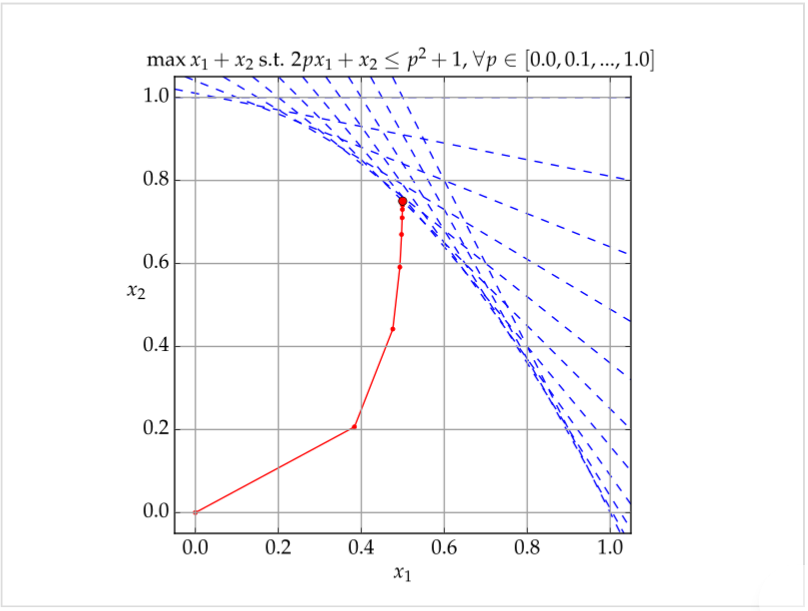
WIKIPEDIA
The Free Encyclopedia

Interior-point method

Interior-point methods (also referred to as **barrier methods** or **IPMs**) are algorithms for solving linear and non-linear convex optimization problems. IPMs combine two advantages of previously-known algorithms:

- Theoretically, their run-time is polynomial—in contrast to the simplex method, which has exponential run-time in the worst case.
- Practically, they run as fast as the simplex method—in contrast to the ellipsoid method, which has polynomial run-time in theory but is very slow in practice.



Example search for a solution. Blue lines show constraints, red points show iterated solutions.

In contrast to the simplex method which traverses the *boundary* of the feasible region, and the ellipsoid method which bounds the feasible region from *outside*, an IPM reaches a best solution by traversing the *interior* of the feasible region—hence the name.

History

An interior point method was discovered by Soviet mathematician I. I. Dikin in 1967.[1] The method was reinvented in the U.S. in the mid-1980s. In 1984, Narendra Karmarkar developed a method for linear programming called Karmarkar's algorithm,[2] which runs in provably polynomial time ($O(n^{3.5}L)$ operations on L-bit numbers, where n is the number of variables and constants), and is also very efficient in practice. Karmarkar's paper created a surge of interest in interior point methods. Two years later, James Renegar invented the first *path-following* interior-point method, with run-time $O(n^3L)$. The method was later extended from linear to convex optimization problems, based on a self-concordant barrier function used to encode the convex set.[3]

Any convex optimization problem can be transformed into minimizing (or maximizing) a linear function over a convex set by converting to the epigraph form.[4]: 143 The idea of encoding the feasible set using a barrier and designing barrier methods was studied by Anthony V. Fiacco, Garth P. McCormick, and others in the early 1960s. These ideas were mainly developed for general nonlinear programming, but they were later abandoned due to the presence of more competitive methods for this class of problems (e.g. sequential quadratic programming).

Yurii Nesterov and Arkadi Nemirovski came up with a special class of such barriers that can be used to encode any convex set. They guarantee that the number of iterations of the algorithm is bounded by a polynomial in the dimension and accuracy of the solution.[5][3]

The class of primal-dual path-following interior-point methods is considered the most successful. Mehrotra's predictor–corrector algorithm provides the basis for most implementations of this class of methods.[6]

Definitions

We are given a convex program of the form:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & x \in G. \end{aligned}$$

where f is a convex function and G is a convex set. Without loss of generality, we can assume that the objective *f* is a linear function. Usually, the convex set *G* is represented by a set of convex inequalities and linear equalities; the linear equalities can be eliminated using linear algebra, so for simplicity we assume there are only convex inequalities, and the program can be described as follows, where the g_i are convex functions:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \text{ for } i = 1, \dots, m. \end{aligned}$$

We assume that the constraint functions belong to some family (e.g. quadratic functions), so that the program can be represented by a finite *vector of coefficients* (e.g. the coefficients to the quadratic functions). The dimension of this coefficient vector is called the *size* of the program. A *numerical solver* for a given family of programs is an algorithm that, given the coefficient vector, generates a sequence of approximate solutions x_t for t=1,2,..., using finitely many arithmetic operations. A numerical solver is called *convergent* if, for any progarm from the family and any positive ε>0, there is some T (which may depend on the program and on ε) such that, for any t>T, the approximate solution x_t is *ε-approximate,* that is:

$f(x) - f^* \leq \varepsilon$

$g_i(x) \leq \varepsilon$ for i in 1,...,m,

x in G,

where f^* is the optimal solution. A solver is called *polynomial* if the total number of arithmetic operations in the first T steps is at most

poly(problem-size) * log(V/ε),

where V is some data-dependent constant, e.g., the difference between the largest and smallest value in the feasible set. In other words, V/ε is the "relative accuracy" of the solution - the accuracy w.r.t. the largest coefficient. log(V/ε) represents the number of "accuracy digits". Therefore, a solver is 'polynomial' if each additional digit of accuracy requires a number of operations that is polynomial in the problem size.

Types

Types of interior point methods include:

- **Potential reduction methods**: Karmarkar's algorithm was the first one.
- **Path-following methods**: the algorithms of James Renegar[7] and Clovis Gonzaga[8] were the first ones.

- **Primal-dual methods**.

Path-following methods

Idea

Given a convex optimization program (P) with constraints, we can convert it to an *unconstrained* program by adding a barrier function. Specifically, let *b* be a smooth convex function, defined in the interior of the feasible region *G*, such that for any sequence {x_j in interior(G)} whose limit is on the boundary of *G*: $\lim_{j\to\infty} b(x_j) = \infty$. We also assume that *b* is non-degenerate, that is: $b''(x)$ is positive definite for all x in interior(G). Now, consider the family of programs:

(P_t) minimize t * f(x) + b(x)

Technically the program is restricted, since *b* is defined only in the interior of *G*. But practically, it is possible to solve it as an unconstrained program, since any solver trying to minimize the function will not approach the boundary, where *b* approaches infinity. Therefore, (P_t) has a unique solution - denote it by *x**(*t*). The function *x** is a continuous function of *t*, which is called the *central path*. All limit points of *x**, as *t* approaches infinity, are optimal solutions of the original program (P).

A **path-following method** is a method of tracking the function *x** along a certain increasing sequence t_1,t_2,..., that is: computing a good-enough approximation x_i to the point *x**(t_i), such that the difference x_i - *x**(t_i) approaches 0 as *i* approaches infinity; then the sequence x_i approaches the optimal solution of (P). This requires to specify three things:

- The barrier function b(x).
- A policy for determining the penalty parameters t_i.
- The unconstrained-optimization solver used to solve (P_i) and find x_i, such as Newton's method. Note that we can use each x_i as a starting-point for solving the next problem (P_{i+1}).

The main challenge in proving that the method is polytime is that, as the penalty parameter grows, the solution gets near the boundary, and the function becomes steeper. The run-time of solvers such as Newton's method becomes longer, and it is hard to prove that the total runtime is polynomial.

Renegar[7] and Gonzaga[8] proved that a specific instance of a path-following method is polytime:

- The constraints (and the objective) are linear functions;
- The barrier function is logarithmic: b(x) := - sum$_j$ log(-g_j(*x*)).
- The penalty parameter *t* is updated geometrically, that is, $t_{i+1} := \mu \cdot t_i$, where *μ* is a constant (they took $\mu = 1 + 0.001 \cdot \sqrt{m}$,where *m* is the number of inequality constraints);
- The solver is Newton's method, and a *single* step of Newton is done for each single step in *t*.

They proved that, in this case, the difference x_i - *x**(t_i) remains at most 0.01, and f(x_i) - f* is at most 2**m*/t_i. Thus, the solution accuracy is proportional to 1/t_i, so to add a single accuracy-digit, it is suffiicent to multiply t_i by 2 (or any other constant factor), which requires O(sqrt(*m*)) Newton steps. Since each Newton step takes O($m\, n^2$) operations, the total complexity is O($m^{3/2}\, n^2$) operations for accuracy digit.

Yuri Nesterov extended the idea from linear to non-linear programs. He noted that the main property of the logarithmic barrier, used in the above proofs, is that it is self-concordant with a finite barrier parameter. Therefore, many other classes of convex programs can be solved in polytime using a path-following method, if we can find a suitable self-concordant barrier function for their feasible region.[3]: Sec.1

Details

We are given a convex optimization problem (P) in "standard form":

minimize $c^T x$ s.t. x in G,

where G is convex and closed. We can also assume that G is bounded (we can easily make it bounded by adding a constraint $|x| \le R$ for some sufficiently large R).[3]: Sec.4

To use the interior-point method, we need a self-concordant barrier for G. Let b be an M-self-concordant barrier for G, where $M \ge 1$ is the self-concordance parameter. We assume that we can compute efficiently the value of b, its gradient, and its Hessian, for every point x in the interior of G.

For every $t>0$, we define the *penalized objective* $\mathbf{f_t(x) := c^T x + b(x)}$. We define the path of minimizers by: $\mathbf{x^*(t) := \arg\min f_t(x)}$. We apporimate this path along an increasing sequence t_i. The sequence is initialized by a certain non-trivial two-phase initialization procedure. Then, it is updated according to the following rule: $t_{i+1} := \mu \cdot t_i$.

For each t_i, we find an approximate minimum of f_{ti}, denoted by x_i. The approximate minimum is chosen to satisfy the following "closeness condition" (where L is the *path tolerance*):

$$\sqrt{[\nabla_x f_t(x_i)]^T [\nabla_x^2 f_t(x_i)]^{-1} [\nabla_x f_t(x_i)]} \le L.$$

To find x_{i+1}, we start with x_i and apply the damped Newton method. We apply several steps of this method, until the above "closeness relation" is satisfied. The first point that satisfies this relation is denoted by x_{i+1}.[3]: Sec.4

Convergence and complexity

The convergence rate of the method is given by the following formula, for every i:[3]: Prop.4.4.1

$$c^T x_i - c^* \le \frac{2M}{t_0} \mu^{-i}$$

Taking $\mu = \left(1 + r/\sqrt{M}\right)$, the number of Newton steps required to go from x_i to x_{i+1} is at most a fixed number, that depends only on r and L. In particular, the total number of Newton steps required to find an ε-approximate solution (i.e., finding x in G such that $c^T x$ - $c^* \le \varepsilon$) is at most:[3]: Thm.4.4.1

$$O(1) \cdot \sqrt{M} \cdot \ln\left(\frac{M}{t_0 \varepsilon} + 1\right)$$

where the constant factor O(1) depends only on *r* and *L*. The number of Newton steps required for the two-step initialization procedure is at most:[3]: Thm.4.5.1

$$O(1)\cdot\sqrt{M}\cdot\ln\left(\frac{M}{1-\pi_{x^*_f}(\bar{x})}+1\right)+O(1)\cdot\sqrt{M}\cdot\ln\left(\frac{M\mathbf{Var}_G(c)}{\epsilon}+1\right)$$

where the constant factor O(1) depends only on *r* and *L*, and $\mathbf{Var}_G(c) := \max_{x\in G} c^T x - \min_{x\in G} c^T x$, and $\bar{x}$ is some point in the interior of *G*. Overall, the overall Newton complexity of finding an *ε*-approximate solution is at most

$$O(1)\cdot\sqrt{M}\cdot\ln\left(\frac{V}{\varepsilon}+1\right)$$, where V is some problem-dependent constant: $V = \frac{\mathbf{Var}_G(c)}{1-\pi_{x^*_f(\bar{x})}}$.

Each Newton step takes O(n^3) arithmetic operations.

Initialization: phase-I methods

To initialize the path-following methods, we need a point in the relative interior of the feasible region *G*. In other words: if *G* is defined by the inequalities $g_i(x) \leq 0$, then we need some *x* for which $g_i(x) < 0$ for all *i* in 1,...,*m*. If we do not have such a point, we need to find one using a so-called **phase I method**.[4]: 11.4 A simple phase-I method is to solve the following convex program:

$$\begin{aligned} \text{minimize} \quad & s \\ \text{subject to} \quad & g_i(x) \leq s \text{ for } i = 1,\dots,m \end{aligned}$$

Denote the optimal solution by x*,*s**.

- If *s**<0, then we know that x* is an interior point of the original problem and can go on to "phase II", which is solving the original problem.
- If *s**>0, then we know that the original program is infeasible - the feasible region is empty.
- If *s**=0 and it is attained by some solution x*, then the problem is feasible but has no interior point; if it is not attained, then the problem is infeasible.

For this program it is easy to get an interior point: we can take arbitrarily *x*=0, and take *s* to be any number larger than max(f_1(0),...,f_m(0)). Therefore, it can be solved using interior-point methods. However, the run-time is proportional to log(1/*s**). As s* comes near 0, it becomes harder and harder to find an exact solution to the phase-I problem, and thus harder to decide whether the original problem is feasible.

Practical considerations

The theoretic guarantees assume that the penalty parameter is increased at the rate $\mu = \left(1 + r/\sqrt{M}\right)$, so the worst-case number of required Newton steps is $O(\sqrt{M})$. In theory, if *μ* is larger (e.g. 2 or more), then the worst-case number of required Newton steps is in $O(M)$. However, in practice, larger *μ* leads to a much faster convergence. These methods are called *long-step methods*.[3]: Sec.4.6 In practice, if *μ* is between 3 and

100, then the program converges within 20-40 Newton steps, regardless of the number of constraints (though the runtime of each Newton step of course grows with the number of constraints). The exact value of μ within this range has little effect on the performane.[4]: chpt.11

Potential-reduction methods

For potential-reduction methods, the problem is presented in the *conic form*:[3]: Sec.5

minimize $c^T x$ s.t. x in $\{b+L\} \cap K$,

where b is a vector in R^n, L is a linear subspace in R^n (so $b+L$ is an affine plane), and K is a closed pointed convex cone with a nonempty interior. Every convex program can be converted to the conic form. To use the potential-reduction method (specifically, the extension of Karmarkar's algorithm to convex programming), we need the following assumptions:[3]: Sec.6

- A. The feasible set $\{b+L\} \cap K$ is bounded, and intersects the interior of the cone K.
- B. We are given in advance a strictly-feasible solution x^, that is, a feasible solution in the interior of K.
- C. We know in advance the optimal objective value, c*, of the problem.
- D. We are given an M-logarithmically-homogeneous self-concordant barrier F for the cone K.

Assumptions A, B and D are needed in most interior-point methods. Assumption C is specific to Karmarkar's approach; it can be alleviated by using a "sliding objective value". It is possible to further reduce the program to the *Karmarkar format*:

minimize $s^T x$ s.t. x in $M \cap K$ and $e^T x = 1$

where M is a linear subspace of in R^n, and the optimal objective value is 0. The method is based on the following scalar potential function:

$v(x) = F(x) + M \ln (s^T x)$

where F is the M-self-concordant barrier for the feasible cone. It is possible to prove that, when x is strictly feasible and $v(x)$ is very small (- very negative), x is approximately-optimal. The idea of the potential-reduction method is to modify x such that the potential at each iteration drops by at least a fixed constant X (specifically, X=1/3-ln(4/3)). This implies that, after i iterations, the difference between objective value and the optimal objective value is at most $V * \exp(-i\, X / M)$, where V is a data-dependent constant. Therefore, the number of Newton steps required for an ε-approximate solution is at most $O(1) \cdot M \cdot \ln\left(\frac{V}{\varepsilon} + 1\right) + 1$.

Note that in path-following methods the expression is $\sqrt{M}$ rather than M, which is better in theory. But in practice, Karmarkar's method allows taking much larger steps towards the goal, so it may converge much faster than the theoretical guarantees.

Primal-dual methods

The primal-dual method's idea is easy to demonstrate for constrained nonlinear optimization.[9][10] For simplicity, consider the following nonlinear optimization problem with inequality constraints:

$$\begin{aligned}\text{minimize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathbb{R}^n, \\ & c_i(x) \geq 0 \text{ for } i = 1, \ldots, m, \\ \text{where} \quad & f : \mathbb{R}^n \to \mathbb{R},\ c_i : \mathbb{R}^n \to \mathbb{R}.\end{aligned} \qquad (1)$$

This inequality-constrained optimization problem is solved by converting it into an unconstrained objective function whose minimum we hope to find efficiently. Specifically, the logarithmic barrier function associated with (1) is

$$B(x,\mu) = f(x) - \mu \sum_{i=1}^{m} \log(c_i(x)). \quad (2)$$

Here μ is a small positive scalar, sometimes called the "barrier parameter". As μ converges to zero the minimum of $B(x,\mu)$ should converge to a solution of (1).

The gradient of a differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ is denoted ∇h. The gradient of the barrier function is

$$\nabla B(x,\mu) = \nabla f(x) - \mu \sum_{i=1}^{m} \frac{1}{c_i(x)} \nabla c_i(x). \quad (3)$$

In addition to the original ("primal") variable x we introduce a Lagrange multiplier-inspired dual variable $\lambda \in \mathbb{R}^m$

$$c_i(x)\lambda_i = \mu, \quad \forall i = 1, \ldots, m. \quad (4)$$

Equation (4) is sometimes called the "perturbed complementarity" condition, for its resemblance to "complementary slackness" in KKT conditions.

We try to find those (x_μ, λ_μ) for which the gradient of the barrier function is zero.

Substituting $1/c_i(x) = \lambda_i/\mu$ from (4) into (3), we get an equation for the gradient:

$$\nabla B(x_\mu, \lambda_\mu) = \nabla f(x_\mu) - J(x_\mu)^T \lambda_\mu = 0, \quad (5)$$

where the matrix J is the Jacobian of the constraints $c(x)$.

The intuition behind (5) is that the gradient of $f(x)$ should lie in the subspace spanned by the constraints' gradients. The "perturbed complementarity" with small μ (4) can be understood as the condition that the solution should either lie near the boundary $c_i(x) = 0$, or that the projection of the gradient ∇f on the constraint component $c_i(x)$ normal should be almost zero.

Let (p_x, p_λ) be the search direction for iteratively updating (x, λ). Applying Newton's method to (4) and (5), we get an equation for (p_x, p_λ):

$$\begin{pmatrix} H(x,\lambda) & -J(x)^T \\ \operatorname{diag}(\lambda)J(x) & \operatorname{diag}(c(x)) \end{pmatrix} \begin{pmatrix} p_x \\ p_\lambda \end{pmatrix} = \begin{pmatrix} -\nabla f(x) + J(x)^T\lambda \\ \mu 1 - \operatorname{diag}(c(x))\lambda \end{pmatrix},$$

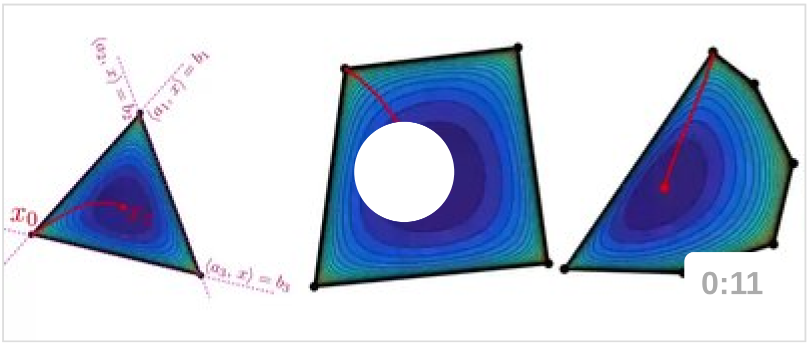
where H is the Hessian matrix of $B(x,\mu)$, $\operatorname{diag}(\lambda)$ is a diagonal matrix of λ, and $\operatorname{diag}(c(x))$ is the diagonal matrix of $c(x)$.

Because of (1), (4) the condition

$$\lambda \geq 0$$

should be enforced at each step. This can be done by choosing appropriate α:

$$(x,\lambda) \to (x + \alpha p_x, \lambda + \alpha p_\lambda).$$


x_0
$\langle a_1, x\rangle = b_1$
$\langle a_2, x\rangle = b_2$
$\langle a_3, x\rangle = b_3$
0:11

Trajectory of the iterates of *x* by using the interior point method.

Types of Convex Programs Solvable via Interior-Point Methods

Here are some special cases of convex programs that can be solved efficiently by interior-point methods.[3]: Sec.10

Linear programs

Consider a linear program of the form:

$$\begin{aligned} \text{minimize} \quad & c^\top x \\ \text{subject to} \quad & Ax \leq b. \end{aligned}.$$

We can apply path-following methods with the barrier

$$b(x) := -\sum_{j=1}^{m} \ln(b_j - a_j^T x).$$

The function b is self-concordant with parameter *M*=*m* (the number of constraints). Therefore, the number of required Newton steps for the path-following method is O(mn^2), and the total runtime complexity is O($m^{3/2} n^2$).

Quadratically constrained quadratic programs

Given a quadratically constrained quadratic program of the form:

$$\begin{aligned} \text{minimize} \quad & d^\top x \\ \text{subject to} \quad & f_j(x) := x^\top A_j x + b_j^\top x + c_j \leq 0 \quad \text{for all } j = 1,\dots,m, \end{aligned}$$

where all matrices A_j are positive-semidefinite matrices. We can apply path-following methods with the barrier

$$b(x) := -\sum_{j=1}^{m} \ln(-f_j(x)).$$

The function b is a self-concordant barrier with parameter M=m. The Newton complexity is O((m+n)n^2), and the total runtime complexity is O($m^{1/2}$ (m+n) n^2).

L_p norm approximation

Consider a problem of the form

$$\text{minimize} \quad \sum_j |v_j - u_j^\top x|_p \, ,$$

where each u_j is a vector, each v_j is a scalar, and $|\cdot|_p$ is an L_p norm with $1 < p < \infty$. After converting to the standard form, we can apply path-following methods with a self-concordant barrier with parameter M=4m. The Newton complexity is O((m+n)n^2), and the total runtime complexity is O($m^{1/2}$ (m+n) n^2).

Geometric programs

Consider the problem

$$\begin{aligned} \text{minimize} \quad & f_0(x) := \sum_{i=1}^{k} c_{i0} \exp(a_i^\top x) \\ \text{subject to} \quad & f_j(x) := \sum_{i=1}^{k} c_{ij} \exp(a_i^\top x) \leq d_j \quad \text{for all } j = 1, \ldots, m. \end{aligned}$$

There is a self-concordant barrier with parameter 2k+m. The path-following method has Newton complexity O(mk^2+k^3+n^3) and total complexity O((k+m)$^{1/2}$[mk^2+k^3+n^3]).

Semidefinite programs

Interior point methods can be used to solve semidefinite programs.[3]: Sec.11

See also

- Affine scaling
- Augmented Lagrangian method
- Chambolle-Pock algorithm
- Karush–Kuhn–Tucker conditions

- Penalty method

References

1. Dikin, I.I. (1967). "Iterative solution of problems of linear and quadratic programming" (https://zbmath.org/?q=an:0189.19504). *Dokl. Akad. Nauk SSSR*. **174** (1): 747–748.
2. Karmarkar, N. (1984). "A new polynomial-time algorithm for linear programming" (https://web.archive.org/web/20131228145520/http://retis.sssup.it/~bini/teaching/optim2010/karmarkar.pdf) (PDF). *Proceedings of the sixteenth annual ACM symposium on Theory of computing – STOC '84*. p. 302. doi:10.1145/800057.808695 (https://doi.org/10.1145%2F800057.808695). ISBN 0-89791-133-4. Archived from the original (http://retis.sssup.it/~bini/teaching/optim2010/karmarkar.pdf) (PDF) on 28 December 2013.
3. Arkadi Nemirovsky (2004). *Interior point polynomial-time methods in convex programming* (https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=8c3cb6395a35cb504019f87f447d65cb6cf1cdf0).
4. Boyd, Stephen; Vandenberghe, Lieven (2004). *Convex Optimization*. Cambridge: Cambridge University Press. ISBN 978-0-521-83378-3. MR 2061575 (https://mathscinet.ams.org/mathscinet-getitem?mr=2061575).
5. Wright, Margaret H. (2004). "The interior-point revolution in optimization: History, recent developments, and lasting consequences" (https://doi.org/10.1090%2FS0273-0979-04-01040-7). *Bulletin of the American Mathematical Society*. **42**: 39–57. doi:10.1090/S0273-0979-04-01040-7 (https://doi.org/10.1090%2FS0273-0979-04-01040-7). MR 2115066 (https://mathscinet.ams.org/mathscinet-getitem?mr=2115066).
6. Potra, Florian A.; Stephen J. Wright (2000). "Interior-point methods" (https://doi.org/10.1016%2FS0377-0427%2800%2900433-7). *Journal of Computational and Applied Mathematics*. **124** (1–2): 281–302. doi:10.1016/S0377-0427(00)00433-7 (https://doi.org/10.1016%2FS0377-0427%2800%2900433-7).
7. Renegar, James (1 January 1988). "A polynomial-time algorithm, based on Newton's method, for linear programming" (https://doi.org/10.1007/BF01580724). *Mathematical Programming*. **40** (1): 59–93. doi:10.1007/BF01580724 (https://doi.org/10.1007%2FBF01580724). ISSN 1436-4646 (https://www.worldcat.org/issn/1436-4646).
8. Gonzaga, Clovis C. (1989), Megiddo, Nimrod (ed.), "An Algorithm for Solving Linear Programming Problems in O(n3L) Operations" (https://doi.org/10.1007/978-1-4613-9617-8_1), *Progress in Mathematical Programming: Interior-Point and Related Methods*, New York, NY: Springer, pp. 1–28, doi:10.1007/978-1-4613-9617-8_1 (https://doi.org/10.1007%2F978-1-4613-9617-8_1), ISBN 978-1-4613-9617-8, retrieved 22 November 2023
9. Mehrotra, Sanjay (1992). "On the Implementation of a Primal-Dual Interior Point Method". *SIAM Journal on Optimization*. **2** (4): 575–601. doi:10.1137/0802028 (https://doi.org/10.1137%2F0802028).
10. Wright, Stephen (1997). *Primal-Dual Interior-Point Methods*. Philadelphia, PA: SIAM. ISBN 978-0-89871-382-4.

- Bonnans, J. Frédéric; Gilbert, J. Charles; Lemaréchal, Claude; Sagastizábal, Claudia A. (2006). *Numerical optimization: Theoretical and practical aspects* (https://www.springer.com/mathematics/applications/book/978-3-540-35445-1). Universitext (Second revised ed. of translation of 1997 French ed.). Berlin: Springer-Verlag. pp. xiv+490. doi:10.1007/978-3-540-35447-5 (https://doi.org/10.1007%2F978-3-540-35447-5). ISBN 978-3-540-35445-1. MR 2265882 (https://mathscinet.ams.org/mathscinet-getitem?mr=2265882).
- Nocedal, Jorge; Stephen Wright (1999). *Numerical Optimization*. New York, NY: Springer. ISBN 978-0-387-98793-4.
- Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007). "Section 10.11. Linear Programming: Interior-Point Methods" (http://apps.nrbook.com/empanel/index.html#pg=537).

Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. ISBN 978-0-521-88068-8.

-