Algorithms for convex optimization

Michal Kočvara

Institute of Information Theory and Automation

Academy of Sciences of the Czech Republic

and

Czech Technical University

kocvara@utia.cas.cz

http://www.utia.cas.cz/kocvara

Convex programs

A general problem of *mathematical programming*:

$$\boxed{\min f(x) \quad \text{subject to} \quad x \in X \subset \mathbb{R}^n} \qquad (\mathbf{MP})$$

where

n is the *design dimension* of the problem;

$f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*;

$X \subset \mathbb{R}^n$ is the *feasible domain* of the problem.

Assume:

- the function f and the feasible domain X are *convex*;
- the feasible domain X is defined by *convex functions* $g_i : \mathbb{R}^n \to \mathbb{R}, i = 1, \ldots, m$:

$$X := \{x \in \mathbb{R}^n \mid g_i(x) \leq 0,\ i = 1, \ldots, m\}\,.$$

Convex programs

A mathematical program satisfying the above assumption is called *convex mathematical program*:

$$\min f(x) \quad \text{subject to} \quad g_i(x) \leq 0,\ i = 1, \ldots, m. \qquad (\mathbf{CP})$$

Why are we interested in this class of optimization problems? That is because:

(i) Convex programs are *computationally tractable*: there exist numerical methods which efficiently solve every convex program satisfying "mild" additional assumption;

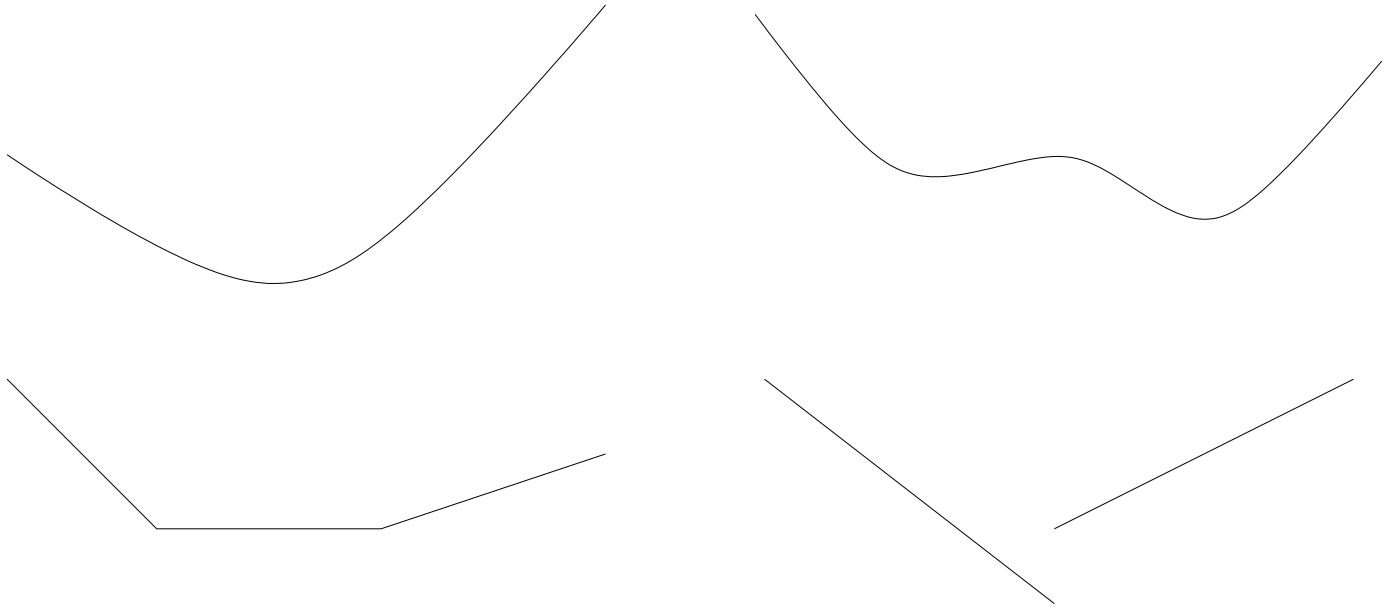
(ii) In contrast, no efficient universal methods for nonconvex mathematical programs are known.

Convex functions

A function $h : \mathbb{R}^n \to \mathbb{R}$ is convex when

$$\forall x, x' \; \forall \lambda \in [0,1] : \qquad h(\lambda x + (1-\lambda)x') \leq \lambda h(x) + (1-\lambda)h(x')$$

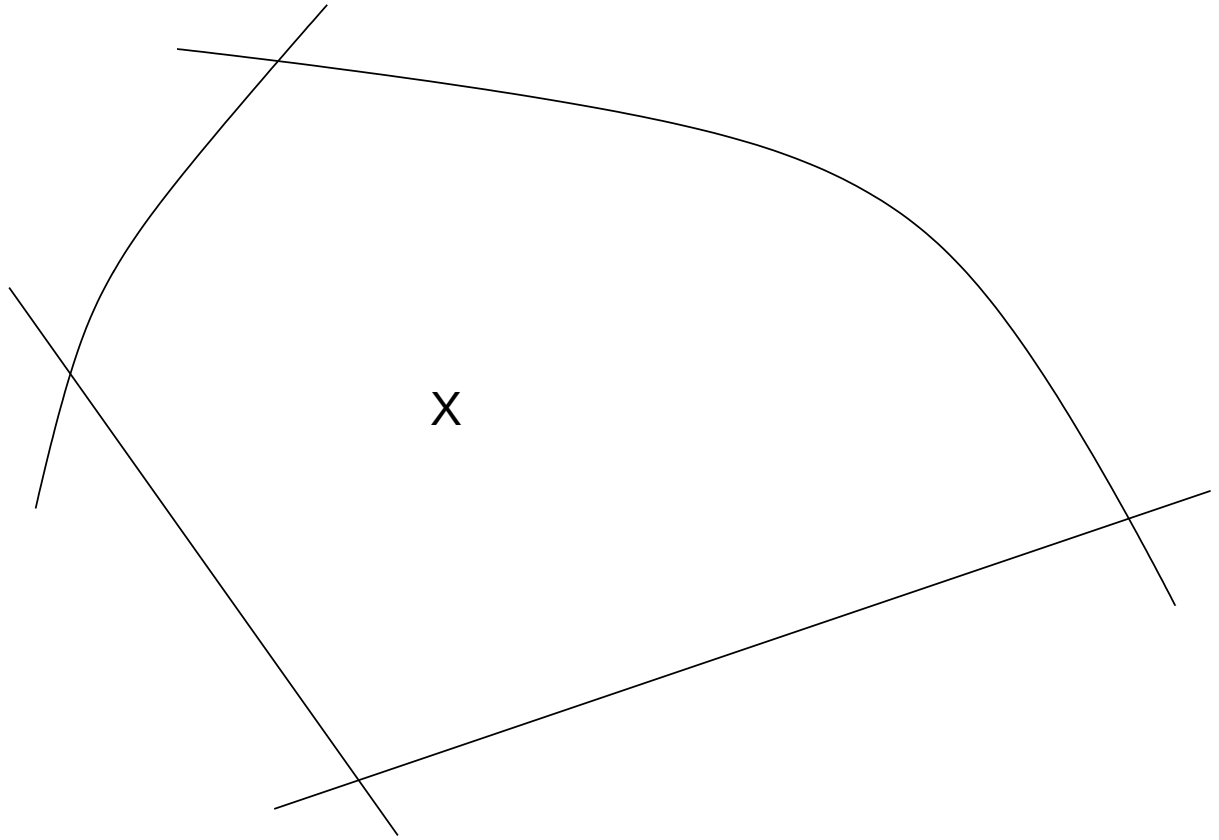
Convex (a) and nonconvex (b) functions in $\mathbb{R}^1$:



(a) (b)

Convex feasible region

Convex feasible region described by convex g_i's:



Method of Newton

The “simplest” convex program— problem without constraints and with a strongly convex objective:

$$\min f(x) \qquad \text{(UCP)};$$

strongly convex—the Hessian matrix $\nabla^2 f(x)$ is positive definite at every point x and that $f(x) \to \infty$ as $\|x\|_2 \to \infty$.

Method of choice: Newton’s method. The idea: at every iterate, we approximate the function f by a quadratic function, the second-order Taylor expansion:

$$f(x) + (y-x)^T \nabla f(x) + \frac{1}{2}(y-x)^T \nabla^2 f(x)(y-x).$$

The next iterate—by minimization of this quadratic function.

Method of Newton

Given a current iterate x, compute the gradient $\nabla f(x)$ and Hessian $H(x)$ of f at x.
Compute the direction vector

$$d = -H(x)^{-1}\nabla f(x).$$

Compute the new iterate

$$x_{\mathbf{new}} = x + d.$$

The method is

- *extremely fast* whenever we are "close enough" to the solution x^*.
  Theory: the method is *locally quadratically convergent*, i.e.,

$$\|x_{\mathbf{new}} - x^*\|_2 \leq c\|x - x^*\|_2^2,$$

  provided that $\|x - x^*\|_2 \leq r$, where r is small enough.
- *rather slow* when we are *not* "close enough" to x^*.

Interior-point methods

Transform the "difficult" constrained problem into an "easy" unconstrained problem, or into a sequence of unconstrained problems.

Once we have an unconstrained problem, we can solve it by Newton's method.

The idea is to use a barrier function that sets a barrier against leaving the feasible region. If the optimal solution occurs at the boundary of the feasible region, the procedure moves from the interior to the boundary, hence *interior-point* methods.

The barrier function approach was first proposed in the early sixties and later popularised and thoroughly investigated by Fiacco and McCormick.

Classic approach

$$\min f(x) \quad \text{subject to} \quad g_i(x) \leq 0,\ i = 1, \ldots, m. \qquad (\mathrm{CP})$$

We introduce a *barrier function* B that is nonnegative and continuous over the region $\{x \mid g_i(x) < 0\}$, and approaches infinity as the boundary of the region $\{x \mid g_i(x) \leq 0\}$ is approached from the interior.

(CP) $\longrightarrow$ a one-parametric family of functions generated by the objective and the barrier:

$$\Phi(\mu; x) := f(x) + \mu B(x)$$

and the corresponding unconstrained convex programs

$$\min_x \Phi(\mu; x);$$

here the *penalty parameter* μ is assumed to be nonnegative.

Classic approach

The idea behind the barrier methods is now obvious:
We start with some μ (say $\mu = 1$) and solve the unconstrained auxiliary problem. Then we decrease μ by some factor and solve again the auxiliary problem, and so on.

- The auxiliary problem has a *unique solution* $x(\mu)$ for any $\mu > 0$.
- The *central path*, defined by the solutions $x(\mu),\ \mu > 0$, is a smooth curve and its limit points (for $\mu \to 0$) belong to the set of optimal solutions of (CP).

Example

One of the most popular barrier functions—*(Frisch's)* *logarithmic barrier function*,

$$B(x) = -\sum_{i=1}^{m} \log(-g_i(x)).$$

Consider a one-dimensional CP

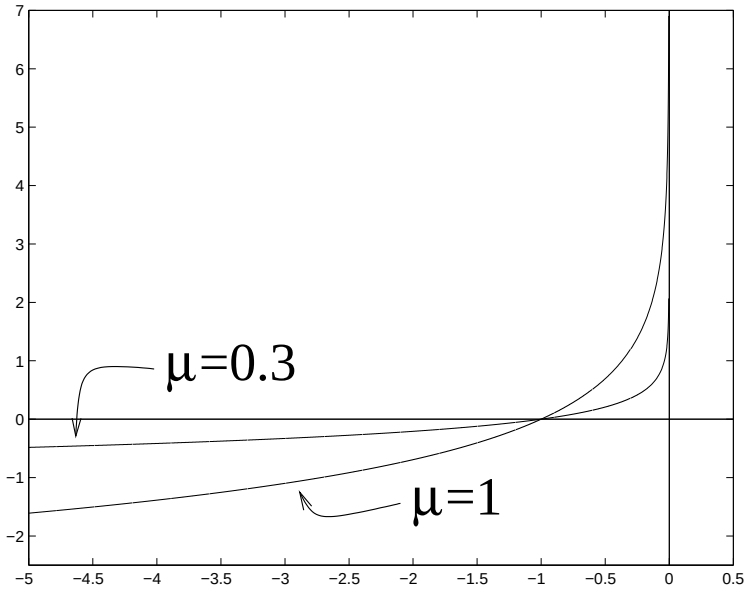
$$\min -x \quad \text{subject to} \quad x \leq 0.$$

Auxiliary problem:

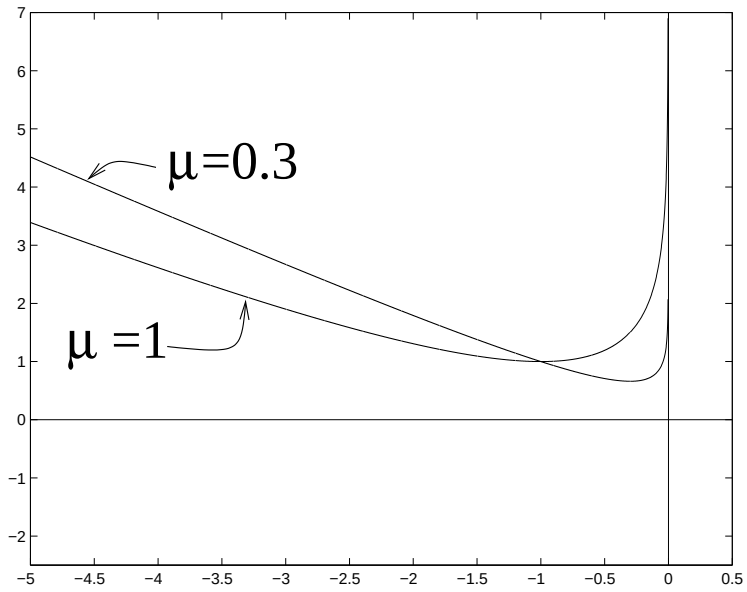
$$\Phi(\mu; x) := -x + \mu B(x)$$

Example

Barrier function (a) and function Φ (b) for $\mu = 1$ and $\mu = 0.3$:



(a)



(b)

Example

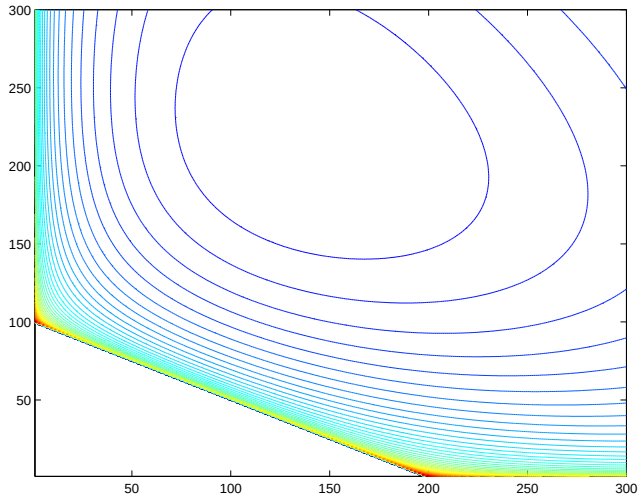
$$\begin{aligned} & \min\ (x_1 + x_2) \\ \text{s. t.} & \\ & -x_1 - 2x_2 + 2 \leq 0 \\ & -x_1 \leq 0,\ -x_2 \leq 0 \end{aligned}$$

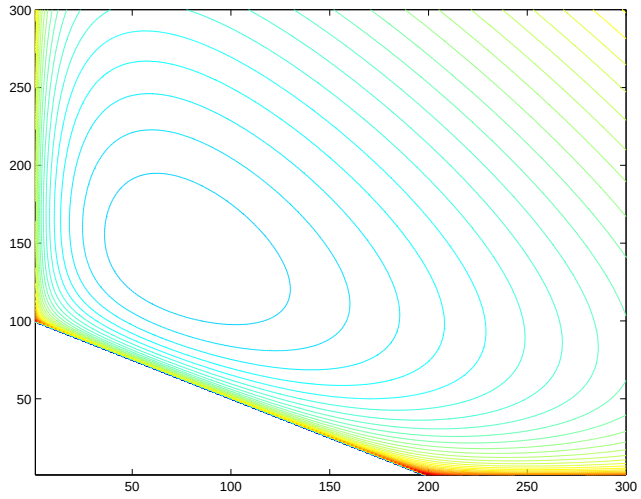
The auxiliary function

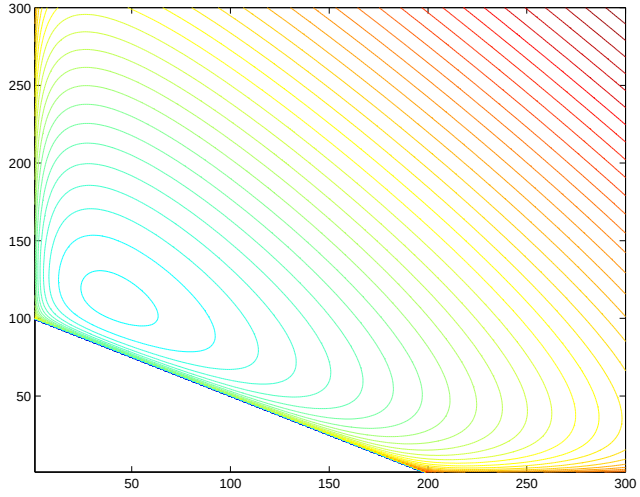
$$\Phi = x_1 + x_2 - \mu\left[\log(x_1 + 2x_2 - 2) + \log(x_1) + \log(x_2)\right]$$

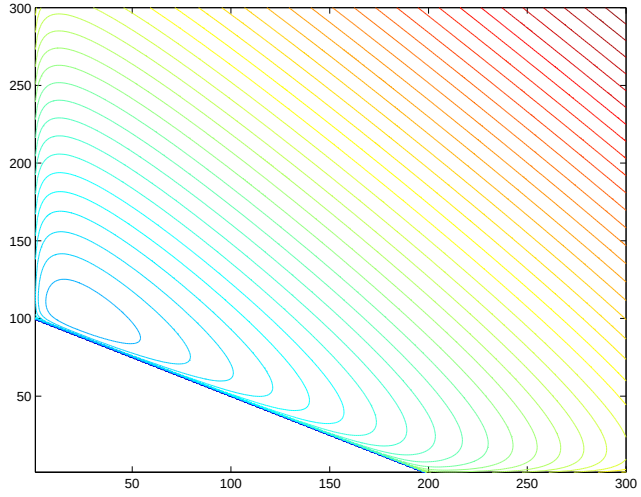
Example

Level lines of the function Φ for $\mu = 1,\ 0.5,\ 0.25,\ 0.125$:

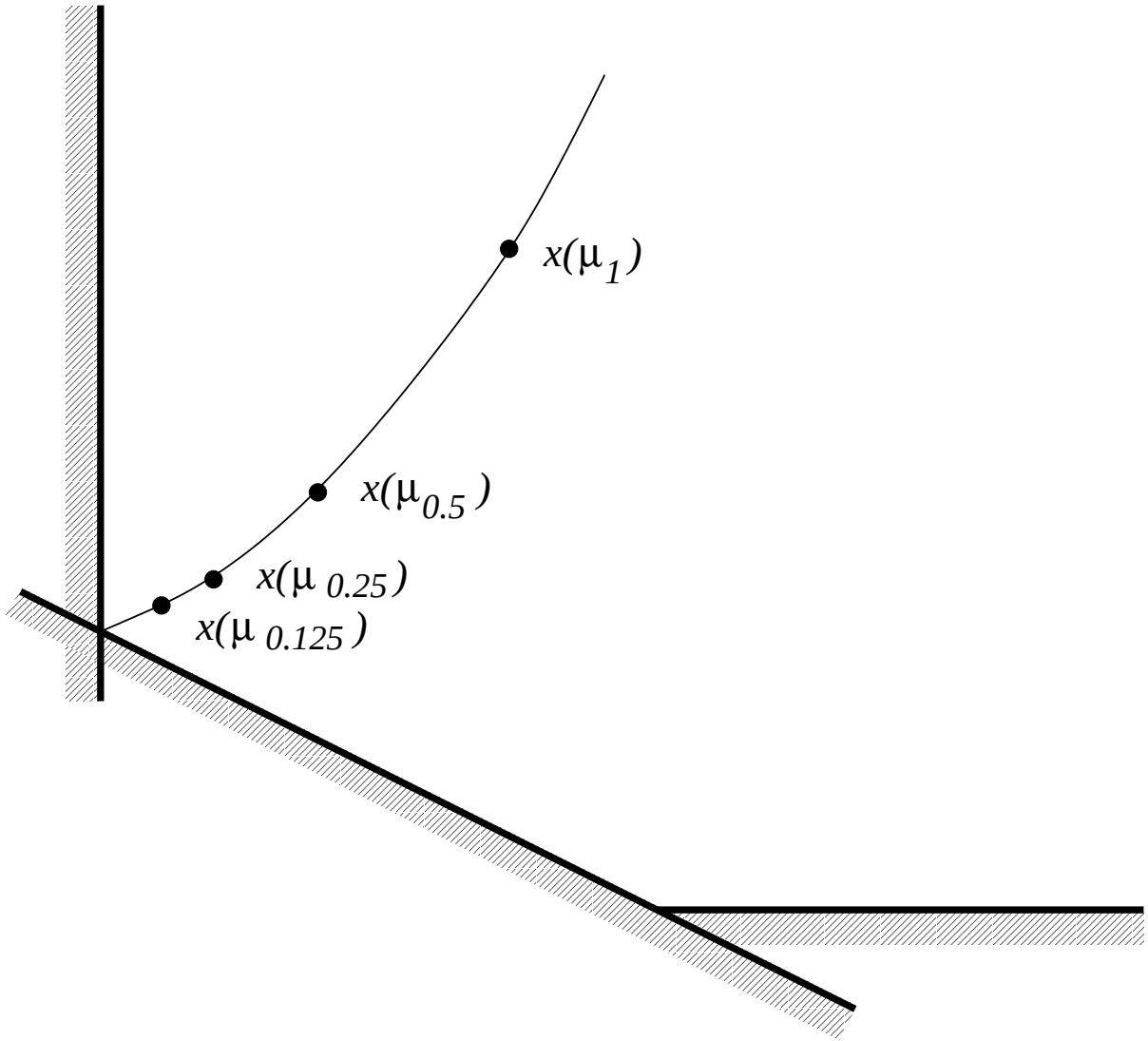








Central path



The "algorithm"

At i-th step, we are at a point $x(\mu_i)$ of the central path.

- decrease a bit μ_i, thus getting a new "target point" $x(\mu_{i+1})$ on the path;
- approach the new target point $x(\mu_{i+1})$ by running the Newton method started *at our current iterate* x_i.

Hope: $x(\mu_i)$ is in the region of quadratic convergence of the Newton method approaching $x(\mu_{i+1})$.

Hence, following the central path, Newton's method is always efficient (always in the region of quadratic convergence)

Theory and practice

THEORY: Under some mild technical assumption, the method converges to the solution of (CP):

$$x(\mu) \to x^* \qquad \text{as} \quad \mu \to 0.$$

PRACTICE: *disappointing*.
The method *may* have serious numerical difficulties.

- The idea to stay on the central path, and thus to solve the auxiliary problems exactly, is too restrictive.
- The idea to stay all the time in the region of quadratic convergence of the Newton method may lead to extremely short steps.
- How to guarantee *in practice* that we are "close enough"? The theory does not give any quantitative results.
- If we take longer steps (i.e., decrease μ more rapidly), then the Newton method may become inefficient and we may even leave the feasible region.

Modern approach—Brief history

The “classic” barrier methods—60’s and 70’s. Due to their disadvantages, practitioners lost interest soon.

Linear programming:
Before 1984 linear programs solved *exclusively* by simplex method (Danzig ’47)

- good practical performance
- bad theoretical behaviour
  (Examples with exponential behaviour)

Looking for *polynomial-time* method

1979 Khachian: ellipsoid method

- polynomial-time (theoretically)
- bad practical performance

1984 Karmakar: polynomial-time method for LP
reported 50-times faster than simplex

1986 Gill et al.: Karmakar = classic barrier method

Asymptotic vs. Complexity analysis

Asymptotic analysis

Classic analysis of the Newton and barrier methods:
uses terms like “sufficiently close”, “sufficiently small”, “close enough”, “asymptotic quadratic convergence”.

Does not give any quantitative estimates like:

- how close is “sufficiently close” in terms of the problem data?
- how much time (operations) do we need to reach our goal?

Asymptotic vs. Complexity analysis

Complexity analysis

Answers the question:

Given an instance of a generic problem and a desired accuracy, how many arithmetic operations do we need to get a solution?

The classic theory for barrier methods does not give a single information in this respect.

Moreover, from the complexity viewpoint,

- Newton's method has no advantage to first-order algorithms (there is no such phenomenon as "local quadratic convergence");
- constrained problems have the same complexity as the unconstrained ones.

Modern approach (Nesterov-Nemirovski)

It appears that all the problems of the classic barrier methods come from the fact that we have too much freedom in the choice of the penalty function B.

Nesterov and Nemirovski (SIAM, 1994):

1. There is a class of "good" (*self-concordant*) barrier functions. Every barrier function B of this type is associated with a real parameter $\theta(B) > 1$.
2. If B is self-concordant, one can specify the notion of "closeness to the central path" and the policy of updating the penalty parameter μ in the following way. If an iterate x_i is close (in the above sense) to $x(\mu_i)$ and we update the parameter μ_i to μ_{i+1}, then in a *single* Newton step we get a new iterate x_{i+1} which is close to $x(\mu_{i+1})$. In other words, after every update of μ we can perform only one Newton step and stay close to the central path. Moreover, points "close to the central path" belong to the interior of the feasible region.

Modern approach (Nesterov-Nemirovski)

3. The penalty updating policy can be defined in terms of problem data:

$$\frac{1}{\mu_{i+1}} = \left(1 + \frac{0.1}{\sqrt{\theta(B)}}\right) \frac{1}{\mu_i};$$

this shows, in particular, that reduction factor is independent on the size of μ, the penalty parameter decreases linearly.

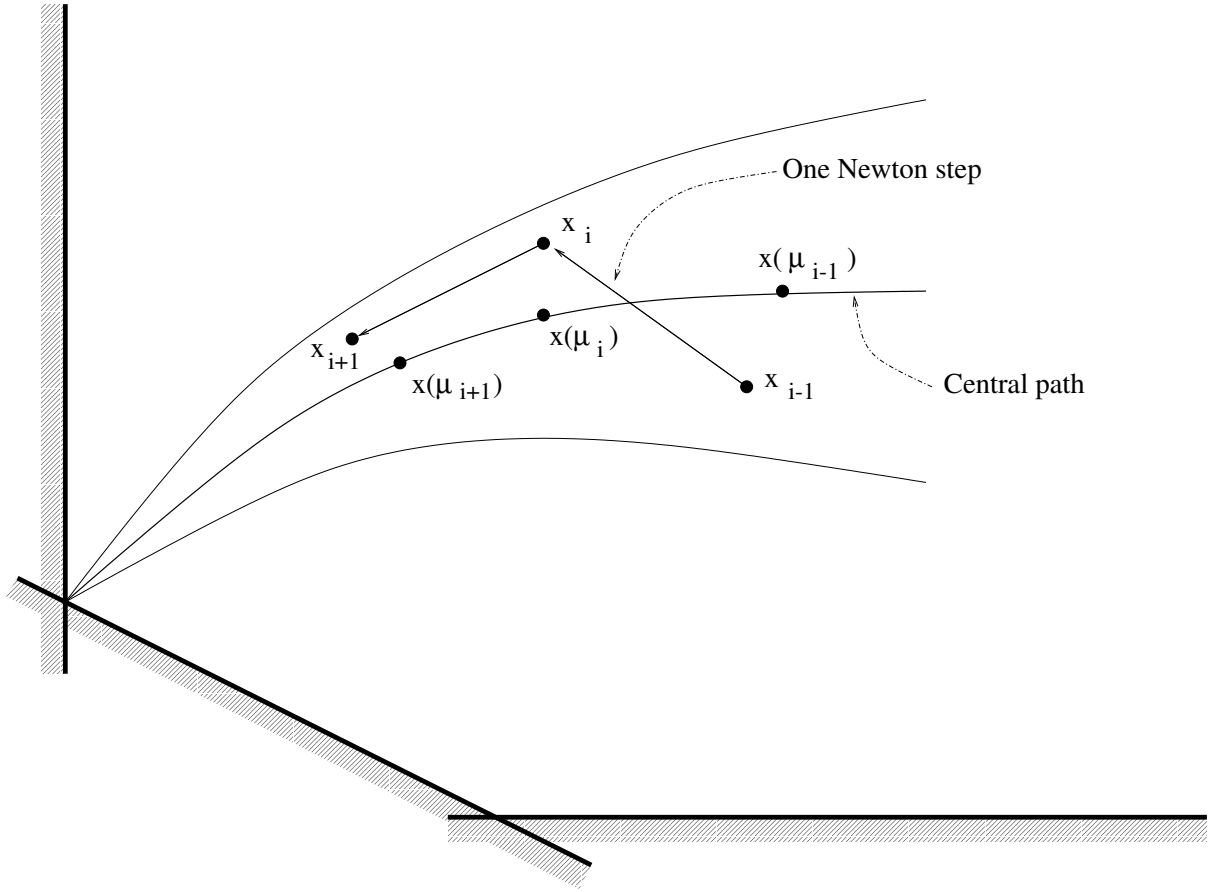
4. Assume that we are close to the central path (this can be realized by certain initialization step). Then every $O(\sqrt{\theta(B)})$ steps of the algorithm improve the quality of the generated approximate solutions by an absolute constant factor. In particular, we need at most

$$O(1)\sqrt{\theta(B)}\log\left(1 + \frac{\mu_0\theta(B)}{\varepsilon}\right)$$

to generate a strictly feasible ε-solution to (CP).

Modern approach (Nesterov-Nemirovski)

Staying close to the central path in a single Newton step:



The class of self-concordant function is sufficiently large and contains many popular barriers, in particular the logarithmic barrier function.