**CS 521: Linear Programming** **Rutgers: Fall 2022**

# Lecture 12

November 23, 2022

*Instructor: Sepehr Assadi* *Scribe: Yi Wang, Haizhou Shi, Jingquan Yan, and Zihao Xu*

**Disclaimer**: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

## Topics of this Lecture



# 1 Newton's Method

The goal of today's lecture is to provide a high level overview of *Interior Point Methods*, yet another general family of algorithms for solving LPs in polynomial time. For this and the subsequent lecture, we follow the excellent book by Vishnoi on this topic [1, Chapters 9 and 10].

To start, we need to go over one of their key components, the *Newton's Method* for solving unconstrained optimization problems. The Newton's method in general is an algorithm for finding *roots* of a given vector-valued function $g : \mathbb{R}^n \to \mathbb{R}^n$, i.e., finding an $r \in \mathbb{R}^n$ such that $g(r)$ is the all-zero vector $(0, 0, \ldots, 0)$; for simplicity of exposition, we denote $g(r) = 0$ in this case. We start by examining Newton's method for one-dimensional functions, i.e., functions from $\mathbb{R}$ to $\mathbb{R}$.

## 1.1 Newton's Method for One-Dimensional Functions

Let $g : \mathbb{R} \to \mathbb{R}$ be a differentiable function and suppose our goal is to find a point $r \in \mathbb{R}$ such that $g(r) = 0$. Let us start at point $x_0$ with $g(x_0) \neq 0$. Can we iteratively update this point to $x_1, x_2, \ldots$ to eventually converge to the point $r$ with $g(r) = 0$? This is basically what the Newton's method does.

We now discuss how to update $x_0$ to obtain the next point $x_1$. Consider the line *tangent* to $g$ at the point $(x_0, g(x_0))$; we are going to "approximate" $g$ by this line $h(x) : \mathbb{R} \to \mathbb{R}$, find the root of $h$ instead—which is simple as it is a line with an explicit formula—and let $x_1$ be this root (see Figure 1 for an illustration).

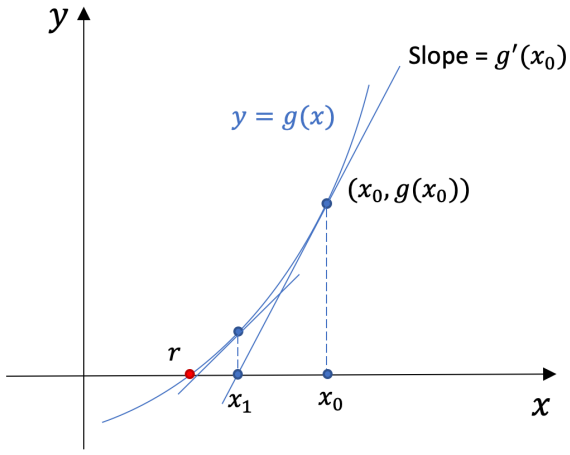


Figure 1: An illustration of Newton's method.

Letting $h(x) = ax + b$ (or equivalently a line in $\mathbb{R}^2$), and $g'$ be the derivative of $g$, we have that

$$\begin{aligned} a &= g'(x_0) && \text{(as } h \text{ is the tangent of } g \text{ on } x_0\text{)} \\ ax_0 + b &= h(x_0) = g(x_0) && \text{(as } h \text{ and } g \text{ intersect on the point } x_0\text{)} \\ ax_1 + b &= h(x_1) = 0. && \text{(as } x_1 \text{ is the root of } h\text{)} \end{aligned}$$

Combining these three equations gives us the following formula for $x_1$:

$$x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}. \tag{1}$$

Thus, as an iterative method, we can simply state the Newton's method as the following algorithm:

**Algorithm:** Newton's Method for a one-dimensional function $g : \mathbb{R} \to \mathbb{R}$.

(*i*) Start with some initial point $x_0 \in \mathbb{R}$.

(*ii*) For $t = 0$ to $T$ iterations: update $x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)}$.

(*iii*) Return $x_T$ as the answer.

Before getting to the analysis of $g$ and to highlight some, perhaps peculiar, aspects of the Newton's method, let us consider an example first.

### 1.1.1 An Example

Consider the following function and its derivative:

$$g(x) = 1 - \frac{1}{x} \qquad g'(x) = \frac{1}{x^2}.$$

Suppose our goal is to find the root of $g$ (which is clearly 1) using the Newton's method. The update rule for this function is

$$x_{t+1} = x_t - \frac{g(x_t)}{g'(x_t)} = x_t - x_t^2 \cdot (1 - \frac{1}{x_t}) = 2x_t - x_t^2.$$

Let us now check whether or not this approach converges to the right answer. For any iteration $t \geqslant 0$, define the **error** $e_t$ as

$$e_t := |1 - x_t|;$$

namely, how far we are currently from the right answer which is 1. Using the update rule, we have,

$$e_{t+1} = |1 - x_{t+1}| = |1 - 2x_t + x_t^2| = (1 - x_t)^2 = e_t^2.$$

Using this, we can consider the following cases:

- **Case 1**: Suppose $e_0 < 1$ or alternatively $0 < x_0 < 2$: then, each iteration quadratically *decreases* the error and thus $\lim_{t\to\infty} e_t = 0$ and we converge to the optimal solution eventually.

  In fact, if we have that $e_0 < 1/2$ or alternatively $1/2 < x_0 < 3/2$, then we get

  $$e_1 < 2^{-2} \qquad e_2 < 2^{-4} \qquad e_3 < 2^{-8} \qquad \ldots \qquad e_t < 2^{-2^t}.$$

  Consequently, for any $\varepsilon > 0$, to reduce $e_t < \varepsilon$ or alternatively get $x_t \in (1 - \varepsilon, 1 + \varepsilon)$, we only need to have $t = \log\log(1/\varepsilon)$ iterations, which is an extremely fast rate of convergence!

- **Case 2**: Suppose now $e_0 = 1$ or alternatively $x_0 \in \{0, 2\}$: then, in each iteration, we have $e_t = 1$ – this means that we never converge to the optimal solution. In fact, as is clear from the update rule, in both these cases, we simply have $x_{t+1} = x_t$, and thus no changes are happening throughout the iterations.

- **Case 3**: Finally, suppose $e_0 > 1$ or alternatively $x_0 \notin [0, 1]$: then, each iteration quadratically *increases* the error and thus $\lim_{t\to\infty} e_t = +\infty$. In other words, each iteration takes us even further from the correct answer!

> **Remark.** The first case of this example is considered the **quadratic convergence** regime of the Newton's method and our goal when analyzing this method is to guarantee that we are in this case. As this example clearly illustrates, the starting point $x_0$ (and/or some other properties of the function $g$) are quite crucial for guaranteeing the quadratic convergence rate. Thus, in our analysis of this algorithm, we crucially need to make *some* assumption about $x_0$ and $g$.

### 1.1.2 The Analysis

In most applications of the Newton's method, including in solving LPs that is our focus, it is customary to state and analyze this result only for a single update step. We shall do the same in the rest of this lecture (and the next one) as well.

**Theorem 1** (Error Bound of Single Update in Newton's Method)**.** *Let $g : \mathbb{R} \to \mathbb{R}$ be a twice differentiable function. Let $r$ be a root of $g$, i.e., $g(r) = 0$. Additionally, let $x_0 \in \mathbb{R}$ be any starting point and define*

$$M := \sup_{\min(r,x_0) < y < \max(r,x_0)} \left| \frac{g''(y)}{2 \cdot g'(x_0)} \right|.$$

*For a single update step of the Newton's method*

$$x_1 = x_0 - \frac{g(x_0)}{g'(x_0)},$$

*we have have the following bound on the errors:*

$$|r - x_1| \leqslant M \cdot |r - x_0|^2.$$

To prove this theorem we need to use the standard *mean value theorem* stated as follows.

**Fact 2.** *For any differentiable function $f$ and interval $(a, b) \in \mathbb{R}^n$, there exists a point $c \in (a, b)$ such that*

$$f'(c) = \frac{f(b) - f(a)}{b - a}.$$

As a corollary of this result and Taylor's expansion, we obtain the following statement.

**Proposition 3.** *Let $g : \mathbb{R} \to \mathbb{R}$ be any twice differentiable function and $a < b$ be two points in $\mathbb{R}$. Then, there exists a point $c \in (a, b)$ such that*

$$g(b) = g(a) + (b - a) \cdot g'(a) + \frac{1}{2} \cdot (b - a)^2 \cdot g''(c).$$

*Proof.* Define the function $h : \mathbb{R} \to \mathbb{R}$ where for all $x > a$,

$$h(x) = g(x) - g(a) - (x - a) \cdot g'(a),$$

namely, the error of the first-order Taylor's approximation of $g$ around the point $a$. We have that

$$h'(x) = g'(x) - g'(a),$$

or alternatively

$$\frac{h'(x)}{x - a} = \frac{g'(x) - g'(a)}{x - a}.$$

By Fact 2, there exists some $c_x \in (a, x)$ such that

$$\frac{g'(x) - g'(a)}{x - a} = g''(c_x),$$

where we used the fact that $g'$ is differentiable (as $g$ is twice differentiable). We thus have

$$h'(x) = (x - a) \cdot g''(c_x),$$

and thus taking its integral, we have,

$$h(x) = \frac{1}{2} \cdot (x - a)^2 \cdot g''(c_x).$$

Plugging in this bound in the definition of $h$ earlier and rearranging the terms imply that

$$g(x) = g(a) + (x - a) \cdot g'(a) + \frac{1}{2} \cdot (x - a)^2 \cdot g''(c_x).$$

Finally, taking $x = b$, we have that there exists some $c \in (a, b)$ where

$$g(b) = g(a) + (b - a) \cdot g'(a) + \frac{1}{2} \cdot (b - a)^2 \cdot g''(c),$$

concluding the proof. □

We are now ready to prove Theorem 1.

*Proof of Theorem 1.* Let us first assume that $r > x_0$. By Proposition 3 applied to $b = r$ and $a = x_0$, we get that there exists some $c \in (x_0, r)$ such that

$$g(r) = g(x_0) + (r - x_0) \cdot g'(x_0) + \frac{1}{2} \cdot (r - x_0)^2 \cdot g''(c).$$

By further substituting $g(r) = 0$ (as $r$ is a root) and $g(x_0) = g'(x_0)(x_0 - x_1)$ (by the update rule), we have

$$0 = g'(x_0)(r - x_1) + \frac{1}{2}(r - x_0)^2 g''(c),$$

which implies that

$$|r - x_1| \leqslant \left|\frac{g''(c)}{2g'(x_0)}\right| \cdot |r - x_0|^2 \leqslant \sup_{r<y<x_0} \left|\frac{g''(y)}{2g'(x_0)}\right| \cdot |r - x_0|^2 = M \cdot |r - x_0|^2.$$

The other case when $r < x_0$ can be proven exactly as above by symmetry by considering the interval $(r, x_0)$ instead of $(x_0, r)$. □

## 1.2 Newton's Method for Multi-Dimensional Functions

We now consider the general case of the Newton's method for multi-dimensional functions that will be needed for our purpose later. In order to do this generalization, we need to define an analogue of derivative for multi-dimensional functions.

**Definition 4.** For a function $g : \mathbb{R}^n \to \mathbb{R}^n$, the **Jacobian** (matrix) of $g$, denoted by $J_g$ is the matrix of all its partial derivatives. I.e., if we denote $g(x) = (g_1(x), \ldots, g_n(x))$, then

$$J_g(x) = \begin{bmatrix} \frac{\partial g_1}{x_1} & \cdots & \frac{\partial g_1}{x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{x_1} & \cdots & \frac{\partial g_n}{x_n} \end{bmatrix}.$$

Using this, we can define the newton step for a function $g : \mathbb{R}^n \to \mathbb{R}^n$ at a point $x_0$ as

$$x_1 = x_0 - (J_g(x_0))^{-1} \cdot g(x_0). \tag{2}$$

We need one more definition before we can state the analysis of this method.

**Definition 5.** For any matrix $A \in \mathbb{R}^{n \times n}$, the **spectral norm** of $A$ is defined as

$$\|A\|_2 := \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}.$$

Intuitively, the spectral norm of a matrix is the maximum scale by which the matrix can stretch a vector.

We now have the following generalization of Theorem 1.

**Theorem 6.** *Let $g : \mathbb{R}^n \to \mathbb{R}^n$ be a twice differentiable function, $r$ be a root of $g$, and $x_0$ be an initial point.*

*(i) $J_g(x_0)$ is full-dimensional and $\|J_g(x_0)^{-1}\|_2 \leqslant \alpha$;*

*(ii) For all $x \in \{t \cdot r + (1 - t) \cdot x_0 \mid t \in [0, 1]\}$, $\|J_g(x) - J_g(x_0)\|_2 \leqslant \beta \cdot \|x - x_0\|_2$.*

*Let $M := \frac{\alpha \cdot \beta}{2}$. For a single update step of the Newton's method*

$$x_1 = x_0 - (J_g(x_0))^{-1} \cdot g(x_0),$$

*we have have the following bound on the errors:*

$$\|r - x_1\|_2 \leqslant M \cdot \|r - x_0\|_2^2.$$

*Proof.* We follow the same strategy as in the proof of Theorem 1. We first claim that

$$g(r) - g(x_0) = \int_{t=0}^{1} J_g(t \cdot r + (1-t) \cdot x_0) \cdot (r - x_0)\ dt. \tag{3}$$

To see this, define the function $h : [0,1] \to \mathbb{R}$ where for $t \in [0,1]$,

$$h(t) = g(t \cdot r + (1-t) \cdot x_0).$$

By the Fundamental Theorem of Calculus,

$$g(r) - g(x_0) = h(1) - h(0) = \int_{t=0}^{1} h'(t)\ dt = \int_{t=0}^{1} J_g(t \cdot r + (1-t) \cdot x_0) \cdot (r - x_0)\ dt,$$

as stated in Eq (3).

We can now restate the "error vector" $r - x_1$ as follows.

$$\begin{aligned}
r - x_1 &= r - x_0 + J_g(x_0)^{-1} \cdot g(x_0) && \text{(by the update rule of Newton's method)} \\
&= r - x_0 - J_g(x_0)^{-1} \cdot (g(r) - g(x_0)) && \text{(as } g(r) = 0 \text{ since } r \text{ is a root)} \\
&= (r - x_0) - J_g(x_0)^{-1} \cdot \int_{t=0}^{1} J_g(t \cdot r + (1-t) \cdot x_0) \cdot (r - x_0)\ dt && \text{(by Eq (3))} \\
&= -J_g(x_0)^{-1} \int_{t=0}^{1} \Big( J_g(t \cdot r + (1-t) \cdot x_0) - J_g(x_0) \Big) \cdot (r - x_0)\ dt. \\
&\quad \text{(by writing } (r - x_0) = J_g(x_0)^{-1} \cdot J_g(x_0) \cdot (r - x_0) \text{ and factoring this in the integral)}
\end{aligned}$$

Using this, we bound the norm of $(r - x_1)$ as follows:

$$\begin{aligned}
\|r - x_1\|_2 &= \|J_g(x_0)^{-1} \int_{t=0}^{1} \Big( J_g(t \cdot r + (1-t) \cdot x_0) - J_g(x_0) \Big) \cdot (r - x_0)\ dt\, \|_2 \\
&\leqslant \|J_g(x_0)^{-1}\|_2 \cdot \| \int_{t=0}^{1} \Big( J_g(t \cdot r + (1-t) \cdot x_0) - J_g(x_0) \Big) \cdot (r - x_0)\ dt\, \|_2 \\
&\qquad \text{(by the definition of the spectral norm of } J_g(x_0)^{-1}) \\
&\leqslant \|J_g(x_0)^{-1}\|_2 \cdot \int_{t=0}^{1} \| \Big( J_g(t \cdot r + (1-t) \cdot x_0) - J_g(x_0) \Big) \cdot r - x_0 \|_2\ dt \\
&\qquad \text{(by the "triangle inequality" of the norm)} \\
&\leqslant \|J_g(x_0)^{-1}\|_2 \cdot \int_{t=0}^{1} \| \Big( J_g(t \cdot r + (1-t) \cdot x_0) - J_g(x_0) \Big) \|_2 \cdot \|r - x_0\|_2\ dt \\
&\qquad \text{(by the definition of the spectral norm of } J_g(t \cdot r + (1-t) \cdot x_0) - J_g(x_0)) \\
&\leqslant \alpha \cdot \int_{t=0}^{1} \beta \cdot \|(t \cdot r + (1-t) \cdot x_0) - x_0\|_2 \cdot \|r - x_0\|_2\ dt \\
&\quad \text{(by conditions } (i) \text{ and } (ii) \text{ of the theorem statement for each term, respectively)} \\
&= \alpha \cdot \beta \cdot \|r - x_0\|_2^2 \cdot \int_{t=0}^{1} t\ dt && \text{(as } (t \cdot r + (1-t) \cdot x_0) - x_0 = t \cdot (r - x_0)) \\
&= M \cdot \|r - x_0\|_2^2. && \text{(as the value of integral is } 1/2 \text{ and } M = \alpha\beta/2)
\end{aligned}$$

This concludes the proof. □

**Remark.** It is worth pointing out the conditions $(i)$ and $(ii)$ and the definition of $M$ in Theorem 6 are natural generalizations of the same concepts in Theorem 1 in the one-dimensional case. Recall that in Theorem 1 we defined

$$M := \sup_{y \in \{t \cdot r + (1-t) \cdot x_0 | t \in (0,1)\}} \left| \frac{g''(y)}{2 \cdot g'(x_0)} \right|.$$

In Theorem 6, we replaced $|g'(x_0)^{-1}|$ with the bound $\|J_g^{-1}(x_0)\|_2 \leqslant \alpha$ which is a direct analogue. But, instead of directly replacing $|g''(y)|$ with some high-dimensional analogue (which is effectively a 3d-tensor), we bounded the Lipschitzness constant of $J_g(\cdot)$ matrix on the interval $y$ is chosen from, namely, upper bounded the rate of change of $J_g$ which is an analogue of bounding the "derivative" of $J_g$ (the same as $g''$ being the function that determines the rate of change of $g'$ in the one-dimensional case).

Finally, we conclude this subsection by pointing out that we only provided Theorem 6 for completeness; for our application in this course, this theorem is *not* the "right" way of looking at Newton's method (because of its dependencies on the Euclidean norm). We will discuss this in more details in the next lecture but also refer the reader to [1, Chapter 9.5] book for a discussion on this topic.

## 1.3 Newton's Method for Unconstrained Convex Optimization

More specific to our applications in this course, Newton's method is an algorithm for solving unconstrained convex optimization problems of the form:

$$\min_{x \in \mathbb{R}^n} f(x),$$

for a convex function $f : \mathbb{R}^n \to \mathbb{R}$. This application is quite straightforward. Recall from Lecture 11 that the minimums of a differentiable convex function are the same as roots of the gradient of $f$, i.e., for $x^* \in \mathbb{R}^n$:

$$x^* \in \arg\min_x f(x) \iff \nabla f(x^*) = 0.$$

Thus, optimization of $f$ reduces to finding the roots of the multi-dimensional function $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$. This problem is precisely what the Newton's method solves! In particular, at any point $x_0$, a Newton's step for minimizing $f$ corresponds to updating to the new point:

$$x_1 = x_0 - \left(\nabla^2 f(x_0)\right)^{-1} \cdot \nabla f(x_0); \tag{4}$$

Here, $\nabla^2 f$ is the *Hessian* matrix of $f$, the matrix of second-order partial derivatives of $f$, or equivalently, the Jacobian of $\nabla f$.

The Newton's method thus lends itself naturally to the following iterative algorithm for (unconstrained) convex optimization:

**Algorithm:** Newton's Method for minimizing a convex function $f : \mathbb{R}^n \to \mathbb{R}$.

$(i)$ Start with some initial point $x_0 \in \mathbb{R}^n$.

$(ii)$ For $t = 0$ to $T$ iterations: update $x_{t+1} = x_t - \left(\nabla^2 f(x_t)\right)^{-1} \cdot \nabla f(x_t)$;

$(iii)$ Return $x_T$ as the answer.

We will postpone stating the precise bounds on this algorithm to the next lecture. Instead, we show an alternative way of deriving this algorithm which is more inline with its second-order optimization viewpoint.

### 1.3.1 A Second-Order Optimization Method

The Newton's method algorithm described above can be seen as a generalization of the gradient descent method which only updated $x_t$ in the (opposite) direction of the gradient, i.e., $x_{t+1} = x_t - \eta \cdot \nabla f(x_t)$.

Compared to gradient descent, the Newton's method is exploiting more information about the function, in particular its Hessian, which is a second-order information about $f$. Let us now see this more explicitly.

Suppose we are iteratively minimizing a convex function $f : \mathbb{R}^n \to \mathbb{R}$ and are currently at some point $x_t$. To decide our next step, we first *approximate* $f$ at the point $x_t$ by its *second-order* Taylor expansion to get the following function $\tilde{f} : \mathbb{R}^n \to \mathbb{R}$:

$$\tilde{f}(x) = f(x_t) + (x - x_t)^\top \cdot \nabla f(x_t) + \frac{1}{2} \cdot (x - x_t)^\top \cdot \nabla^2 f(x_t) \cdot (x - x_t).$$

We can now "pretend" that from now on $f$ is replaced by $\tilde{f}$ instead and thus optimize $\tilde{f}$ instead to get the point $x_{t+1}$, i.e., let

$$x_{t+1} = \arg\min_x \tilde{f}(x).$$

But since $\tilde{f}$ is a "quadratic" function, we can find its minimum easily by computing its gradient and finds its roots, i.e., compute

$$\nabla \tilde{f}(x) = \nabla f(x_t) + \nabla^2 f(x_t) \cdot (x - x_t).$$

Thus, we have $x_{t+1}$ is the point where $\nabla \tilde{f}(x_{t+1}) = 0$ which implies that

$$\nabla f(x_t) + \nabla^2 f(x_t) \cdot (x_{t+1} - x_t) = 0.$$

Reorganizing the terms then gives us

$$x_{t+1} = x_t - \left(\nabla^2 f(x_t)\right)^{-1} \cdot \nabla f(x_t).$$

But this is precisely the formula we have for our update rule in the Newton's method!

**Remark.** In the above formulation, we can think of each step of the gradient descent, say at a point $x_t$, as approximating $f$ by its *first-order* Taylor expansion at the point $x_t$ and then simply move in the direction that minimizes this function (the given approximate function will be linear and thus does not have an absolute minimum value). Newton's method on the other hand approximates $f$ by its second-order Taylor expansion. This typically results in (much) faster convergence (assuming the right set of properties) but in turns also require a much stronger *second-order oracle*, namely, access to the Hessian of $f$ not only its gradient.

# 2 A Quick Introduction to Interior Point Methods (IPMs)

In this section, we will introduce the intuition behind the Interior Point Methods for solving LPs, in particular, the path following IPMs. We postpone most of the details of these methods to the next lecture and will be informal in this part for the sake of intuition.

Recall that our goal is to solve the following standard LP formulation:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{subject to} \quad & Ax \leqslant b, \end{aligned}$$

with $m$ constraints of the form $\langle a_i , x \rangle \leqslant b_i$ in $(A, b)$. Throughout, we are going to assume that $A$ is a full-dimensional and feasible *polytope* (recall that these were common assumptions for our prior algorithms as well, in particular, the first one was also used by the Ellipsoid method, and the second by the Simplex; we also provided a way for lifting each assumption in the past which continue to work here as well).

At a high level, an IPM starts with a feasible solution somewhere "deep" inside the polytope and walks through an interior path of the feasible solution set $P$. It will never approach the boundary of $P$ until the very last step (Figure 2), hence the name "*interior-point*" method. This traversing of the interior points is done via solving a collection of intermediate *unconstrained* convex optimization problems (approximately) using the Newton's method.

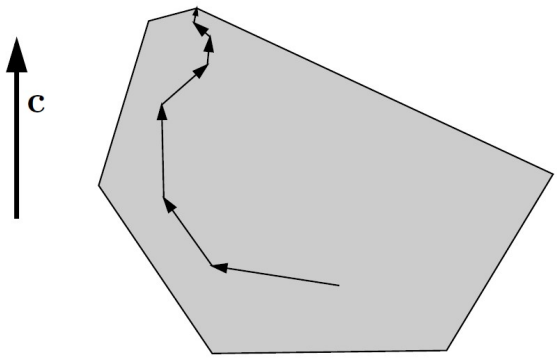


Figure 2: An illustration of interior point method. Here, the arrow for $c$ denotes the direction that optimizes the objective function (which is the opposite direction of the vector $c$).

## 2.1 Reducing Linear Programming to Unconstrained Convex Optimization

Let us denote the feasible polytope by $P := \{x \mid Ax \leqslant b\}$ and define $\text{int}(P)$ as the interior of $P$ and $\partial P$ as its boundary points. The optimization problem we would like to solve, namely,

$$\min_{x \in P} c^\top x$$

is a *constrained* convex optimization problem. Yet, to be able to apply Newton's method, we need to work with an unconstrained optimization problem. So, how do we use Newton method to solve this problem? An intuitive way is to *penalize* the violated constraints in the minimization objective:

$$\min_{x \in \mathbb{R}^n} \quad c^\top x + h(x),$$

where

$$h(x) = \begin{cases} 0 & \text{if } x \in P \\ +\infty & \text{otherwise} \end{cases}.$$

It is clear that the optimums of this new program coincide with that of the original LP. This is also now an unconstrained optimization problem. The problem however is that this new objective function is not convex, nor even continuous. Thus, we are no longer in the realm of convex optimization and cannot use our prior ideas. The next idea for fixing this is to use the notion of a **barrier** function $F(x)$ in place of $h(x)$, which, informally speaking, has the following roles:

(*i*) $F(x)$ should be defined on the interior of $P$, i.e., $F : \text{int}(P) \to \mathbb{R}$, and be (strongly) convex[1];

(*ii*) $F(x)$ should approach infinity the closer we get to the boundary of $P$, i.e., $\lim_{x \to \partial P} F(x) = +\infty$.

To make things concrete, we are going to pick a standard choice of a barrier function (but it would help to "see" this $F(x)$ abstractly for now as just satisfying the conditions).

[1]We will define strong convexity formally in the next lecture but for now simply think of it as a convex function which is "far from" being a line on any interval – in other words, the tangent line on the function at any point is strictly below the function.

**Definition 7.** For any polytope $P := \{x \mid Ax \leqslant b\}$ we define the **logarithmic barrier** $F : \text{int}(P) \to \mathbb{R}$:

$$F(x) = -\sum_{i=1}^{m} \ln\left(b_i - a_i^\top x\right);$$

This way, as long as $x \in \text{int}(P)$, we have $a_i^\top x < b_i$ for all $i \in [m]$, so $F(x)$ is well-defined and attains a real value. But if $x$ approaches $\partial F$ by having $a_i^\top x \approx b_i$, then the corresponding term in $F(x)$ becomes $-\ln(\approx 0)$ which is $+\infty$. More formally, we indeed have that $\lim_{x \to \partial P} F(x) = +\infty$ as desired (we postpone proving the strong convexity of $F$ to the next lecture).

Thus, at this point our optimization becomes

$$\min_{x \in \mathbb{R}^n} c^\top x + F(x),$$

which is indeed an unconstrained convex optimization problem. But now there is a different problem: the optimum of this new optimization problem can be quite far away from the original LP optimum solution we would like to solve; in fact, quite problematically, while the optimum solutions of the LP happen at the boundary of $P$ (recall that we proved earlier in the course that the optimum solutions happen on vertices of the polytope $P$), all the boundary points of $P$ have a value of $+\infty$ and thus can never be an optimum solution to the above program. This gives rise to the *path following* algorithm that reduces solving LPs not to one unconstrained convex optimization problem, but rather a whole family of them.

## 2.2 A Path Following IPM

Define the following (uncountable) family of functions:

$$\mathcal{F} := \left\{ f_\eta(x) = \eta \cdot c^\top x + F(x) \mid \eta \geqslant 0 \right\}.$$

Notice that any function $f_\eta \in \mathcal{F}$ is again defined from $\text{int}(P) \to \mathbb{R}$ and is basically a rescaled version of the previous objective plus barrier function we defined in the previous subsection. For any $\eta \geqslant 0$, define:

$$x_\eta^* := \arg\min_x f_\eta(x);$$

we will prove in the next lecture that since $f_\eta$ is strongly convex, $x_\eta^\star$ is unique. Moreover, by increasing the value of $\eta$, we are prioritizing the role of LP objective over the barrier function so that we eventually have

$$\lim_{\eta \to \infty} x_\eta^* = x^*,$$

where $x^*$ is the optimum value of the original LP. On the other hand, for $\eta = 0$, $x_0^*$ has nothing to do with the objective function and is instead a point inside $P$ which is in some sense "furthest away" from all boundaries defined by the "force" introduced by $F(x)$ from each constraint of the LP. This point, namely, $x_0^*$ is typically called the **analytical center** of $P$[2]. The **central path** now is the set

$$\Gamma_{center} := \left\{ x_\eta^* \mid \eta \geqslant 0 \right\},$$

which starts from the analytical center $x_0^*$ of the polytope $P$ and converges to the optimum solution of LP $x^*$ in the limit (while we do not prove this explicitly, this path is indeed continuous).

The path following IPMs then try to follow the central path *approximately* by starting with a pair $(x_1, \eta_1)$ with $\eta_1 > 0$ but also possibly close to 0, and $x_1$ being *near* $x_{\eta_1}^*$ (the algorithm starts sufficiently close to the analytical center so that we can show how to find such a pair – this is also a topic for the next lecture).

[2]This should not be confused with the *center of gravity* of $P$ defined for the Ellipsoid algorithm; unlike the center of gravity, the analytical center is somewhat "easier" to compute at least approximately.

The algorithm then iteratively updates $(x_t, \eta_t)$ to $(x_{t+1}, \eta_{t+1})$ by increasing $\eta_t$ *slightly* to get to $\eta_{t+1}$, while updating $x_{t+1}$ correspondingly to become *near* $x^*_{\eta_{t+1}}$. In particular, the algorithm maintains the invariant that at any iteration $t$, $x_t$ is near $x^*_{\eta_t}$. At the end of $T$ iterations, once the algorithm has a pair $(x_T, \eta_T)$ for a sufficiently large $\eta_T$, we can say that $x_T$ is sufficiently close to $x^*$ itself, so that a final *rounding* approach can solve the problem.

There are several questions that needs to be answered here:

1. What does it mean for $x_t$ to be "near" $x^*_{\eta_t}$?
2. At what rate we can "slightly" update $\eta_t$ to obtain $\eta_{t+1}$?
3. And, most importantly, how can we obtain $x_{t+1}$ from $x_t, \eta_t$, and $\eta_{t+1}$?

The short answer to all these questions is the Newton's method! The general idea is the following. Suppose we already have $x_t$ as a (good approximate) minimizer of $f_{\eta_t}$. Now, let us slightly increase $\eta_t$ to $\eta_{t+1}$ by some factor, i.e., $\eta_{t+1} = (1+\gamma) \cdot \eta_t$ for some small $\gamma > 0$. The function $f_{\eta_{t+1}}$ is not "that different" from $f_{\eta_t}$ so we would expect that $x_t$ is a *good starting point* for optimizing $f_{\eta_{t+1}}$. So, we are now going to run Newton's method with initial point for the unconstrained optimization of $f_{\eta_{t+1}}$ to optimize this to get $x_{t+1}$ as a proxy for $x^*_{\eta_{t+1}}$ which is the optimizer of $f_{\eta_{t+1}}$. Thus, the key point in all of these is to make sure that we can place $x_t$ in the *quadratic convergence* regime of $f_{\eta_{t+1}}$. This will be the key issue that determines how large we can update $\gamma$ and how much we require $x_t$ to be near to $x^*_{\eta_t}$. There are also further issues that we need to address like how to start with the points $(x_1, \eta_1)$ and when to end this process; and how to round the final answer to a true optimum solution $x^*$ for the original LP.

We formalize this algorithm in the next lecture (at least partially). For now, we only mention that we will be able to eventually to set the parameters so that the entire IPM needs

$$O\left(\sqrt{m} \cdot \ln\left(\frac{m}{\epsilon \cdot \eta_0}\right)\right)$$

iterations to output an answer with objective value within additive $\varepsilon$ factor of the optimum solution. Moreover, each iteration only involves $O(1)$ Newton's step, each of which requires solving a system of linear equations (primarily, plus some low cost bookkeeping). Finally, as we pointed out earlier in the course for the Ellipsoid algorithm, we can take $\varepsilon$ sufficiently small so that $\log(1/\varepsilon)$ is still polynomial in the input size, while an $\varepsilon$-additive error can be rounded to an optimal solution. We can pick $\eta_0$ also in a similar manner with a similar logic so that $\log(1/\eta_0)$ is polynomial in the input size. Putting all these together then gives us a polynomial time algorithm for solving any linear program.

# References

[1] N. K. Vishnoi. *Algorithms for convex optimization.* Cambridge University Press, 2021. 1, 7