n^{3.5}\log\frac{n\mu_0}{\varepsilon}\right)$$

arithmetic operations[21].

5.2 Preprocessing

In most cases, the linear program we want to solve is not formulated in the standard form (1.2). The first task for an interior-point solver is thus to convert it by adding variables and constraints

- Inequality constraints can be transformed into equality constraints with a slack variable: $f^T x \geq b \Leftrightarrow f^T x - s = b$ with $s \geq 0$.
- A free variable can be split into two nonnegative variables: $x = x^+ - x^-$ with $x^+ \geq 0$ and $x^- \geq 0$. However this procedure has some drawbacks[22] and practical solvers usually include a modification of the algorithm to handle free variables directly.
- Lower bounds $l \leq x$ are handled using a translation $x = x' + l$ with $x' \geq 0$.
- Upper bounds $x \leq u$ could be handled using a slack variable, but practical solvers usually implement a variation of the standard form that takes these bounds directly into account.

After this initial conversion, it is not unusual that a series of simple transformations can greatly reduce the size of the problem

- Empty lines and columns are either redundant (and thus may be removed) or make the problem impossible.
- Equality constraints involving only one variable are removed and used to fix the value of this variable.
- Equality constraints involving exactly two variables can be used to pivot out one the variables.
- Two identical lines are either redundant (one of them may thus be removed) or inconsistent (and make the problem impossible).
- Some constraints may allow to compute lower and upper bounds for some variables. These bounds can improve existing bounds, detect redundant constraints or diagnose an impossible problem.

Every practical solver applies these rules (and some others) repeatedly before starting to solve the problem.

[21] A technique of partial updating of the coefficient matrix $AD_k^2A^T$ in the normal equation can reduce this total complexity to $O(n^3)$.

[22] It makes for example the optimal solution set unbounded and the primal-dual strictly feasible set empty.

5.3 Starting point and stopping criteria

The problem of finding a suitable starting point has already been addressed by the homogeneous self-dual embedding technique and the infeasible methods. In both cases, any iterate satisfying $x_0 > 0$ and $s_0 > 0$ can be chosen as starting point. However, the actual performance of the algorithm can be greatly influenced by this choice.

Although there is no theoretical justification for it, the following heuristic is often used to find a starting point. We first solve

$$\min_{x \in \mathbb{R}^n} c^T x + \frac{\omega}{2} x^T x \quad \text{s.t.} \quad Ax = b \quad \text{and} \quad \min_{(y,s) \in \mathbb{R}^m \times \mathbb{R}^n} b^T y + \frac{\omega}{2} s^T s \quad \text{s.t.} \quad A^T y + s = c \,.$$

These convex quadratic programs can be solved analytically at a cost comparable to a single interior-point iteration. The negative components of the optimal x and s the solution are then replaced with a small positive constant to give x_0 and (y_0, s_0).

As described earlier, the stopping criteria is usually a small predefined duality gap ε_g. In the case of an infeasible method, primal and dual infeasibility are also monitored and are required to fall below some predefined value ε_i. One can use for example the following formulas

$$\frac{\|Ax - b\|}{\|b\| + 1} < \varepsilon_i, \quad \frac{\left\|A^T y + s - c\right\|}{\|c\| + 1} < \varepsilon_i, \quad \frac{\left\|c^T x - b^T y\right\|}{\|c^T x\| + 1} < \varepsilon_g \,.$$

The denominators are used to make these measures relative and the +1 constant to avoid division by zero. However, when dealing with an impossible problem, infeasible methods tend to see their iterates diverging towards infinity. Practical solvers usually detect this blow-up and diagnose an impossible problem.

6 Conclusions

The theory of interior-point methods for linear programming is now well established ; textbooks on the topic have been published (see e.g. [Wri97, CRV97, Ye97]). From a practical point of view, interior-point methods compete with the best simplex implementations, especially for large-scale problems.

However some unsatisfying issues remain, in particular the gap between theoretical and implemented algorithms. Another intriguing point is the number of iterations that is practically observed, almost independent from the problem size or varying like $\log n$ or $n^{1/4}$, instead of the $\sqrt{n}$ theoretical bound.

Research is now concentrating on the adaptation of these methods to the nonlinear framework. We'd like to mention the following directions:

- ⋄ In their brilliant monograph [NN94], Nesterov and Nemirovsky develop a complete theory of interior-point methods for *convex* programming. They are able to prove the polynomiality of their method and relate its efficiency to the existence of a certain type of barrier depending on the problem structure (the so-called *self-concordant* barriers).

◇ *Semidefinite programming* is a very promising generalization of linear programming in which the nonnegativity condition on a vector $x \geq 0$ is replaced by the requirement that a symmetric matrix X is positive semidefinite. This kind of problem has numerous applications in various fields, e.g. combinatorial optimization, control, classification, structural optimization, etc. (see [VB96] for more information). The methods we presented here can be adapted in a rather seamless way and practical algorithms are able to solve this kind of problem quite efficiently.

References

[Ans90] K. M. Anstreicher, *On long step path following and SUMT for linear and quadratic programming*, Tech. report, Yale School of Management, Yale University, New Haven, CT, 1990.

[Ans96] K. M. Anstreicher, *Potential reduction algorithms*, In Terlaky [Ter96], pp. 125–158.

[CRV97] T. Terlaky C. Roos and J.-Ph. Vial, *Theory and algorithms for linear optimization. an interior point approach*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley & Sons, Chichester, UK, 1997.

[Dan63] G. B. Dantzig, *Linear programming and extensions*, Princeton University Press, Princeton, N.J., 1963.

[Dik67] I. I. Dikin, *Iterative solution of problems of linear and quadratic programming*, Doklady Akademii Nauk SSSR **174** (1967), 747–748.

[EDAX96] C. Mészáros E. D. Andersen, J. Gondzio and X. Xu, *Implementation of interior-point methods for large scale linear programs*, In Terlaky [Ter96], pp. 189–252.

[FM68] A. V. Fiacco and G. P. McCormick, *Nonlinear programming: Sequential unconstrained minimization techniques*, John Wiley & Sons, New York, 1968, Reprinted in *SIAM Classics in Applied Mathematics*, SIAM Publications, 1990.

[Fri55] K. R. Frisch, *The logarithmic potential method of convex programming*, Tech. report, University Institute of Economics, Oslo, Norway, 1955.

[Gli97] Fr. Glineur, *Etude des méthodes de point intérieur appliquées à la programmation linéaire et à la programmatiuon semidéfinie*, Travail de fin d'études, Faculté Polytechnique de Mons, Mons, Belgium, June 1997.

[Kar84] N. K. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica **4** (1984), 373–395.

[Kha79] L. G. Khachiyan, *A polynomial algorithm in linear programming*, Soviet Mathematics Doklady **20** (1979), 191–194.

[KM72] V. Klee and G. J. Minty, *How good is the simplex algorithm ?*, Inequalities, O. Shisha ed., pp. 159–175, Academic Press, New York, 1972.

[Mas93] W. F. Mascarenhas, *The affine scaling algorithm fails for $\lambda = 0.999$*, Tech. report, Universidade Estadual de Campinas, Campinas S. P., Brazil, October 1993.

[Meh92] S. Mehrotra, *On the implementation of a primal-dual interior point method*, SIAM Journal on Optimization **2** (1992), 575–601.

[NN94] Y. E. Nesterov and A. S. Nemirovsky, *Interior-point polynomial methods in convex programming*, SIAM Studies in Applied Mathematics, SIAM Publications, Philadelphia, 1994.

[Roc70] R. T. Rockafellar, *Convex analysis*, Princeton University Press, Princeton, N. J., 1970.

[Sho70] N. Z. Shor, *Utilization of the operation of space dilatation in the minimization of convex functions*, Kibernetika **1** (1970), 6–12.

[Ter96] T. Terlaky (ed.), *Interior point methods of mathematical programming*, Applied Optimization, vol. 5, Kluwer Academic Publishers, 1996.

[VB96] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Review **38** (1996), 49–95.

[Wri97] S. J. Wright, *Primal-dual interior-point methods*, SIAM, Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[Ye97] Y. Ye, *Interior point algorithms, theory and analysis*, John Wiley & Sons, Chichester, UK, 1997.