IOE 511/Math 652: Continuous Optimization Methods, Section 1

Marina A. Epelman

Fall 2007

In turn, I would like to thank Robert M. Freund, from whose courses *15.084: Nonlinear Programming* and *15.094: Systems Optimization: Models and Computation* at MIT these notes borrow in many ways.

Contents

1 Examples of nonlinear programming problems formulations

1.1 Forms and components of a mathematical programming problems

A *mathematical programming problem* or, simply, a *mathematical program* is a mathematical formulation of an optimization problem.

Unconstrained Problem:

$$\begin{array}{lll} \text{(P)} & \min_x & f(x) \\ & \text{s.t.} & x \in X, \end{array}$$

where $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, $f(x) : \mathbb{R}^n \to \mathbb{R}$, and X is an open set (usually $X = \mathbb{R}^n$).[1]

Constrained Problem:

$$\begin{array}{lll} \text{(P)} & \min_x & f(x) \\ & \text{s.t.} & g_i(x) \leq 0 \quad i = 1, \ldots, m \\ & & h_i(x) = 0 \quad i = 1, \ldots, l \\ & & x \in X, \end{array}$$

where $g_1(x), \ldots, g_m(x), h_1(x), \ldots, h_l(x) : \mathbb{R}^n \to \mathbb{R}$.

Let $g(x) = (g_1(x), \ldots, g_m(x))^T : \mathbb{R}^n \to \mathbb{R}^m$, $h(x) = (h_1(x), \ldots, h_l(x))^T : \mathbb{R}^n \to \mathbb{R}^l$. Then (P) can be written as

$$\begin{array}{lll} \text{(P)} & \min_x & f(x) \\ & \text{s.t.} & g(x) \leq 0 \\ & & h(x) = 0 \\ & & x \in X. \end{array} \tag{1}$$

Some terminology: Function $f(x)$ is the *objective function*. Restrictions "$h_i(x) = 0$" are referred to as *equality constraints*, while "$g_i(x) \leq 0$" are *inequality constraints*. Notice that we do not use constraints in the form "$g_i(x) < 0$"!

A point x is *feasible* for (P) if it satisfies all the constraints. (For an unconstrained problem, $x \in X$.) The set of all feasible points forms the *feasible region*, or *feasible set* (let us denote it by S). The goal of an optimization problem in minimization form, as above, is to find a feasible point $\bar{x}$ such that $f(\bar{x}) \leq f(x)$ for any other feasible point x.

1.2 Markowitz portfolio optimization model

Suppose one has the opportunity to invest in n assets. Their future returns are represented by random variables, $R_1, \ldots, R_n$, whose expected values and covariances, $\mathrm{E}[R_i]$, $i = 1, \ldots, n$ and $\mathrm{Cov}(R_i, R_j)$, $i, j = 1, \ldots, n$, respectively, can be estimated based on historical data and, possibly, other considerations. At least one of these assets is a risk-free asset.

Suppose x_i, $i = 1, \ldots, n$, are the fractions of your wealth allocated to each of the assets (that is, $x \geq 0$ and $\sum_{i=1}^n x_i = 1$). The return of the resulting portfolio is a random variable $\sum_{i=1}^n x_i R_i$

[1] BSS uses $(\cdot)^T$ notation for transpose.

with mean $\sum_{i=1}^n x_i \mathrm{E}[R_i]$ and variance $\sum_{i=1}^n \sum_{j=1}^n x_i x_j \mathrm{Cov}(R_i, R_j)$. A portfolio is usually chosen to optimize some measure of a tradeoff between the expected return and the risk, such as

$$\begin{array}{ll} \max & \sum_{i=1}^n x_i \mathrm{E}[R_i] - \mu \sum_{i=1}^n \sum_{j=1}^n x_i x_j \mathrm{Cov}(R_i, R_j) \\ \text{s.t.} & \sum_{i=1}^n x_i = 1 \\ & x \geq 0, \end{array}$$

where $\mu > 0$ is a (fixed) parameter reflecting the investor's preferences in the above tradeoff. Since it is hard to assess anybody's value of μ, the above problem can (and should) be solved for a variety of values of μ, thus generating a variety of portfolios on the *efficient frontier*.

1.3 Least squares problem (parameter estimation)

Applications in model constructions, statistics (e.g., linear regression), neural networks, etc.

We consider a linear measurement model, i.e., we stipulate that an (output) quantity of interest $y \in \mathbb{R}$ can be expressed as a linear function $y \approx a^T x$ of input $a \in \mathbb{R}^n$ and model parameters $x \in \mathbb{R}^n$. Our goal is to find the vector of parameters x which provide the "best fit" for the available set of input-output pairs $(a_i, y_i),\ i = 1, \ldots, m$. If "fit" is measured by sum of squared errors between estimated and measured outputs, solution to the following optimization problem

$$\begin{array}{lll} \min\limits_{x \in \mathbb{R}^n} & \sum_{i=1}^m (v_i)^2 & = \min_{x \in \mathbb{R}^n} \sum_{i=1}^m (y_i - a_i^T x)^2 \quad = \min_{x \in \mathbb{R}^n} \|Ax - y\|_2^2, \\ \text{s.t.} & v_i = y_i - a_i^T x, \quad i = 1, \ldots, m & \end{array}$$

provides the best fit. Here, A is the matrix with rows a_i^T.

1.4 Maximum likelihood estimation

Consider a family of probability distributions $p_x(\cdot)$ on $\mathbb{R}$, parameterized by vector $x \in \mathbb{R}^n$. When considered as a function of x for a particular observation of a random variable $y \in \mathbb{R}$, the function $p_x(y)$ is called the *likelihood function*. It is more convenient to work with its logarithm, which is called the *log-likelihood function*:

$$l(x) = \log p_x(y).$$

Consider the problem of estimating the value of the parameter vector x based on observing one sample y from the distribution. One possible method, *maximum likelihood (ML) estimation*, is to estimate x as

$$\hat{x} = \mathrm{argmax}_x\, p_x(y) = \mathrm{argmax}_x\, l(x),$$

i.e., to choose as the estimate the value of the parameter that maximizes the likelihood (or the log-likelihood) function for the observed value of y.

If there is prior information available about x, we can add constraint $x \in C \subseteq \mathbb{R}^n$ explicitly, or impose it implicitly, by redefining $p_x(y) = 0$ for $x \notin C$ (note that in that case $l(x) = -\infty$ for $x \notin C$).

For m iid samples $(y_1, \ldots, y_m)$, the log-likelihood function is

$$l(x) = \log(\prod_{i=1}^m p_x(y_i)) = \sum_{i=1}^m \log p_x(y_i).$$

The ML estimation is thus an optimization problem:

$$\max l(x) \text{ subject to } x \in C.$$

For example, returning to the linear measurement model $y = a^T x + v$, let us now assume that the error v is iid random noise with density $p(v)$. If there are m measurement/output pairs (a_i, y_i) available, then the likelihood function is

$$p_x(y) = \prod_{i=1}^{m} p(y_i - a_i^T x),$$

and the log-likelihood function is

$$l(x) = \sum_{i=1}^{m} \log p(y_i - a_i^T x).$$

For example, suppose the noise is Gaussian (or Normal) with mean 0 and standard deviation σ. Then $p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{z^2}{2\sigma^2}}$ and the log-likelihood function is

$$l(x) = -\frac{1}{2}\log(2\pi\sigma) - \frac{1}{2\sigma^2}\|Ax - y\|_2^2.$$

Therefore, the ML estimate of x is $\arg\min_x \|Ax - y\|_2^2$, the solution of the least squares approximation problem.

1.5 Cantilever beam design

Consider the design of a cantilever beam of length l and density ρ whose height $x_1 \geq 0.1$ inch and width $x_2 \geq 0.1$ inch are to be selected to minimize total volume, which is proportional to lx_1x_2. The displacement of the beam under load P should not exceed a pre-specified amount δ. The displacement is given by $\frac{4Pl^3}{Yx_1x_2^3}$, where $Y > 0$ is the Young's modulus of the material. Therefore, the design problem can be formulated as follows:

$$\begin{array}{ll} \min_{x_1,\, x_2} & lx_1x_2 \\ \text{s.t.} & \frac{\delta Y}{4l^3}x_1x_2^3 - P \geq 0 \\ & x_1 \geq 0.1,\ x_2 \geq 0.1. \end{array} \tag{2}$$

In practice, however, Y (property of the material) and P (load on the beam) are not known a priori, or exactly. One common way to account for this uncertainty in the model is to view these parameters as random variables. Then the displacement of the beam is also a random variable, and the constraint "displacement of the beam does not exceed δ" is replaced with "with high probability, displacement of the beam does not exceed δ. This leads to the following optimization problem, which is commonly referred to as the *chance-constrained* problem:

$$\begin{array}{ll} \min & lx_1x_2 \\ \text{s.t.} & \text{Pr}.\left(\frac{\delta Y}{4l^3}x_1x_2^3 - P \geq 0\right) \geq \alpha \\ & x_1 \geq 0.1,\ x_2 \geq 0.1, \end{array} \tag{3}$$

where $\alpha \in [0,1]$ is a parameter selected by the designer to indicate the desired reliability. (Higher values of α correspond to greater probability that the condition on displacement will be met.)

Reddy, Grandhi and Hopkins[2] analyze this problem with

- $l = 30$ inches
- $\delta = 0.15$ inches
- Y and P are independent random variables
- Y is Gaussian with mean $\mu_Y = 3 \times 10^7$ psi and standard deviation $\sigma_Y = 3 \times 10^6$ psi
- P is Gaussian with mean $\mu_P = 400$ lbs and standard deviation $\sigma_P = 120$ lbs

Let us define random variable $R(x) = \frac{\delta Y}{4l^3} x_1 x_2^3 - P$. Then $R(x)$ is Gaussian with mean and variance

$$\mu(x) = \frac{\delta \mu_Y}{4l^3} x_1 x_2^3 - \mu_P, \ \sigma(x)^2 = \frac{\delta^2 \sigma_Y^2}{16 l^6} x_1^2 x_2^6 + \sigma_P^2.$$

Substituting parameter values into these expressions, we obtain:

$$\mu(x) = \frac{25}{3}(5 x_1 x_2^3 - 48), \ \sigma(x) = \frac{5}{6}\sqrt{25 x_1^2 x_2^6 + (144)^2}.$$

We also define

$$\rho(x) = \frac{\mu(x)}{\sigma(x)} = 10 \, \frac{5 x_1 x_2^3 - 48}{\sqrt{25 x_1^2 x_2^6 + (144)^2}}. \tag{4}$$

The chance constraint of the above formulation can be re-written as

$$\Pr.\{R(x) \geq 0\} \geq \alpha \ \Leftrightarrow \Phi(\rho(x)) \geq \alpha \ \Leftrightarrow \rho(x) \geq \Phi^{-1}(\alpha).$$

Here, $\Phi(\cdot)$ is the CDF of a standard Gaussian random variable, and the second equivalence follows by its monotonicity. Let $\beta := \Phi^{-1}(\alpha)$. Since we are interested in large values of α, β is going to assume positive values. Therefore, by squaring the expression of $\rho(x)$ given in the previous paragraph, we can re-state the chance constraint as the following two inequalities:

$$\mu(x) \geq 0, \ \mu(x)^2 - \beta^2 \sigma(x)^2 \geq 0.$$

Substituting expressions for $\mu(x)$ and $\sigma(x)$, and letting $\gamma = \beta^2/100$, these become:

$$5 x_1 x_2^3 - 48 \geq 0, \ 25(1-\gamma) x_1^2 x_2^6 - 480 x_1 x_2^3 + \left(48^2 - 144^2 \gamma\right) \geq 0.$$

Thus, the chance-constrained optimization model of the cantilever beam design problem can be re-stated as the following nonlinear optimization problem:

$$\begin{array}{ll} \min & l x_1 x_2 \\ \text{s.t.} & 5 x_1 x_2^3 - 48 \geq 0 \\ & 25(1-\gamma) x_1^2 x_2^6 - 480 x_1 x_2^3 + \left(48^2 - 144^2 \gamma\right) \geq 0 \\ & x_1 \geq 0.1, \ x_2 \geq 0.1. \end{array} \tag{5}$$

[2] Mahidhar V. Reddy, Ramana V. Grandhi, and Dale A. Hopkins. Reliability based structural optimization – A simplified safety index approach. *Comput. Struct.,* 53(6):1407-1418, 1994.

2 Calculus and analysis review

Almost all books on nonlinear programming have an appendix reviewing the relevant notions.

Most of these should be familiar to you from a course in analysis. Most of material in this course is based in some form on these concepts, therefore, to succeed in this course you should be not just familiar, but comfortable working with these concepts.

Vectors and Norms

- $\mathbb{R}^n$: set of n-dimensional real vectors $(x_1, \ldots, x_n)^T$ ("x^T" — transpose)
- Definition: *norm* $\|\cdot\|$ on $\mathbb{R}^n$: a mapping of $\mathbb{R}^n$ onto $\mathbb{R}$ such that:
  1. $\|x\| \geq 0\ \forall x \in \mathbb{R}^n$; $\|x\| = 0 \Leftrightarrow x = 0$.
  2. $\|cx\| = |c| \cdot \|x\|\ \forall c \in \mathbb{R}, x \in \mathbb{R}^n$.
  3. $\|x + y\| \leq \|x\| + \|y\|\ \forall x, y \in \mathbb{R}^n$.
- Euclidean norm: $\|\cdot\|_2$: $\|x\|_2 = \sqrt{x^T x} = \left(\sum_{i=1}^n x_i^2\right)^{1/2}$.
- Schwartz inequality: $|x^T y| \leq \|x\|_2 \cdot \|y\|_2$ with equality $\Leftrightarrow x = \alpha y$.
- All norms in $\mathbb{R}^n$ are *equivalent*, i.e., for any $\|\cdot\|^1$ and $\|\cdot\|^2$ $\exists \alpha_1, \alpha_2 > 0$ s.t. $\alpha_1\|x\|^1 \leq \|x\|^2 \leq \alpha_2\|x\|^1\ \forall x \in \mathbb{R}^n$.
- ϵ*–Neighborhood*: $N_\epsilon(x) = B(x, \epsilon) = \{y\ :\ \|y - x\| \leq \epsilon\}$ (sometimes — strict inequality).

Sequences and Limits.

Sequences in $\mathbb{R}$

- Notation: a *sequence*: $\{x_k\ :\ k = 1, 2, \ldots\} \subset \mathbb{R}$, $\{x_k\}$ for short.
- Definition: $\{x_k\} \subset \mathbb{R}$ *converges* to $x \in \mathbb{R}$ ($x_k \to x$, $\lim_{k\to\infty} x_k = x$) if

$$\forall \epsilon > 0\ \exists K :\ |x_k - x| \leq \epsilon \text{ (equiv. } x_k \in B(x, \epsilon)) \ \forall k \geq K.$$

  $x_k \to \infty$ $(-\infty)$ if

$$\forall A\ \exists K :\ x_k \geq A\ (x_k \leq A)\ \forall k \geq K.$$

- Definition: $\{x_k\}$ is *bounded above (below)*: $\exists A : x_k \leq A\ (x_k \geq A)\ \forall k$.
- Definition: $\{x_k\}$ is *bounded*: $\{|x_k|\}$ is bounded; equiv.,$\{x_k\}$ bounded above and below.
- Definition: $\{x_k\}$ is a *Cauchy sequence*: $\forall \epsilon > 0\ \exists K :\ |x_k - x_m| < \epsilon\ \forall k, m \geq K$
- Definition: $\{x_k\}$ is *nonincreasing (nondecreasing)*: $x_{k+1} \leq x_k\ (x_{k+1} \geq x_k)\ \forall k$;
  monotone: nondecreasing or nonincreasing.
- Proposition: Every monotone sequence in $\mathbb{R}$ has a limit (possibly infinite). If it is also bounded, the limit is finite.

Sequences in $\mathbb{R}^n$

- Definition: $\{x_k\} \subset \mathbb{R}^n$ *converges to $x \in \mathbb{R}^n$ (is bounded, is Cauchy)* if $\{x_k^i\}$ (the sequence of ith coordinates of x_k's) converges to the x^i (is bounded, is Cauchy) $\forall i$.

- Propositions:
  - $x_k \to x \Leftrightarrow \|x_k - x\| \to 0$
  - $\{x_k\}$ is Cauchy$\Leftrightarrow \forall \epsilon > 0 \; \exists K : \; \|x_k - x_m\| < \epsilon \; \forall k, m \geq K$
  - $\{x_k\}$ is bounded$\Leftrightarrow \{\|x_k\|\}$ is bounded
- Note: $\|x_n\| \to \|x\|$ does not imply that $x_n \to x$!! (Unless $x = 0$).

Limit Points

- Definition: x is a limit point of $\{x_k\}$ if there exists an infinite subsequence of $\{x_k\}$ that converges to x.
- Definition: x is a limit point of $A \subseteq \mathbb{R}^n$ if there exists an infinite sequence $\{x_k\} \subset A$ that converges to x.
- To see the difference between limits and limit points, consider the sequence

$$\{(1,0),(0,1),(-1,0),(0,-1),(1,0),(0,1),(-1,0),(0,-1),\ldots\}$$

- Proposition: let $\{x_k\} \subset \mathbb{R}^n$
  - $\{x_k\}$ converges $\Leftrightarrow$ it's a Cauchy sequence
  - If $\{x_k\}$ is bounded, $\{x_k\}$ converges $\Leftrightarrow$ it has a unique limit point
  - If $\{x_k\}$ is bounded, it has at least one limit point

Infimum and Supremum

- Let $A \subset \mathbb{R}$.
  Supremum of A ($\sup A$): smallest $y : \; x \leq y \; \forall x \in A$.
  Infimum of A ($\inf A$): largest $y : \; x \geq y \; \forall x \in A$.
- Not the same as max and min! Consider, for example, $(0, 1)$.

Closed and Open Sets

- Definition: a set $A \subseteq \mathbb{R}^n$ is closed if it contains all its limit points. In other words, for any sequence $\{x_k\} \subset A$ that has a limit x, $x \in A$.
- Definition: a set $A \subseteq \mathbb{R}^n$ is open if its complement, $\mathbb{R}^n \backslash A$, is closed
- Definition: a point $x \in A$ is *interior* if there is a neighborhood of x contained in A
- Proposition
  1. Union of *finitely many* closed sets is closed.
  2. Intersection of closed sets is closed.
  3. Union of open sets is open.
  4. Intersection of *finitely many* open sets is open.
  5. A set is open $\Leftrightarrow$ All of its elements are interior points.
  6. Every subspace of $\mathbb{R}^n$ is closed.

- Examples: neighborhoods of x:
  $\{y \ : \ \|y - x\| \leq \epsilon\}$ — closed
  $\{y \ : \ \|y - x\| < \epsilon\}$ — open
- Some sets are neither: $(0, 1]$.

Functions and Continuity

- $A \subseteq \mathbb{R}^m$, $f : A \to \mathbb{R}$ – a function.
- Definition: f is *continuous* at $\bar{x}$ if

$$\forall \epsilon > 0 \ \exists \delta > 0 : \ x \in A, \ \|x - \bar{x}\| < \delta \Rightarrow |f(x) - f(\bar{x})| < \epsilon.$$

- Proposition: f is continuous at $\bar{x} \Leftrightarrow$ for any $\{x_n\} \subset A : \ x_n \to \bar{x}$ we have $f(x_n) \to f(\bar{x})$. (In other words, $\lim f(x_n) = f(\lim x_n)$.)
- Proposition:
  - Sums, products and inverses of continuous functions are continuous (in the last case, provided the function is never zero).
  - Composition of two continuous functions is continuous.
  - Any vector norm is a continuous function.

Differentiation

Real-valued functions: Let $f : X \to \mathbb{R}$, where $X \subset \mathbb{R}^n$ is open.

- Definition: f is *differentiable* at $\bar{x} \in X$ if there exists a vector $\nabla f(\bar{x})$ (the *gradient* of f at $\bar{x}$) and a function $\alpha(\bar{x}, y) : X \to \mathbb{R}$ satisfying $\lim_{y \to 0} \alpha(\bar{x}, y) = 0$, such that for each $x \in X$

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

  f is *differentiable on* X if f is differentiable $\forall \bar{x} \in X$. The gradient vector is a vector of partial derivatives:

$$\nabla f(\bar{x}) = \left(\frac{\partial f(\bar{x})}{\partial x_1}, \ldots, \frac{\partial f(\bar{x})}{\partial x_n} \right)^T.$$

  The *directional derivative* of f at $\bar{x}$ in the direction d is

$$\lim_{\lambda \to 0} \frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda} = \nabla f(\bar{x})^T d$$

- Definition: the function f is *twice differentiable* at $\bar{x} \in X$ if there exists a vector $\nabla f(\bar{x})$ and an $n \times n$ symmetric matrix $H(\bar{x})$ (the *Hessian* of f at $\bar{x}$) such that for each $x \in X$

$$f(x) = f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) + \frac{1}{2} (x - \bar{x})^T H(\bar{x}) (x - \bar{x}) + \|x - \bar{x}\|^2 \alpha(\bar{x}, x - \bar{x}),$$

  and $\lim_{y \to 0} \alpha(\bar{x}, y) = 0$. f is *twice differentiable on* X if f is twice differentiable $\forall \bar{x} \in X$. The Hessian, which we often denote by $H(x)$ for short, is a matrix of second partial derivatives:

$$[H(x)]_{ij} = \frac{\partial^2 f(\bar{x})}{\partial x_i \partial x_j},$$

and for functions with continuous second derivatives, it will always be symmetric:

$$\frac{\partial^2 f(\bar{x})}{\partial x_i \partial x_j} = \frac{\partial^2 f(\bar{x})}{\partial x_j \partial x_i}$$

- Example:

$$f(x) = 3x_1^2 x_2^3 + x_2^2 x_3^3$$

$$\nabla f(x) = \begin{pmatrix} 6x_1 x_2^3 \\ 9x_1^2 x_2^2 + 2x_2 x_3^3 \\ 3x_2^2 x_3^2 \end{pmatrix}$$

$$H(x) = \begin{bmatrix} 6x_2^3 & 18x_1 x_2^2 & 0 \\ 18x_1 x_2^2 & 18x_1^2 x_2 + 2x_3^3 & 6x_2 x_3^2 \\ 0 & 6x_2 x_3^2 & 6x_2^2 x_3 \end{bmatrix}$$

- See additional handout to verify your understanding and derive the gradient and Hessian of linear and quadratic functions.

Vector-valued functions: Let $f : X \to \mathbb{R}^m$, where $X \subset \mathbb{R}^n$ is open.

- $$f(x) = f(x_1, \ldots, x_n) = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ f_2(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix},$$

  where each of the functions f_i is a real-valued function.

- Definition: the *Jacobian* of f at point $\bar{x}$ is the matrix whose jth row is the gradient of f_j at $\bar{x}$, transposed. More specifically, the Jacobian of f at $\bar{x}$ is defined as $\nabla f(\bar{x})^T$, where $\nabla f(\bar{x})$ is the matrix with entries:

$$[\nabla f(\bar{x})]_{ij} = \frac{\partial f_j(\bar{x})}{\partial x_i}.$$

  Notice that the jth column of $\nabla f(\bar{x})$ is the gradient of f_j at $\bar{x}$ (what happens when $m = 1$?)

- Example:

$$f(x) = \begin{pmatrix} \sin x_1 + \cos x_2 \\ e^{3x_1 + x_2^2} \\ 4x_1^3 + 7x_1 x_2^2 \end{pmatrix}.$$

  Then

$$\nabla f(x)^T = \begin{pmatrix} \cos x_1 & -\sin x_2 \\ 3e^{3x_1 + x_2^2} & 2x_2 e^{3x_1 + x_2^2} \\ 12x_1^2 + 7x_2^2 & 14x_1 x_2 \end{pmatrix}.$$

Other well-known results from calculus and analysis will be introduced throughout the course as needed.

3 Basic notions in optimization

3.1 Types of optimization problems

Unconstrained Optimization Problem:

$$\begin{array}{lll} \text{(P)} & \min_x & f(x) \\ & \text{s.t.} & x \in X, \end{array}$$

where $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, $f(x) : \mathbb{R}^n \to \mathbb{R}$, and X is an open set (usually $X = \mathbb{R}^n$).

Constrained Optimization Problem:

$$\begin{array}{lll} \text{(P)} & \min_x & f(x) \\ & \text{s.t.} & g_i(x) \leq 0 \quad i = 1, \ldots, m \\ & & h_i(x) = 0 \quad i = 1, \ldots, l \\ & & x \in X, \end{array}$$

where $g_1(x), \ldots, g_m(x), h_1(x), \ldots, h_l(x) : \mathbb{R}^n \to \mathbb{R}$.

Let $g(x) = (g_1(x), \ldots, g_m(x))^T : \mathbb{R}^n \to \mathbb{R}^m$, $h(x) = (h_1(x), \ldots, h_l(x))^T : \mathbb{R}^n \to \mathbb{R}^l$. Then (P) can be written as

$$\begin{array}{lll} \text{(P)} & \min_x & f(x) \\ & \text{s.t.} & g(x) \leq 0 \\ & & h(x) = 0 \\ & & x \in X. \end{array} \tag{6}$$

3.2 Constraints and feasible regions

A point x is *feasible* for (P) if it satisfies all the constraints. (For an unconstrained problem, $x \in X$.) The set of all feasible points forms the *feasible region*, or *feasible set* (let us denote it by S).

At a feasible point $\bar{x}$ an inequality constraint $g_i(x) \leq 0$ is said to be *binding*, or *active* if $g_i(\bar{x}) = 0$, and *nonbinding*, or *nonactive* if $g_i(\bar{x}) < 0$ (all equality constraints are considered to be active at any feasible point).

3.3 Types of optimal solutions

Consider a general optimization problem

$$\text{(P)} \ \min_{x \in S} \ \text{or} \ \max_{x \in S} f(x).$$

Recall: an ϵ-neighborhood of $\bar{x}$, or a *ball* centered at $\bar{x}$ with radius ϵ is the set:

$$B(\bar{x}, \epsilon) = N_\epsilon(\bar{x}) := \{x : \|x - \bar{x}\| \leq \epsilon\}.$$

We have the following definitions of local/global, strict/non-strict minimizers/maximizers.[3]

Definition 1 (cf. BSS 3.4.1) *In the optimization problem (P),*

- $x \in S$ *is a* global minimizer *of (P) if* $f(x) \leq f(y)$ *for all* $y \in S$.
- $x \in S$ *is a* strict global minimizer *of (P) if* $f(x) < f(y)$ *for all* $y \in S$, $y \neq x$.
- $x \in S$ *is a* local minimizer *of (P) if there exists* $\epsilon > 0$ *such that* $f(x) \leq f(y)$ *for all* $y \in B(x, \epsilon) \cap S$.
- $x \in S$ *is a* strict local minimizer *of (P) if there exists* $\epsilon > 0$ *such that* $f(x) < f(y)$ *for all* $y \in B(x, \epsilon) \cap \mathcal{F}$, $y \neq x$.
- $x \in S$ *is a* strict global maximizer *of (P) if* $f(x) > f(y)$ *for all* $y \in S$, $y \neq x$.
- $x \in S$ *is a* global maximizer *of (P) if* $f(x) \geq f(y)$ *for all* $y \in S$.
- $x \in S$ *is a* local maximizer *of (P) if there exists* $\epsilon > 0$ *such that* $f(x) \geq f(y)$ *for all* $y \in B(x, \epsilon) \cap \mathcal{F}$.
- $x \in S$ *is a* strict local maximizer *of (P) if there exists* $\epsilon > 0$ *such that* $f(x) > f(y)$ *for all* $y \in B(x, \epsilon) \cap S$, $y \neq x$.

3.4 Existence of solutions of optimization problems

Most of the topics of this course are concerned with

- existence of optimal solutions,
- characterization of optimal solutions, and
- algorithms for computing optimal solutions.

To illustrate the questions arising in the first topic, consider the following optimization problems:

- $$\begin{array}{lll} \text{(P)} & \min_x & \dfrac{1+x}{2x} \\ & \text{s.t.} & x \geq 1 \ . \end{array}$$

  Here there is no optimal solution because the feasible region is unbounded

- $$\begin{array}{lll} \text{(P)} & \min_x & \dfrac{1}{x} \\ & \text{s.t.} & 1 \leq x < 2 \ . \end{array}$$

  Here there is no optimal solution because the feasible region is not closed.

- $$\begin{array}{lll} \text{(P)} & \min_x & f(x) \\ & \text{s.t.} & 1 \leq x \leq 2 \ , \end{array}$$

[3] I will try to reserve the terms "minimum," "maximum," and "optimum" for the objective function values at the appropriate points $x \in S$, as opposed to the points themselves. Many books, however, do not make such distinctions, referring as, say, a "minimum" to both the point at which the function is minimized, and the function value at that point. Which one is being talked about is usually clear from the context, and I might inadvertently slip on occasion.

where

$$f(x) = \begin{cases} 1/x, & x < 2 \\ 1, & x = 2 \end{cases}$$

Here there is no optimal solution because the function $f(\cdot)$ is not continuous.

Theorem 2 (Weierstrass' Theorem for sequences) *Let $\{x_k\}$, $k \to \infty$ be an infinite sequence of points in the compact (i.e., closed and bounded) set S. Then some infinite subsequence of points x_{k_j} converges to a point contained in S.*

Theorem 3 (Weierstrass' Theorem for functions, BSS 2.3.1) *Let $f(x)$ be a continuous real-valued function on the compact nonempty set $S \subset \mathbb{R}^n$. Then S contains a point that minimizes (maximizes) f on the set S.*

4 Optimality conditions for unconstrained problems

The definitions of global and local solutions of optimization problems are intuitive, but usually impossible to check directly. Hence, we will derive easily verifiable conditions that are either necessary for a point to be a local minimizer (thus helping us to identify candidates for minimizers), or sufficient (thus allowing us to confirm that the point being considered is a local minimizer), or, sometimes, both.

$$\begin{array}{lll} \text{(P)} & \min & f(x) \\ & \text{s.t.} & x \in X, \end{array}$$

where $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, and X — an open set (usually, $X = \mathbb{R}^n$).

4.1 Optimality conditions: the necessary and the sufficient

Necessary condition for local optimality: "if $\bar{x}$ is a local minimizer of (P), then $\bar{x}$ must satisfy..." Such conditions help us identify all candidates for local optimizers.

Theorem 4 (BSS 4.1.2) *Suppose that f is differentiable at $\bar{x}$. If there is a vector d such that $\nabla f(\bar{x})^T d < 0$, then for all $\lambda > 0$ sufficiently small, $f(\bar{x}+\lambda d) < f(\bar{x})$ (d is called a* descent *direction if it satisfies the latter condition).*[4]

Proof: We have:

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \nabla f(\bar{x})^T d + \lambda \|d\| \alpha(\bar{x}, \lambda d),$$

where $\alpha(\bar{x}, \lambda d) \to 0$ as $\lambda \to 0$. Rearranging,

$$\frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda} = \nabla f(\bar{x})^T d + \|d\| \alpha(\bar{x}, \lambda d).$$

Since $\nabla f(\bar{x})^T d < 0$ and $\alpha(\bar{x}, \lambda d) \to 0$ as $\lambda \to 0$, $f(\bar{x} + \lambda d) - f(\bar{x}) < 0$ for all $\lambda > 0$ sufficiently small. ∎

Corollary 5 *Suppose f is differentiable at $\bar{x}$. If $\bar{x}$ is a local minimizer, then $\nabla f(\bar{x}) = 0$ (such a point is called a* stationary point*).*

Proof: If $\nabla f(\bar{x}) \neq 0$, then $d = -\nabla f(\bar{x})$ is a descent direction, whereby $\bar{x}$ cannot be a local minimizer. ∎

The above corollary is a *first order necessary optimality condition* for an unconstrained minimization problem. However, a stationary point can be a local minimizer, a local maximizer, or neither. The following theorem will provide a *second order necessary optimality condition*. First, a definition:

Definition 6 *An $n \times n$ matrix M is called* symmetric *if $M_{ij} = M_{ji}$ $\forall i, j$. A symmetric $n \times n$ matrix M is called*

- positive definite *if $x^T M x > 0$ $\forall x \in \mathbb{R}^n$, $x \neq 0$*
- positive semidefinite *if $x^T M x \geq 0$ $\forall x \in \mathbb{R}^n$*
- negative definite *if $x^T M x < 0$ $\forall x \in \mathbb{R}^n$, $x \neq 0$*

[4] The book is trying to be more precise about the "sufficiently small" statement, but I believe makes a typo.

- negative semidefinite *if* $x^T M x \leq 0 \ \forall x \in \mathbb{R}^n$
- indefinite *if* $\exists x, y \in \mathbb{R}^n : \ x^T M x > 0, \ y^T M y < 0.$

We say that M is SPD if M is symmetric and positive definite. Similarly, we say that M is SPSD if M is symmetric and positive semi-definite.

Example 1

$$M = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$$

is positive definite.

Example 2

$$M = \begin{pmatrix} 8 & -1 \\ -1 & 1 \end{pmatrix}$$

is positive definite. To see this, note that for $x \neq 0$,

$$x^T M x = 8x_1^2 - 2x_1 x_2 + x_2^2 = 7x_1^2 + (x_1 - x_2)^2 > 0 \ .$$

Since M is a symmetric matrix, all it eigenvalues are real numbers. It can be shown that M is positive semidefinite if and only if all of its eigenvalues are nonnegative, positive definite if all of its eigenvalues are positive, etc.

Theorem 7 (BSS 4.1.3) *Suppose that* f *is twice continuously differentiable at* $\bar{x} \in X$*. If* $\bar{x}$ *is a local minimizer, then* $\nabla f(\bar{x}) = 0$ *and* $H(\bar{x})$ *(the Hessian at* $\bar{x}$*) is positive semidefinite.*

Proof: From the first order necessary condition, $\nabla f(\bar{x}) = 0$. Suppose $H(\bar{x})$ is not positive semidefinite. Then $\exists d$ such that $d^T H(\bar{x}) d < 0$. We have:

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \nabla f(\bar{x})^T d + \frac{1}{2}\lambda^2 d^T H(\bar{x}) d + \lambda^2 \|d\|^2 \alpha(\bar{x}, \lambda d)$$

$$= f(\bar{x}) + \frac{1}{2}\lambda^2 d^T H(\bar{x}) d + \lambda^2 \|d\|^2 \alpha(\bar{x}, \lambda d),$$

where $\alpha(\bar{x}, \lambda d) \to 0$ as $\lambda \to 0$. Rearranging,

$$\frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda^2} = \frac{1}{2} d^T H(\bar{x}) d + \|d\|^2 \alpha(\bar{x}, \lambda d).$$

Since $d^T H(\bar{x}) d < 0$ and $\alpha(\bar{x}, \lambda d) \to 0$ as $\lambda \to 0$, $f(\bar{x} + \lambda d) - f(\bar{x}) < 0$ for all $\lambda > 0$ sufficiently small — contradiction. ∎

Example 3 *Let*

$$f(x) = \frac{1}{2}x_1^2 + x_1 x_2 + 2x_2^2 - 4x_1 - 4x_2 - x_2^3 \ .$$

Then

$$\nabla f(x) = \left(x_1 + x_2 - 4, x_1 + 4x_2 - 4 - 3x_2^2\right)^T \ ,$$

and

$$H(x) = \begin{pmatrix} 1 & 1 \\ 1 & 4 - 6x_2 \end{pmatrix} \ .$$

$\nabla f(x) = 0$ *has exactly two solutions:* $\bar{x} = (4, 0)$ *and* $\tilde{x} = (3, 1)$. *But*

$$H(\tilde{x}) = \begin{pmatrix} 1 & 1 \\ 1 & -2 \end{pmatrix}$$

is indefinite, therefore, the only possible candidate for a local minimum is $\bar{x} = (4, 0)$.

Necessary conditions only allow us to come up with a list of candidate points for minimizers. *Sufficient* condition for local optimality: "if $\bar{x}$ satisfies ..., then $\bar{x}$ is a local minimizer of (P)."

Theorem 8 (BSS 4.1.4) *Suppose that* f *is twice differentiable at* $\bar{x}$. *If* $\nabla f(\bar{x}) = 0$ *and* $H(\bar{x})$ *is positive definite, then* $\bar{x}$ *is a (strict) local minimizer.*

Proof:

$$f(x) = f(\bar{x}) + \frac{1}{2}(x - \bar{x})^T H(\bar{x})(x - \bar{x}) + \|x - \bar{x}\|^2 \alpha(\bar{x}, x - \bar{x}).$$

Suppose that $\bar{x}$ is not a strict local minimizer. Then there exists a sequence $x_k \to \bar{x}$ such that $x_k \neq \bar{x}$ and $f(x_k) \leq f(\bar{x})$ for all k. Define $d_k = \frac{x_k - \bar{x}}{\|x_k - \bar{x}\|}$. Then

$$f(x_k) = f(\bar{x}) + \|x_k - \bar{x}\|^2 \left(\frac{1}{2} d_k^T H(\bar{x}) d_k + \alpha(\bar{x}, x_k - \bar{x}) \right), \text{ so}$$

$$\frac{1}{2} d_k^T H(\bar{x}) d_k + \alpha(\bar{x}, x_k - \bar{x}) = \frac{f(x_k) - f(\bar{x})}{\|x_k - \bar{x}\|^2} \leq 0.$$

Since $\|d_k\| = 1$ for any k, there exists a subsequence of $\{d_k\}$ converging to some point d such that $\|d\| = 1$ (by Theorem 2). Assume wolog that $d_k \to d$. Then

$$0 \geq \lim_{k \to \infty} \frac{1}{2} d_k^T H(\bar{x}) d_k + \alpha(\bar{x}, x_k - \bar{x}) = \frac{1}{2} d^T H(\bar{x}) d,$$

which is a contradiction with positive definiteness of $H(\bar{x})$. ∎

Note:

- If $\nabla f(\bar{x}) = 0$ and $H(\bar{x})$ is negative definite, then $\bar{x}$ is a local maximizer.
- If $\nabla f(\bar{x}) = 0$ and $H(\bar{x})$ is positive *semi*definite, we cannot be sure if $\bar{x}$ is a local minimizer.

Example 4 *Consider the function*

$$f(x) = \frac{1}{3} x_1^3 + \frac{1}{2} x_1^2 + 2 x_1 x_2 + \frac{1}{2} x_2^2 - x_2 + 9.$$

Stationary points are candidates for optimality; to find them we solve

$$\nabla f(x) = \begin{pmatrix} x_1^2 + x_1 + 2x_2 \\ 2x_1 + x_2 - 1 \end{pmatrix} = 0.$$

Solving the above system of equations results in two stationary points: $x_a = (1, -1)^T$ *and* $x_b = (2, -3)$. *The Hessian is*

$$H(x) = \begin{pmatrix} 2x_1 + 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

In particular,

$$H(x_a) = \begin{pmatrix} 3 & 2 \\ 2 & 1 \end{pmatrix}, \text{ and } H(x_b) = \begin{pmatrix} 5 & 2 \\ 2 & 1 \end{pmatrix}.$$

Here, $H(x_a)$ *is indefinite, hence* x_a *is neither a local minimizer or maximizer.* $H(x_b)$ *is positive definite, hence* x_b *is a local minimizer. Therefore, the function has only one local minimizer — does this mean that it is also a global minimizer?*

4.2 Convexity and minimization

Definitions:

- Let $x, y \in \mathbb{R}^n$. Points of the form $\lambda x + (1-\lambda)y$ for $\lambda \in [0,1]$ are called *convex combinations* of x and y. More generally, point y is a convex combination of points $x_1, \ldots, x_k$ if $y = \sum_{i=1}^k \alpha_i x_i$ where $\alpha_i \geq 0 \ \forall i$, and $\sum_{i=1}^k \alpha_i = 1$.
- A set $S \subset \mathbb{R}^n$ is called *convex* if $\forall x, y \in S$ and $\forall \lambda \in [0,1]$, $\lambda x + (1-\lambda)y \in S$.
- A function $f : S \to \mathbb{R}$, where S is a nonempty convex set is a *convex function* if
$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \ \forall x, y \in S, \ \forall \lambda \in [0,1].$$
- A function f as above is called a *strictly convex* function if the inequality above is strict for all $x \neq y$ and $\lambda \in (0,1)$.
- A function $f : S \to \mathbb{R}$ is called *concave* (*strictly concave*) if $(-f)$ is convex (strictly convex).

Consider the problem:

$$\begin{array}{ll} \text{(CP)} & \min_x \quad f(x) \\ & \text{s.t.} \quad x \in S \ . \end{array}$$

Theorem 9 (BSS 3.4.2) *Suppose S is a nonempty convex set, $f : S \to \mathbb{R}$ is a convex function, and $\bar{x}$ is a local minimizer of (CP). Then $\bar{x}$ is a global minimizer of f over S.*

Proof: Suppose $\bar{x}$ is not a global minimizer, i.e., $\exists y \in S : \ f(y) < f(\bar{x})$. Let $y(\lambda) = \lambda \bar{x} + (1-\lambda)y$, which is a convex combination of $\bar{x}$ and y for $\lambda \in [0,1]$ (and therefore, $y(\lambda) \in S$ for $\lambda \in [0,1]$). Note that $y(\lambda) \to \bar{x}$ as $\lambda \to 1$.

From the convexity of f,

$$f(y(\lambda)) = f(\lambda \bar{x} + (1-\lambda)y) \leq \lambda f(\bar{x}) + (1-\lambda) f(y) < \lambda f(\bar{x}) + (1-\lambda) f(\bar{x}) = f(\bar{x})$$

for all $\lambda \in (0,1)$. Therefore, $f(y(\lambda)) < f(\bar{x})$ for all $\lambda \in (0,1)$, and so $\bar{x}$ is not a local minimizer, resulting in a contradiction. ∎

Note:

- A problem of minimizing a convex function over a convex feasible region (such as we considered in the theorem) is a *convex programming problem.*
- If f is strictly convex, a local minimizer is the *unique* global minimizer.
- If f is (strictly) concave, a local maximizer is a (unique) global maximizer.

The following results help us to determine when a function is convex.

Theorem 10 (BSS 3.3.3) *Suppose $X \subseteq \mathbb{R}^n$ is a non-empty open convex set, and $f : X \to \mathbb{R}$ is differentiable. Then f is convex iff ("if and only if") it satisfies the* gradient inequality*:*

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \quad \forall x, y \in X.$$

Proof: Suppose f is convex. Then, for any $\lambda \in (0,1]$,

$$f(\lambda y + (1-\lambda)x) \leq \lambda f(y) + (1-\lambda) f(x) \Rightarrow \frac{f(x + \lambda(y-x)) - f(x)}{\lambda} \leq f(y) - f(x).$$

Letting $\lambda \to 0$, we obtain: $\nabla f(x)^T(y-x) \leq f(y) - f(x)$, establishing the "only if" part.

Now, suppose that the gradient inequality holds $\forall x, y \in X$. Let w and z be any two points in X. Let $\lambda \in [0,1]$, and set $x = \lambda w + (1-\lambda)z$. Then

$$f(w) \geq f(x) + \nabla f(x)^T(w-x) \text{ and } f(z) \geq f(x) + \nabla f(x)^T(z-x).$$

Taking a convex combination of the above inequalities,

$$\begin{aligned}\lambda f(w) + (1-\lambda) f(z) &\geq f(x) + \nabla f(x)^T(\lambda(w-x) + (1-\lambda)(z-x)) \\ &= f(x) + \nabla f(x)^T 0 = f(\lambda w + (1-\lambda)z),\end{aligned}$$

so that $f(x)$ is convex. ∎

In one dimension, the gradient inequality has the form $f(y) \geq f(x) + f'(x)(y-x)\ \forall x, y \in X$.

The following theorem provides another necessary and sufficient condition, for the case when f is twice continuously differentiable.

Theorem 11 (BSS 3.3.7) *Suppose X is a non-empty open convex set, and $f : X \to \mathbb{R}$ is twice continuously differentiable. Then f is convex iff the Hessian of f, $H(x)$, is positive semidefinite $\forall x \in X$.*

Proof: Suppose f is convex. Let $\bar{x} \in X$ and d be any direction. Since X is open, for $\lambda > 0$ sufficiently small, $\bar{x} + \lambda d \in X$. We have:

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \nabla f(\bar{x})^T(\lambda d) + \frac{1}{2}(\lambda d)^T H(\bar{x})(\lambda d) + \|\lambda d\|^2 \alpha(\bar{x}, \lambda d),$$

where $\alpha(\bar{x}, y) \to 0$ as $y \to 0$. Using the gradient inequality, we obtain

$$\lambda^2 \left(\frac{1}{2} d^T H(\bar{x}) d + \|d\|^2 \alpha(\bar{x}, \lambda d) \right) \geq 0.$$

Dividing by $\lambda^2 > 0$ and letting $\lambda \to 0$, we obtain $d^T H(\bar{x}) d \geq 0$, proving the "only if" part.

Conversely, suppose that $H(z)$ is positive semidefinite for all $z \in X$. Let $x, y \in S$ be arbitrary. Invoking the second-order version of the Taylor's theorem, we have:

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

for some z which is a convex combination of x and y (and hence $z \in X$). Since $H(z)$ is positive semidefinite, the gradient inequality holds, and hence f is convex. ∎

In one dimension, the Hessian is the second derivative of the function, the positive semidefiniteness condition can be stated as $f'' \geq 0\ \forall x \in X$.

One can also show the following sufficient (but not necessary!) condition:

Theorem 12 (BSS 3.3.8) *Suppose X is a non-empty open convex set, and $f : X \to \mathbb{R}$ is twice continuously differentiable. Then f is strictly convex if the Hessian of f, $H(x)$, is positive definite $\forall x \in X$.*

For convex (unconstrainted) optimization problems, the optimality conditions of the previous subsection can be simplified significantly, providing a single necessary and sufficient condition for *global* optimality:

Theorem 13 *Suppose $f : X \to \mathbb{R}$ is convex and differentiable on X. Then $\bar{x} \in X$ is a global minimizer iff $\nabla f(\bar{x}) = 0$.*

Proof: The necessity of the condition $\nabla f(\bar{x}) = 0$ was established regardless of convexity of the function.

Suppose $\nabla f(\bar{x}) = 0$. Then, by gradient inequality, $f(y) \geq f(\bar{x}) + \nabla f(\bar{x})^T(y - \bar{x}) = f(\bar{x})$ for all $y \in X$, and so $\bar{x}$ is a global minimizer. ∎

Example 5 *Let*

$$f(x) = -\ln(1 - x_1 - x_2) - \ln x_1 - \ln x_2 \ .$$

Then

$$\nabla f(x) = \begin{pmatrix} \frac{1}{1-x_1-x_2} - \frac{1}{x_1} \\ \frac{1}{1-x_1-x_2} - \frac{1}{x_2} \end{pmatrix},$$

and

$$H(x) = \begin{pmatrix} \left(\frac{1}{1-x_1-x_2}\right)^2 + \left(\frac{1}{x_1}\right)^2 & \left(\frac{1}{1-x_1-x_2}\right)^2 \\ \left(\frac{1}{1-x_1-x_2}\right)^2 & \left(\frac{1}{1-x_1-x_2}\right)^2 + \left(\frac{1}{x_2}\right)^2 \end{pmatrix}.$$

It is actually easy to prove that $f(x)$ is a strictly convex function, and hence that $H(x)$ is positive definite on its domain $X = \{(x_1, x_2) : x_1 > 0, x_2 > 0, x_1 + x_2 < 1\}$. At $\bar{x} = \left(\frac{1}{3}, \frac{1}{3}\right)$ we have $\nabla f(\bar{x}) = 0$, and so $\bar{x}$ is the unique global minimizer of $f(x)$.

5 Line search methods: one-dimensional optimization

5.1 General optimization algorithm

Recall: we are attempting to solve the problem

$$\begin{array}{lll} \text{(P)} & \min & f(x) \\ & \text{s.t.} & x \in \mathbb{R}^n \end{array}$$

where $f(x)$ is differentiable.

Solutions to optimization problems are almost always impossible to obtain directly (or "in closed form") — with a few exceptions. Hence, for the most part, we will solve these problems with iterative algorithms. These algorithms typically require the user to supply a starting point x_0. Beginning at x_0, an iterative algorithm will generate a sequence of points $\{x_k\}_{k=0}^{\infty}$ called *iterates*. In deciding how to generate the next iterate, x_{k+1}, the algorithms use information about the function f at the current iterate, x_k, and sometimes past iterates $x_0, \ldots, x_{k-1}$. In practice, rather than constructing an infinite sequence of iterates, algorithms stop when an appropriate *termination criterion* is satisfied, indicating either that the problem has been solved within a desired accuracy, or that no further progress can be made.

Most algorithms for unconstrained optimization we will discuss fall into the category of *directional search* algorithms:

General directional search optimization algorithm

Initialization Specify an initial guess of the solution x_0

Iteration For $k = 0, 1, \ldots,$
If x_k is optimal, stop
Otherwise,

- Determine d_k — a *search directions*
- Determine $\lambda_k > 0$ — a *step size*
- Determine $x_{k+1} = x_k + \lambda_k d_k$ — a new estimate of the solution.

Choosing the direction Typically, we require that d_k is a *descent direction* of f at x_k, that is,

$$f(x_k + \lambda d_k) < f(x_k) \ \forall \lambda \in (0, \epsilon]$$

for some $\epsilon > 0$. For the case when f is differentiable, we have shown in Theorem 4 that whenever $\nabla f(x_k) \neq 0$, any d_k such that $\nabla f(x_k)^T d_k < 0$ is a descent direction.

Often, direction is chosen to be of the form

$$d_k = -D_k \nabla f(x_k),$$

where D_k is a positive definite symmetric matrix. (Why is it important that D_k is positive definite?)

The following are the two basic methods for choosing the matrix D_k at each iteration; they give rise to two classic algorithms for unconstrained optimization we are going to discuss in class:

- *Steepest descent:* $D_k = I,\ k = 0, 1, 2, \ldots$
- *Newton's method:* $D_k = H(x_k)^{-1}$ (provided $H(x_k) = \nabla^2 f(x_k)$ is positive definite.)

Choosing the stepsize After d_k is fixed, λ_k ideally would solve the one-dimensional optimization problem

$$\min_{\lambda \geq 0} f(x_k + \lambda d_k).$$

This optimization problem is usually also impossible to solve exactly. Instead, λ_k is computed (via an iterative procedure referred to as *line search*) either to approximately solve the above optimization problem, or to ensure a "sufficient" decrease in the value of f.

Testing for optimality: Based on the optimality conditions, x_k is a locally optimal if $\nabla f(x_k) = 0$ and $H(x_k)$ is positive definite. However, such a point is unlikely to be found. In fact, the most of the analysis of the algorithms in the above form deals with their *limiting behavior*, i.e., analyzes the limit points of the infinite sequence of iterates generated by the algorithm. Thus, to implement the algorithm in practice, more realistic termination criteria need to be specified. They often hinge, at least in part, on approximately satisfying, to a certain tolerance, the first order necessary condition for optimality discussed in the previous section.

We begin by commenting on how the line search can be implemented in practice, and then discuss methods for choosing d_k in more detail.

5.2 Stepsize selection

In the statement of the algorithm above we assumed that at each iteration of the steepest descent algorithm we are selecting an appropriate stepsize λ_k. Although in some cases a simple stepsize selection rule (e.g., $\lambda_k = 1$ for all k, or a pre-determined sequence $\{\lambda_k\}_{k=0}^{\infty}$) is used, often the step size is chosen by performing a line search, i.e., solving a one-dimensional optimization problem

$$\lambda_k = \arg\min_{\lambda} \theta(\lambda) \stackrel{\triangle}{=} \arg\min_{\lambda} f(\bar{x} + \lambda d).$$

In some (small number of) cases it is possible to find the optimal stepsize "in closed form," however in general we need an iterative method to find the solution of this one-dimensional optimization problem.

There are many methods for solving such problems. We are going to describe two: the bisection method and Armijo rule.

5.2.1 A bisection algorithm for a line search of a convex function

Suppose that $f(x)$ is a differentiable *convex* function, and that we seek to solve:

$$\bar{\lambda} = \arg\min_{\lambda > 0} f(\bar{x} + \lambda \bar{d}),$$

where $\bar{x}$ is our current iterate, and $\bar{d}$ is the current direction generated by an algorithm that seeks to minimize $f(x)$. We assume that $\bar{d}$ is a descent direction. Let

$$\theta(\lambda) = f(\bar{x} + \lambda \bar{d}),$$

whereby $\theta(\lambda)$ is a convex function in the scalar variable λ, and our problem is to solve for

$$\bar{\lambda} = \arg\min_{\lambda>0} \theta(\lambda).$$

Applying the necessary and sufficient optimality condition to the convex function $\theta(\lambda)$, we want to find a value $\bar{\lambda}$ for which $\theta'(\bar{\lambda}) = 0$. It is elementary to show that $\theta'(\lambda) = \nabla f(\bar{x} + \lambda\bar{d})^T\bar{d}$. Therefore, since d is a descent direction, $\theta'(0) < 0$.

Suppose that we know a value $\hat{\lambda} > 0$ such that $\theta'(\hat{\lambda}) > 0$. Consider the following *bisection algorithm* for solving the equation $\theta'(\lambda) \approx 0$.

Step 0 Set $k = 0$. Set $\lambda_l := 0$ and $\lambda_u := \hat{\lambda}$.

Step k Set $\tilde{\lambda} = \frac{\lambda_u+\lambda_l}{2}$ and compute $\theta'(\tilde{\lambda})$.

- If $\theta'(\tilde{\lambda}) > 0$, re-set $\lambda_u := \tilde{\lambda}$. Set $k \leftarrow k + 1$.
- If $\theta'(\tilde{\lambda}) < 0$, re-set $\lambda_l := \tilde{\lambda}$. Set $k \leftarrow k + 1$.
- If $\theta'(\tilde{\lambda}) = 0$, stop.

Below are some observations on which the convergence of the algorithm rests:

- After every iteration of the bisection algorithm, the current interval $[\lambda_l, \lambda_u]$ must contain a point $\bar{\lambda}$ such that $\theta'(\bar{\lambda}) = 0$.
- At the k^{th} iteration of the bisection algorithm, the length of the current interval $[\lambda_l, \lambda_u]$ is

$$L = \left(\frac{1}{2}\right)^k (\hat{\lambda}).$$

- A value of λ such that $|\lambda - \bar{\lambda}| \leq \epsilon$ can be found in at most

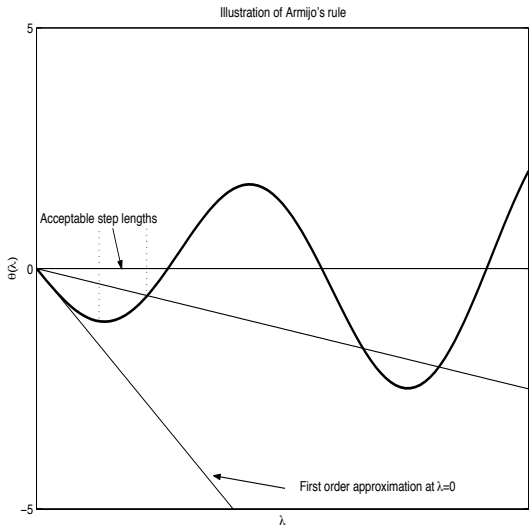
$$\left\lceil \log_2 \left(\frac{\hat{\lambda}}{\epsilon}\right) \right\rceil$$

  steps of the bisection algorithm.

Suppose that we do not have available a convenient value of a point $\hat{\lambda}$ for which $\theta'(\hat{\lambda}) > 0$. One way to proceed is to pick an initial "guess" of $\hat{\lambda}$ and compute $\theta'(\hat{\lambda})$. If $\theta'(\hat{\lambda}) > 0$, then proceed to the bisection algorithm; if $\theta'(\hat{\lambda}) \leq 0$, then re-set $\hat{\lambda} \leftarrow 2\hat{\lambda}$ and repeat the process.

In practice, we need to run the bisection algorithm with a stopping criterion. Some relevant stopping criteria are:

- Stop after a fixed number of iterations. That is stop when $k = \bar{k}$, where $\bar{k}$ specified by the user.
- Stop when the interval becomes small. That is, stop when $\lambda_u - \lambda_l \leq \epsilon$, where ϵ is specified by the user.
- Stop when $|\theta'(\tilde{\lambda})|$ becomes small. That is, stop when $|\theta'(\tilde{\lambda})| \leq \epsilon$, where ϵ is specified by the user.



5.2.2 Armijo rule

Very often performing an exact line search by a method such as the bisection method is too expensive computationally in the context of selecting a step size in an optimization algorithm. (Recall that we need to perform a line search at every iteration of our algorithm!) On the other hand, if we sacrifice accuracy of the line search, this can cause inferior performance of the overall algorithm.

The Armijo rule is one of several inexact line search methods which guarantees a sufficient degree of accuracy to ensure the algorithm convergence.

Armijo rule requires two parameters: $0 < \epsilon < 1$ and $\alpha > 1$. Suppose we are minimizing a function $\theta(\lambda)$ such that $\theta'(0) < 0$ (which is indeed the case for the line search problems arising in descent algorithms). Then the first order approximation of $\theta(\lambda)$ at $\lambda = 0$ is given by $\theta(0) + \lambda\theta'(0)$. Define $\hat{\theta}(\lambda) = \theta(0) + \lambda\epsilon\theta'(0)$ for $\lambda > 0$ (see Figure 5.2.2). A stepsize $\bar{\lambda}$ is considered acceptable by Armijo rule if

- $\theta(\bar{\lambda}) \leq \hat{\theta}(\bar{\lambda})$ (to assure sufficient decrease of θ) and
- $\theta(\alpha\bar{\lambda}) \geq \hat{\theta}(\alpha\bar{\lambda})$ (to prevent the step size from being too small).

The above rule yields a range of acceptable stepsizes. In practice, to find a step size in this range, Armijo rule is usually implemented in an iterative fashion (in this description we use $\alpha = 2$), using a fixed initial value of $\bar{\lambda} > 0$:

Step 0 Set k=0. $\lambda^0 = \bar{\lambda} > 0$.

Step k If $\theta(\lambda_k) \leq \hat{\theta}(\lambda_k)$, choose λ_k as the step size; stop. If $\theta(\lambda_k) > \hat{\theta}(\lambda_k)$, let $\lambda_{k+1} \leftarrow \frac{1}{2}\lambda_k$, $k \leftarrow k + 1$.

This iterative scheme is often referred to as *backtracking*. Note that as a result of backtracking, the chosen stepsize is $\lambda^t = \bar{\lambda}/2^t$, where $t \geq 0$ is the smallest integer such that $\theta(\bar{\lambda}/2^t) \leq \hat{\theta}(\bar{\lambda}/2^t)$.

6 The steepest descent algorithm for unconstrained optimization

6.1 The algorithm

Recall: we are attempting to solve the problem

$$\begin{array}{lll} \text{(P)} & \min & f(x) \\ & \text{s.t.} & x \in \mathbb{R}^n \end{array}$$

where $f(x)$ is differentiable. If $x = \bar{x}$ is a given point, a direction d is called a descent direction of f at $\bar{x}$ if there exists $\delta > 0$ such that $f(\bar{x} + \lambda d) < f(\bar{x})$ for all $\lambda \in (0, \delta)$. In particular, if

$$\nabla f(\bar{x})^T d = \lim_{\lambda \to 0^+} \frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda} < 0,$$

then d is a descent direction.

The steepest descent algorithm moves along the direction d with $\|d\| = 1$ that minimizes the above inner product (as a source of motivation, note that $f(x)$ can be approximated by its linear expansion $f(\bar{x} + \lambda d) \approx f(\bar{x}) + \lambda \nabla f(\bar{x})^T d$.)

It is not hard to see that so long as $\nabla f(\bar{x}) \neq 0$, the direction

$$\bar{d} = \frac{-\nabla f(\bar{x})}{\|\nabla f(\bar{x})\|} = \frac{-\nabla f(\bar{x})}{\sqrt{\nabla f(\bar{x})^T \nabla f(\bar{x})}}$$

is the (unit length) direction that minimizes the above inner product. Indeed, for any direction d with $\|d\| = 1$, the Schwartz inequality yields

$$\nabla f(\bar{x})^T d \geq -\|\nabla f(\bar{x})\| \cdot \|d\| = -\|\nabla f(\bar{x})\| = \nabla f(\bar{x})^T \bar{d}.$$

The direction $\bar{d} = -\nabla f(\bar{x})$ is called the *direction of steepest descent* at the point $\bar{x}$.

Note that $\bar{d} = -\nabla f(\bar{x})$ is a descent direction as long as $\nabla f(\bar{x}) \neq 0$. To see this, simply observe that $\bar{d}^T \nabla f(\bar{x}) = -\nabla f(\bar{x})^T \nabla f(\bar{x}) < 0$ so long as $\nabla f(\bar{x}) \neq 0$. Of course, if $\nabla f(\bar{x}) = 0$, then $\bar{x}$ is a candidate for local minimizer, i.e., $\bar{x}$ satisfies the first order necessary optimality condition.

A natural consequence of this is the following algorithm, called the *steepest descent algorithm.*

Steepest Descent Algorithm:

Step 0 Given x^0, set $k \leftarrow 0$

Step 1 $d_k = -\nabla f(x_k)$. If $d_k = 0$, then stop.

Step 2 Choose stepsize λ_k by performing an exact (or inexact) line search, i.e., solving $\lambda_k = \arg\min_{\lambda > 0} f(\bar{x} + \lambda d_k)$.

Step 3 Set $x_{k+1} \leftarrow x_k + \lambda_k d_k,\ k \leftarrow k + 1$. Go to **Step 1**.

Note from Step 2 and the fact that $d_k = -\nabla f(x_k)$ is a descent direction, it follows that $f(x_{k+1}) < f(x_k)$.

6.2 Global convergence

In this section we will show that, under certain assumptions on the behavior of the function $f(\cdot)$, the steepest descent algorithm converges to a point that satisfies the first order necessary conditions for optimality. We will consider two different stepsize selection rules, and correspondingly, will need to impose different assumptions on the function for each of them to "work."

Theorem 14 (Convergence Theorem) *Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable on the set $S(x_0) = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$, and that $S(x_0)$ is a closed and bounded set. Suppose further that the sequence $\{x_k\}$ is generated by the steepest descent algorithm with stepsizes λ_k chosen by an exact line search. Then every point $\bar{x}$ that is a limit point of the sequence $\{x_k\}$ satisfies $\nabla f(\bar{x}) = 0$.*

Proof: Notice that $f(x_{k+1}) \leq f(x_k) \leq f(x_0)$, and thus $\{x_k\} \subseteq S(x_0)$. By the Weierstrass' Theorem, at least one limit point of the sequence $\{x_k\}$ must exist. Let $\bar{x}$ be any such limit point. Without loss of generality, assume that $\lim_{k\to\infty} x_k = \bar{x}$.[5]

We will prove the theorem by contradiction, i.e., assume that $\nabla f(\bar{x}) \neq 0$. This being the case, there is a value of $\bar{\lambda} > 0$ such that $\delta \stackrel{\triangle}{=} f(\bar{x}) - f(\bar{x} + \bar{\lambda}\bar{d}) > 0$, where $\bar{d} = -\nabla f(\bar{x})$. Then also $(\bar{x} + \bar{\lambda}\bar{d}) \in \text{int}S$. (Why?)

Let $\{d_k\}$ be the sequence of directions generated by the algorithm, i.e., $d_k = -\nabla f(x_k)$. Since f is continuously differentiable, $\lim_{k\to\infty} d_k = \bar{d}$. Then since $(\bar{x} + \bar{\lambda}\bar{d}) \in \text{int}S$, and $(x_k + \bar{\lambda}d_k) \to (\bar{x} + \bar{\lambda}\bar{d})$, for k sufficiently large we have

$$f(x_k + \bar{\lambda}d_k) \leq f(\bar{x} + \bar{\lambda}\bar{d}) + \frac{\delta}{2} = f(\bar{x}) - \delta + \frac{\delta}{2} = f(\bar{x}) - \frac{\delta}{2}.$$

However,

$$f(\bar{x}) \leq f(x_k + \lambda_k d_k) \leq f(x_k + \bar{\lambda}d_k) \leq f(\bar{x}) - \frac{\delta}{2},$$

which is of course a contradiction. Thus $\bar{d} = -\nabla f(\bar{x}) = 0$. ∎

Next, we will study the convergence properties of the steepest descent algorithm with the stepsizes chosen at each iteration by the Armijo rule; in particular, its backtracking implementation.

We will assume that the function $f(x)$ satisfies the following property: for some $G > 0$,

$$\|\nabla f(x) - \nabla f(y)\| \leq G\|x - y\| \ \forall x, y \in S(x_0) = \{x : f(x) \leq f(x_0)\}.$$

It is said that the gradient function $\nabla f(x)$ is *Lipschitz continuous with constant $G > 0$* on the set $S(x_0)$. Note that this assumption is stronger than just continuity of $\nabla f(x)$.

For example, if the (matrix) norm of the Hessian $H(x)$ is bounded by a constant $G > 0$ everywhere on the set $S(x_0)$, the gradient function will be Lipschitz continuous.

Theorem 15 (8.6.3) *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is such that its gradient is Lipschitz continuous with constant $G > 0$ on the set $S(x_0)$. Pick some step parameter $\bar{\lambda} > 0$, and let $0 < \epsilon < 1$. Suppose the sequence $\{x_k\}$ is generated by the steepest descent algorithm with stepsizes chosen by backtracking. Then every limit point $\bar{x}$ of the sequence $\{x_k\}$ satisfies $\nabla f(\bar{x}) = 0$.*

[5] To make sure this is, in fact, without loss of generality, follow the steps of the proof carefully, and make sure that no generality is lost in making this assumption, i.e., in limiting the consideration of $\{x_k\}$ to just a subsequence that converges to $\bar{x}$.

Proof: Let x_k be the current iterate generated by the algorithm, λ_k – the stepsize, and x_{k+1} – the next iterate of the steepest descent algorithm. Using the mean value theorem, there is a value $\tilde{x}$, which is a convex combination of x_k and x_{k+1}, such that

$$\begin{aligned} f(x_{k+1}) - f(x_k) &= -\lambda_k \nabla f(x_k)^T \nabla f(\tilde{x}) \\ &= -\lambda_k \nabla f(x_k)^T (\nabla f(x_k) - \nabla f(x_k) + \nabla f(\tilde{x})) \\ &= -\lambda_k \|\nabla f(x_k)\|^2 + \lambda_k \nabla f(x_k)^T (\nabla f(x_k) - \nabla f(\tilde{x})) \\ &\leq -\lambda_k \|\nabla f(x_k)\|^2 + \lambda_k \|\nabla f(x_k)\| \cdot \|\nabla f(x_k) - \nabla f(\tilde{x})\| \\ &\leq -\lambda_k \|\nabla f(x_k)\|^2 + \lambda_k \|\nabla f(x_k)\| \cdot G \|x_k - \tilde{x}\| \\ &\leq -\lambda_k \|\nabla f(x_k)\|^2 + \lambda_k \|\nabla f(x_k)\| \cdot G \|x_k - x_{k+1}\| \\ &= -\lambda_k \|\nabla f(x_k)\|^2 + \lambda_k^2 G \|\nabla f(x_k)\|^2 \\ &= -\lambda_k \|\nabla f(x_k)\|^2 (1 - \lambda_k G) = -\frac{\bar{\lambda}}{2^t} \|\nabla f(x_k)\|^2 \left(1 - \frac{\bar{\lambda}}{2^t} G\right). \end{aligned} \tag{7}$$

The last equality uses the form of the stepsize generated by Armijo rule.

Using the fact that $\theta(\lambda) = f(x_k - \lambda \nabla f(x_k))$, the stopping condition of Armijo rule implementation, $\theta(\bar{\lambda}/2^t) \leq \hat{\theta}(\bar{\lambda}/2^t)$, can be rewritten as

$$f(x_{k+1}) \leq f(x_k) - (\epsilon \bar{\lambda}/2^t) \|\nabla f(x_k)\|^2.$$

Hence, t is the smallest integer such that

$$f(x_{k+1}) - f(x_k) \leq -\frac{\bar{\lambda}\epsilon}{2^t} \|\nabla f(x_k)\|^2. \tag{8}$$

Note that if t is large enough to satisfy

$$1 - \frac{\bar{\lambda} G}{2^t} \geq \epsilon, \tag{9}$$

then (7) would imply (8). Therefore, at the next-to-last step of the line search, (9) was not satisfied, i.e.,

$$1 - \frac{\bar{\lambda} G}{2^{t-1}} < \epsilon,$$

or, rearranging, $\frac{\bar{\lambda}\epsilon}{2^t} > \frac{\epsilon(1-\epsilon)}{2G}$. Substituting this into (8), we get

$$f(x_{k+1}) - f(x_k) < -\frac{\epsilon(1-\epsilon)}{2G} \|\nabla f(x_k)\|^2.$$

Since $\{f(x_k)\}$ is a monotone decreasing sequence, it has a limit (as long as $\{x_k\}$ has a limit point), taking limit as $k \to \infty$, we get

$$0 \leq \frac{-\epsilon(1-\epsilon)}{2G} \lim_{k \to \infty} \|\nabla f(x_k)\|^2,$$

implying that $\|\nabla f(x_k)\| \to 0$. ∎

7 Rate of convergence of steepest descent algorithm

7.1 Properties of quadratic forms

In this subsection we are going to analyze the properties of some of the simplest nonlinear functions — quadratic forms. The results developed here will be very useful to us in the future, when we analyze *local* properties of more complex functions, and the behavior of algorithms.

- A *quadratic form* is a function $f(x) = \frac{1}{2}x^TQx + q^Tx$.
- If Q is not symmetric, let $\bar{Q} = \frac{1}{2}(Q + Q^T)$. Then $\bar{Q}$ is symmetric, and $x^T\bar{Q}x = x^TQx$ for any x. So, we can assume wolog that Q is symmetric.
- $\nabla f(x) = Qx + q$
- $H(x) = Q$ — a constant matrix independent of x.

Proposition 16 *$f(x) = \frac{1}{2}x^TQx + q^Tx$ is convex iff $Q \succeq 0$.*

Proof: Follows from Theorem 11. ∎

Corollaries:
$f(x)$ is strictly convex iff $Q \succ 0$
$f(x)$ is concave iff $Q \preceq 0$
$f(x)$ is strictly concave iff $Q \prec 0$
$f(x)$ is neither convex nor concave iff Q is indefinite.

Examples of strictly convex quadratic forms:

- $f(x) = x^Tx$
- $f(x) = (x-a)^T(x-a)$
- $f(x) = (x-a)^TD(x-a)$, where $D = \begin{bmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{bmatrix}$ is diagonal with $d_j > 0$, $j = 1, \ldots, n$.
- $f(x) = (x-a)^TM^TDM(x-a)$, where M is a non-singular matrix and D is as above.

7.2 The rate of convergence of the steepest descent algorithm for the case of a quadratic function

In this section we explore answers to the question of how fast the steepest descent algorithm converges. We say that an algorithm exhibits *linear convergence* in the objective function values if there is a constant $\delta < 1$ such that for all k sufficiently large, the iterates x^k satisfy:

$$\frac{f(x^{k+1}) - f(x^\star)}{f(x^k) - f(x^\star)} \leq \delta,$$

where $x^\star$ is an optimal solution of the problem (P). The above statement says that the optimality gap shrinks by at least δ at each iteration. Notice that if $\delta = 0.1$, for example, then the iterates gain an extra digit of accuracy in the optimal objective function value at each iteration. If $\delta = 0.9$, for

example, then the iterates gain an extra digit of accuracy in the optimal objective function value every 22 iterations, since $(0.9)^{22} \approx 0.10$. The quantity δ above is called the *convergence constant*. We would like this constant to be smaller rather than larger.

We will show now that the steepest descent algorithm with stepsizes selected by exact line search exhibits linear convergence, but that the convergence constant depends very much on the ratio of the largest to the smallest eigenvalue of the Hessian matrix $H(x)$ at the optimal solution $x = x^\star$. In order to see how this dependence arises, we will examine the case where the objective function $f(x)$ is itself a simple quadratic function of the form:

$$f(x) = \frac{1}{2}x^T Q x + q^T x,$$

where Q is a positive definite symmetric matrix. We will suppose that the eigenvalues of Q are

$$A = a_1 \geq a_2 \geq \ldots \geq a_n = a > 0,$$

i.e, A and a are the largest and smallest eigenvalues of Q. The optimal solution of (P) is easily computed as:

$$x^\star = -Q^{-1}q$$

and direct substitution shows that the optimal objective function value is:

$$f(x^\star) = -\frac{1}{2}q^T Q^{-1} q.$$

For convenience, let x denote the current point in the steepest descent algorithm. We have:

$$f(x) = \frac{1}{2}x^T Q x + q^T x$$

and let d denote the current direction, which is the negative of the gradient, i.e.,

$$d = -\nabla f(x) = -Qx - q.$$

Now let us compute the next iterate of the steepest descent algorithm. If λ is the generic step-length, then

$$\begin{aligned} f(x + \lambda d) &= \frac{1}{2}(x + \lambda d)^T Q (x + \lambda d) + q^T (x + \lambda d) \\ &= \frac{1}{2}x^T Q x + \lambda d^T Q x + \frac{1}{2}\lambda^2 d^T Q d + q^T x + \lambda q^T d \\ &= f(x) - \lambda d^T d + \frac{1}{2}\lambda^2 d^T Q d. \end{aligned}$$

Optimizing over the value of λ in this last expression yields

$$\lambda = \frac{d^T d}{d^T Q d},$$

and the next iterate of the algorithm then is

$$x' = x + \lambda d = x + \frac{d^T d}{d^T Q d} d,$$

and

$$f(x') = f(x + \lambda d) = f(x) - \lambda d^T d + \frac{1}{2}\lambda^2 d^T Q d = f(x) - \frac{1}{2}\frac{(d^T d)^2}{d^T Q d}.$$

Therefore,

$$\begin{aligned}
\frac{f(x') - f(x^\star)}{f(x) - f(x^\star)} &= \frac{f(x) - \frac{1}{2}\frac{(d^T d)^2}{d^T Q d} - f(x^\star)}{f(x) - f(x^\star)} \\
&= 1 - \frac{\frac{1}{2}\frac{(d^T d)^2}{d^T Q d}}{\frac{1}{2}x^T Q x + q^T x + \frac{1}{2}q^T Q^{-1} q} \\
&= 1 - \frac{\frac{1}{2}\frac{(d^T d)^2}{d^T Q d}}{\frac{1}{2}(Qx + q)^t Q^{-1}(Qx + q)} \\
&= 1 - \frac{(d^T d)^2}{(d^T Q d)(d^T Q^{-1} d)} \\
&= 1 - \frac{1}{\beta},
\end{aligned}$$

where

$$\beta = \frac{(d^T Q d)(d^T Q^{-1} d)}{(d^T d)^2}.$$

In order for the convergence constant to be good, which will translate to fast linear convergence, we would like the quantity β to be small. The following result provides an upper bound on the value of β.

Kantorovich Inequality: Let A and a be the largest and the smallest eigenvalues of Q, respectively. Then

$$\beta \leq \frac{(A + a)^2}{4Aa}.$$

We will prove this inequality later. For now, let us apply this inequality to the above analysis. Continuing, we have

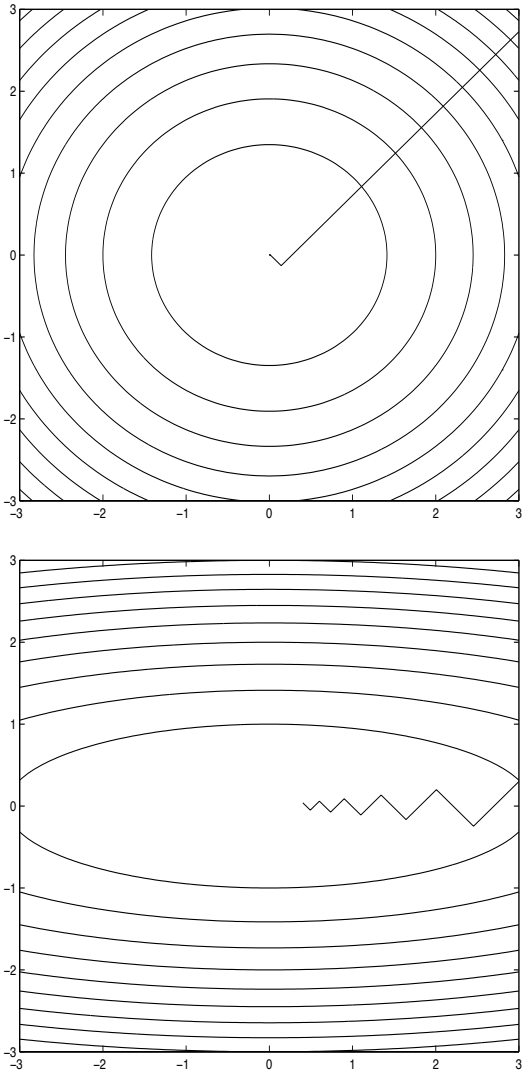
$$\frac{f(x') - f(x^\star)}{f(x) - f(x^\star)} = 1 - \frac{1}{\beta} \leq 1 - \frac{4Aa}{(A + a)^2} = \frac{(A - a)^2}{(A + a)^2} = \left(\frac{A/a - 1}{A/a + 1}\right)^2.$$

Note by definition that A/a is always at least 1. If A/a is small (not much bigger than 1), then the convergence constant will be much smaller than 1. However, if A/a is large, then the convergence constant will be only slightly smaller than 1. The following table shows some sample values:

A	a	Upper Bound on $1 - \frac{1}{\beta}$	Number of Iterations to Reduce the Optimality Gap by 0.10
1.1	1.0	0.0023	1
3.0	1.0	0.25	2
10.0	1.0	0.67	6
100.0	1.0	0.96	58
200.0	1.0	0.98	116
400.0	1.0	0.99	231

Note that the number of iterations needed to reduce the optimality gap by 0.10 grows linearly in the ratio A/a.

Two pictures of possible iterations of the steepest descent algorithm are as follows:



7.3 An example

Suppose that

$$f(x) = \frac{1}{2}x^T Q x + q^T x$$

where

$$Q = \begin{pmatrix} +4 & -2 \\ -2 & +2 \end{pmatrix} \quad \text{and} \quad q = \begin{pmatrix} +2 \\ -2 \end{pmatrix}.$$

Then

$$\nabla f(x) = \begin{pmatrix} +4 & -2 \\ -2 & +2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} +2 \\ -2 \end{pmatrix}$$

and so

$$x^\star = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

and

$$f(x^\star) = -1.$$

Direct computation shows that the eigenvalues of Q are $A = 3 + \sqrt{5}$ and $a = 3 - \sqrt{5}$, whereby the bound on the convergence constant is

$$1 - \frac{1}{\beta} \leq 0.556.$$

Suppose that $x_0 = (0, 0)$. Then we have:

$$x_1 = (-0.4, 0.4), \qquad x_2 = (0, 0.8),$$

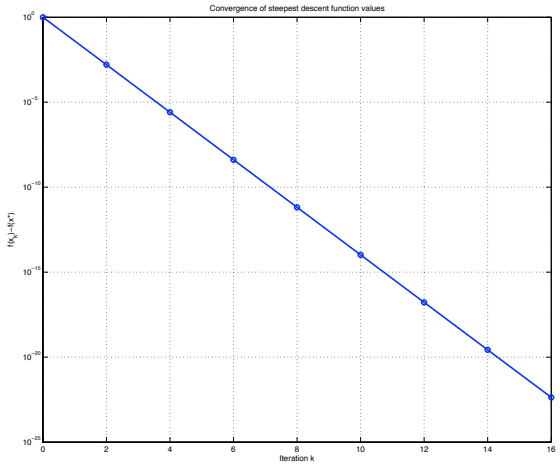
and the even numbered iterates satisfy

$$x_{2k} = (0, 1 - 0.2^k) \quad \text{and} \quad f(x_{2k}) = \left(1 - 0.2^k\right)^2 - 2 + 2(0.2)^k,$$

and so

$$\|x_{2k} - x^\star\| = 0.2^k \quad \text{and} \quad f(x_{2k}) - f(x^\star) = (0.2)^{2k}.$$

Therefore, starting from the point $x_0 = (0, 0)$, the optimality gap goes down by a factor of 0.04 after every two iterations of the algorithm. This convergence is illustrated in this picture (note that the Y-axis is in logarithmic scale!).



Some remarks:

- The bound on the rate of convergence *is* attained in practice quite often, which is unfortunate. The ratio of the largest to the smallest eigenvalue of a matrix is called the *condition number* of the matrix.
- What about non-quadratic functions? Most functions behave as near-quadratic functions in a neighborhood of the optimal solution. The analysis of the non-quadratic case gets very involved; fortunately, the key intuition is obtained by analyzing the quadratic case.

Practical termination criteria Ideally, the algorithm will terminate at a point x^k such that $\nabla f(x^k) = 0$. However, the algorithm is not guaranteed to be able to find such point in finite amount of time. Moreover, due to rounding errors in computer calculations, the calculated value of the gradient will have some imprecision in it.

Therefore, in practical algorithms the termination criterion is designed to test if the above condition is satisfied approximately, so that the resulting output of the algorithm is an approximately optimal solution. A natural termination criterion for the steepest descent could be $\|\nabla f(x^k)\| \leq \epsilon$, where $\epsilon > 0$ is a pre-specified tolerance. However, depending on the scaling of the function, this requirement can be either unnecessarily stringent, or too loose to ensure near-optimality (consider a problem concerned with minimizing distance, where the objective function can be expressed in inches, feet, or miles). Another alternative, that might alleviate the above consideration, is to terminate when $\|\nabla f(x^k)\| \leq \epsilon |f(x^k)|$ — this, however, may lead to problems when the objective function at the optimum is zero. A combined approach is then to terminate when

$$\|\nabla f(x^k)\| \leq \epsilon(1 + |f(x^k)|).$$

The value of ϵ is typically taken to be at most the square root of the machine tolerance (e.g., $\epsilon = 10^{-8}$ if 16-digit computing is used), due to the error incurred in estimating derivatives.

7.4 Proof of Kantorovich Inequality

Kantorovich Inequality: Let A and a be the largest and the smallest eigenvalues of Q, respectively. Then

$$\beta = \frac{(d^T Q d)(d^T Q^{-1} d)}{(d^T d)^2} \leq \frac{(A+a)^2}{4Aa}.$$

Proof: Let $Q = RDR^T$, and then $Q^{-1} = RD^{-1}R^T$, where $R = R^T$ is an orthonormal matrix, and the eigenvalues of Q are

$$0 < a = a_1 \leq a_2 \leq \ldots \leq a_n = A,$$

and

$$D = \begin{pmatrix} a_1 & 0 & \ldots & 0 \\ 0 & a_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_n \end{pmatrix}.$$

Then

$$\beta = \frac{(d^T RDR^T d)(d^T RD^{-1}R^T d)}{(d^T RR^T d)(d^T RR^T d)} = \frac{f^T Df \cdot f^T D^{-1} f}{f^T f \cdot f^T f}$$

where $f = R^T d$. Let $\lambda_i = \frac{f_i^2}{f^T f}$. Then $\lambda_i \geq 0$ and $\sum_{i=1}^n \lambda_i = 1$, and

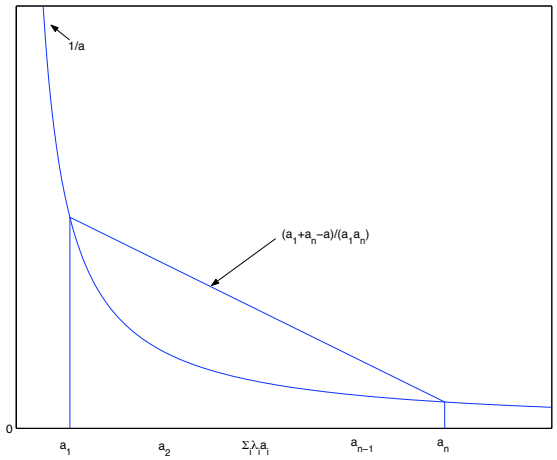
$$\beta = \sum_{i=1}^n \lambda_i a_i \cdot \sum_{i=1}^n \lambda_i \left(\frac{1}{a_i}\right) = \frac{\sum_{i=1}^n \lambda_i \left(\frac{1}{a_i}\right)}{\left(\frac{1}{\sum_{i=1}^n \lambda_i a_i}\right)}.$$

The largest value of β is attained when $\lambda_1 + \lambda_n = 1$ (see the following illustration to see why this must be true). Therefore,

$$\beta \leq \frac{\lambda_1 \frac{1}{a} + \lambda_n \frac{1}{A}}{\frac{1}{\lambda_1 a + \lambda_n A}} = \frac{(\lambda_1 a + \lambda_n A)(\lambda_1 A + \lambda_n a)}{Aa} \leq \frac{(\frac{1}{2}A + \frac{1}{2}a)(\frac{1}{2}a + \frac{1}{2}A)}{Aa} = \frac{(A+a)^2}{4Aa}.$$

■

Illustration of Kantorovich construction:



8 Newton's method for minimization

Again, we want to solve

$$\text{(P)} \quad \begin{array}{l} \min f(x) \\ x \in \mathbb{R}^n. \end{array}$$

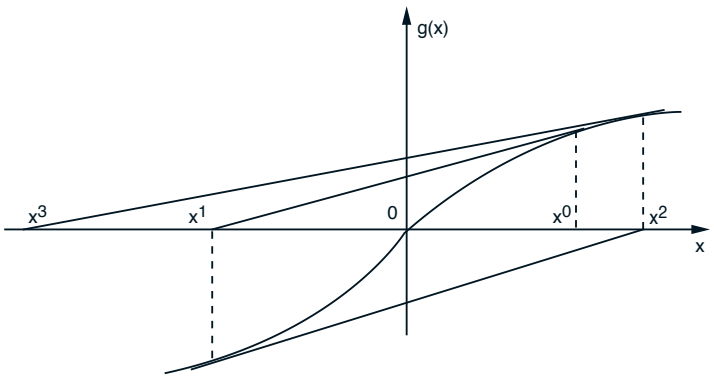
The Newton's method can also be interpreted in the framework of the general optimization algorithm, but it truly stems from the Newton's method for solving systems of nonlinear equations. Recall that if $g : \mathbb{R}^n \to \mathbb{R}^n$, to solve the system of equations

$$g(x) = 0,$$

one can apply an iterative method. Starting at a point x_0, approximate the function g by $g(x_0+d) \approx g(x_0) + \nabla g(x_0)^T d$, where $\nabla g(x_0)^T \in \mathbb{R}^{n\times n}$ is the Jacobian of g at x_0, and provided that $\nabla g(x_0)$ is non-singular, solve the system of linear equations

$$\nabla g(x_0)^T d = -g(x_0)$$

to obtain d. Set the next iterate $x_1 = x_0 + d$, and continue. This method is well-studied, and is well-known for its good performance when the starting point x_0 is chosen appropriately. However, for other choices of x_0 the algorithm may not converge, as demonstrated in the following well-known picture:



The Newton's method for minimization is precisely an application of this equation-solving method to the (system of) first-order optimality conditions $\nabla f(x) = 0$. As such, the algorithm does not distinguish between local minimizers, maximizers, or saddle points.

Here is another view of the motivation behind the Newton's method for optimization. At $x = \bar{x}$, $f(x)$ can be approximated by

$$f(x) \approx q(x) \stackrel{\triangle}{=} f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) + \frac{1}{2}(x - \bar{x})^T H(\bar{x})(x - \bar{x}),$$

which is the quadratic Taylor expansion of $f(x)$ at $x = \bar{x}$. $q(x)$ is a quadratic function which, if it is convex, is minimized by solving $\nabla q(x) = 0$, i.e., $\nabla f(\bar{x}) + H(\bar{x})(x - \bar{x}) = 0$, which yields

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

The direction $-H(\bar{x})^{-1}\nabla f(\bar{x})$ is called the *Newton direction*, or the *Newton step*.

This leads to the following algorithm for solving (P):

Newton's Method:

Step 0 Given x_0, set $k \leftarrow 0$

Step 1 $d_k = -H(x_k)^{-1}\nabla f(x_k)$. If $d_k = 0$, then stop.

Step 2 Choose stepsize $\lambda_k = 1$.

Step 3 Set $x_{k+1} \leftarrow x_k + \lambda_k d_k,\ k \leftarrow k+1$. Go to **Step 1**.

Proposition 17 *If $H(x) \succ 0$, then $d = -H(x)^{-1}\nabla f(x)$ is a descent direction.*

Proof: It is sufficient to show that $\nabla f(x)^T d = -\nabla f(x)^T H(x)^{-1}\nabla f(x) < 0$. Since $H(x)$ is positive definite, if $v \neq 0$,

$$0 < (H(x)^{-1}v)^T H(x)(H(x)^{-1}v) = v^T H(x)^{-1} v,$$

completing the proof. ∎

Note that:

- Work per iteration: $O(n^3)$
- The iterates of Newton's method are, in general, equally attracted to local minima and local maxima. Indeed, the method is just trying to solve the system of equations $\nabla f(x) = 0$.
- There is no guarantee that $f(x_{k+1}) \leq f(x_k)$.
- Step 2 could be augmented by a linesearch of $f(x_k + \lambda d_k)$ over the value of λ; then previous consideration would not be an issue.
- The method assumes $H(x_k)$ is nonsingular at each iteration. Moreover, unless $H(x_k)$ is positive definite, d_k is not guaranteed to be a descent direction.
- What if $H(x_k)$ becomes increasingly singular (or not positive definite)? Use $H(x_k) + \epsilon I$.
- In general, points generated by the Newton's method as it is described above, may not converge. For example, $H(x_k)^{-1}$ may not exist. Even if $H(x)$ is always non-singular, the method may not converge, unless started "close enough" to the right point.

Example 1: Let $f(x) = 7x - \ln(x)$. Then $\nabla f(x) = f'(x) = 7 - \frac{1}{x}$ and $H(x) = f''(x) = \frac{1}{x_2}$. It is not hard to check that $x^\star = \frac{1}{7} = 0.142857143$ is the unique global minimizer. The Newton direction at x is

$$d = -H(x)^{-1}\nabla f(x) = -\frac{f'(x)}{f''(x)} = -x_2\left(7 - \frac{1}{x}\right) = x - 7x_2,$$

and is defined so long as $x > 0$. So, Newton's method will generate the sequence of iterates $\{x_k\}$ with $x_{k+1} = x_k + (x_k - 7(x_k)^2) = 2x_k - 7(x_k)^2$. Below are some examples of the sequences generated

by this method for different starting points:

k	x_k	x_k	x_k
0	1	0.1	0.01
1	-5	0.13	0.0193
2		0.1417	0.03599257
3		0.14284777	0.062916884
4		0.142857142	0.098124028
5		0.142857143	0.128849782
6			0.1414837
7			0.142843938
8			0.142857142
9			0.142857143
10			0.142857143

(note that the iterate in the first column is not in the domain of the objective function, so the algorithm has to terminate...).

Example 2: $f(x) = -\ln(1 - x_1 - x_2) - \ln x_1 - \ln x_2$.

$$\nabla f(x) = \begin{bmatrix} \frac{1}{1-x_1-x_2} - \frac{1}{x_1} \\ \frac{1}{1-x_1-x_2} - \frac{1}{x_2} \end{bmatrix},$$

$$H(x) = \begin{bmatrix} \left(\frac{1}{1-x_1-x_2}\right)^2 + \left(\frac{1}{x_1}\right)^2 & \left(\frac{1}{1-x_1-x_2}\right)^2 \\ \left(\frac{1}{1-x_1-x_2}\right)^2 & \left(\frac{1}{1-x_1-x_2}\right)^2 + \left(\frac{1}{x_2}\right)^2 \end{bmatrix}.$$

$x^\star = \left(\frac{1}{3}, \frac{1}{3}\right)$, $f(x^\star) = 3.295836866$.

k	$(x_k)_1$	$(x_k)_2$	$\|x_k - \bar{x}\|$
0	0.85	0.05	0.58925565098879
1	0.717006802721088	0.0965986394557823	0.450831061926011
2	0.512975199133209	0.176479706723556	0.238483249157462
3	0.352478577567272	0.273248784105084	0.0630610294297446
4	0.338449016006352	0.32623807005996	0.00874716926379655
5	0.333337722134802	0.333259330511655	$7.41328482837195e^{-5}$
6	0.333333343617612	0.33333332724128	$1.19532211855443e^{-8}$
7	0.333333333333333	0.333333333333333	$1.57009245868378e^{-16}$

Termination criteria Since Newton's method is working with the Hessian as well as the gradient, it would be natural to augment the termination criterion we used in the Steepest Descent algorithm with the requirement that $H(x_k)$ is positive semi-definite, or, taking into account the potential for the computational errors, that $H(x_k) + \epsilon I$ is positive semi-definite for some $\epsilon > 0$ (this parameter may be different than the one used in the condition on the gradient).

8.1 Convergence analysis of Newton's method

8.1.1 Rate of convergence

Suppose we have a *converging* sequence $\lim_{k\to\infty} s_k = \bar{s}$, and we would like to characterize the speed, or rate, at which the iterates s_k approach the limit $\bar{s}$.

A converging sequence of numbers $\{s_k\}$ exhibits *linear* convergence if for some $0 \leq C < 1$,

$$\limsup_{k\to\infty} \frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|} = C.$$

"$\limsup_{k\to\infty}$" denotes the largest of the limit points of a sequence (possibly infinite). C in the above expression is referred to as the *rate constant*; if $C = 0$, the sequence exhibits *superlinear* convergence.

A sequence of numbers $\{s_k\}$ exhibits *quadratic* convergence if it converges to some limit $\bar{s}$ and

$$\limsup_{k\to\infty} \frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|^2} = \delta < \infty.$$

Examples:

Linear convergence $s_k = \left(\frac{1}{10}\right)^k$: 0.1, 0.01, 0.001, etc. $\bar{s} = 0$.

$$\frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|} = 0.1.$$

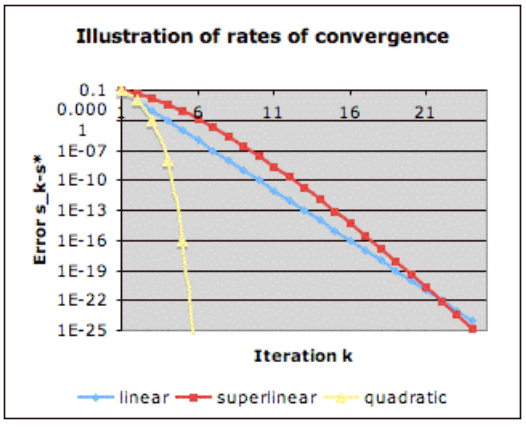
Superlinear convergence $s_k = 0.1 \cdot \frac{1}{k!}$: $\frac{1}{10}$, $\frac{1}{20}$, $\frac{1}{60}$, $\frac{1}{240}$, $\frac{1}{1250}$, etc. $\bar{s} = 0$.

$$\frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|} = \frac{k!}{(k+1)!} = \frac{1}{k+1} \to 0 \text{ as } k \to \infty.$$

Quadratic convergence $s_k = \left(\frac{1}{10}\right)^{(2^{k-1})}$: 0.1, 0.01, 0.0001, 0.00000001, etc. $\bar{s} = 0$.

$$\frac{|s_{k+1} - \bar{s}|}{|s_k - \bar{s}|^2} = \frac{(10^{2^{k-1}})^2}{10^{2^k}} = 1.$$

This illustration compares the rates of convergence of the above sequences:



Since an algorithm for nonlinear optimization problems, in its abstract form, generates an *infinite* sequence of points $\{x_k\}$ converging to a solution $\bar{x}$ only in the limit, it makes sense to discuss the rate of convergence of the sequence $\|e^k\| = \|x_k - \bar{x}\|$, or $E^k = |f(x_k) - f(\bar{x})|$, which both have limit 0. For example, in the previous section we've shown that, on a convex quadratic function, the steepest descent algorithm exhibits linear convergence, with rate bounded by the condition number of the Hessian. For non-quadratic functions, the steepest descent algorithm behaves similarly in the limit, i.e., once the iterates reach a small neighborhood of the limit point.

8.1.2 Rate of convergence of the pure Newton's method

We have seen from our examples that, even for convex functions, the Newton's method in its pure form (i.e., with stepsize of 1 at every iteration) does not guarantee descent at each iteration, and may produce a diverging sequence of iterates. Moreover, each iteration of the Newton's method is much more computationally intensive then that of the steepest descent. However, under certain conditions, the method exhibits *quadratic* rate of convergence, making it the "ideal" method for solving convex optimization problems. Recall that a method exhibits quadratic convergence when $\|e_k\| = \|x_k - \bar{x}\| \to 0$ and

$$\lim_{k\to\infty} \frac{\|e^{k+1}\|}{\|e^k\|^2} = C.$$

Roughly speaking, if the iterates converge quadratically, the accuracy (i.e., the number of correct digits) of the solution doubles in a fixed number of iterations.

There are many ways to state and prove results regarding the convergence on the Newton's method. We provide one that provides a particular insight into the circumstances under which pure Newton's method demonstrates quadratic convergence (compare the theorem below to BSS 8.6.5).

Let $\|v\|$ denote the usual Euclidian norm of a vector, namely $\|v\| := \sqrt{v^T v}$. Recall that the operator norm of a matrix M is defined as follows:

$$\|M\| := \max_x \{\|Mx\| : \|x\| = 1\}.$$

As a consequence of this definition, for any x, $\|Mx\| \le \|M\| \cdot \|x\|$.

Theorem 18 (Quadratic convergence) *Suppose $f(x)$ is twice continuously differentiable and $x^\star$ is a point for which $\nabla f(x^\star) = 0$. Suppose $H(x)$ satisfies the following conditions:*

- *there exists a scalar $h > 0$ for which $\|[H(x^\star)]^{-1}\| \le \frac{1}{h}$*
- *there exists scalars $\beta > 0$ and $L > 0$ for which $\|H(x) - H(y)\| \le L\|x - y\|$ for all x and y satisfying $\|x - x^\star\| \le \beta$ and $\|y - x^\star\| \le \beta$.*

Let x satisfy $\|x - x^\star\| \le \delta\gamma$, where $0 < \delta < 1$ and $\gamma := \min\left\{\beta, \frac{2h}{3L}\right\}$, and let $x_N := x - H(x)^{-1}\nabla f(x)$. Then:

(i) $\|x_N - x^\star\| \le \|x - x^\star\|^2 \left(\frac{L}{2(h - L\|x - x^\star\|)}\right)$

(ii) $\|x_N - x^\star\| < \delta\|x - x^\star\|$, *and hence the iterates converge to $x^\star$*

(iii) $\|x_N - x^\star\| \le \|x - x^\star\|^2 \left(\frac{3L}{2h}\right)$.

The proof relies on the following two "elementary" facts.

Proposition 19 *Suppose that M is a symmetric matrix. Then the following are equivalent:*

1. *$h > 0$ satisfies $\|M^{-1}\| \le \frac{1}{h}$*
2. *$h > 0$ satisfies $\|Mv\| \ge h \cdot \|v\|$ for any vector v*

Proof: Left as an exercise. ∎

Proposition 20 *Suppose that $f(x)$ is twice differentiable. Then*

$$\nabla f(z) - \nabla f(x) = \int_0^1 [H(x + t(z - x))]\,(z - x)dt\ .$$

Proof: Let $\phi(t) := \nabla f(x + t(z - x))$. Then $\phi(0) = \nabla f(x)$ and $\phi(1) = \nabla f(z)$, and $\phi^{'}(t) = [H(x + t(z - x))]\,(z - x)$. From the fundamental theorem of calculus, we have:

$$\begin{aligned}\nabla f(z) - \nabla f(x) &= \phi(1) - \phi(0) \\ &= \int_0^1 \phi^{'}(t)dt \\ &= \int_0^1 [H(x + t(z - x))]\,(z - x)dt \ . \blacksquare\end{aligned}$$

Proof of Theorem 18

We have:

$$\begin{aligned}x_N - x^\star &= x - H(x)^{-1}\nabla f(x) - x^\star \\ &= x - x^\star + H(x)^{-1}\left(\nabla f(x^\star) - \nabla f(x)\right) \\ &= x - x^\star + H(x)^{-1}\int_0^1 [H(x + t(x^\star - x))]\,(x^\star - x)dt \text{ (from Proposition 20)} \\ &= H(x)^{-1}\int_0^1 [H(x + t(x^\star - x)) - H(x)]\,(x^\star - x)dt.\end{aligned}$$

Therefore

$$\begin{aligned}\|x_N - x^\star\| &\leq \|H(x)^{-1}\|\int_0^1 \|\,[H(x + t(x^\star - x)) - H(x)]\,\| \cdot \|(x^\star - x)\|dt \\ &\leq \|x^\star - x\| \cdot \|H(x)^{-1}\|\int_0^1 L \cdot t \cdot \|(x^\star - x)\|dt \\ &= \|x^\star - x\|^2\|H(x)^{-1}\|L\int_0^1 tdt \\ &= \frac{\|x^\star - x\|^2\|H(x)^{-1}\|L}{2}.\end{aligned}$$

We now bound $\|H(x)^{-1}\|$. Let v be any vector. Then

$$\begin{aligned}\|H(x)v\| &= \|H(x^\star)v + (H(x) - H(x^\star))v\| \\ &\geq \|H(x^\star)v\| - \|(H(x) - H(x^\star))v\| \\ &\geq h \cdot \|v\| - \|H(x) - H(x^\star)\|\|v\| \text{ (from Proposition 19)} \\ &\geq h \cdot \|v\| - L\|x^\star - x\| \cdot \|v\| \\ &= (h - L\|x^\star - x\|) \cdot \|v\| \ .\end{aligned}$$

Invoking Proposition 19 again, we see that this implies that

$$\|H(x)^{-1}\| \leq \frac{1}{h - L\|x^\star - x\|} \ .$$

Combining this with the above yields

$$\|x_N - x^\star\| \leq \|x^\star - x\|^2 \cdot \frac{L}{2\,(h - L\|x^\star - x\|)} \ ,$$

which is *(i)* of the theorem. Because $L\|x^\star - x\| \leq \delta \cdot \frac{2h}{3} < \frac{2h}{3}$ we have:

$$\|x_N - x^\star\| \leq \|x^\star - x\| \cdot \frac{L\|x^\star - x\|}{2\left(h - L\|x^\star - x\|\right)} \leq \frac{\delta \cdot \frac{2h}{3}}{2\left(h - \frac{2h}{3}\right)}\|x^\star - x\| = \delta\|x^\star - x\| \ ,$$

which establishes *(ii)* of the theorem. Finally, we have

$$\|x_N - x^\star\| \leq \|x^\star - x\|^2 \frac{L}{2\left(h - L\|x^\star - x\|\right)} \leq \|x^\star - x\|^2 \cdot \frac{L}{2\left(h - \frac{2h}{3}\right)} = \frac{3L}{2h}\|x^\star - x\|^2 \ ,$$

which establishes *(iii)* of the theorem. ∎

Notice that the results regarding the convergence and rate of convergence in the above theorem are *local*, i.e., they apply only if the algorithm is initialized at certain starting points (the ones "sufficiently close" to the desired limit). In practice, it is not known how to pick such starting points, or to check if the proposed starting point is adequate. (With the very important exception of self-concordant functions.)

8.2 Further discussion and modifications of the Newton's method

8.2.1 Global convergence for strongly convex functions with a two-phase Newton's method

We have noted that, to ensure descent at each iteration, the Newton's method can be augmented by a line search. This idea can be formalized, and the efficiency of the resulting algorithm can be analyzed (see, for example, "Convex Optimization" by Stephen Boyd and Lieven Vandenberghe, available at `http://www.stanford.edu/~boyd/cvxbook.html` for a fairly simple presentation of the analysis).

Suppose that $f(x)$ is *strongly convex* on it domain, i.e., assume there exists $\mu > 0$ such that $H(x) - \mu I$ is positive semidefinite for all x (I is the identity matrix), and that the Hessian is Lipschitz continuous everywhere on the domain of f. Suppose we apply the Newton's method with the stepsize at each iteration determined by the backtracking procedure of section 5.2.2. That is, at each iteration of the algorithm we first attempt to take a full Newton step, but reduce the stepsize if the decrease in the function value is not sufficient. Then there exist positive numbers η and γ such that

- if $\|\nabla f(x_k)\| \geq \eta$, then $f(x_{k+1}) - f(x_k) \leq -\gamma$, and
- if $\|\nabla f(x_k)\| < \eta$, then stepsize $\lambda_k = 1$ will be selected, and the next iterate will satisfy $\|\nabla f(x_{k+1})\| < \eta$, and so will all the further iterates. Moreover, quadratic convergence will be observed in this phase.

As hinted above, the algorithm will proceed in two phases: while the iterates are far from the minimizer, a "dampening" of the Newton step will be required, but there will be a guaranteed decrease in the objective function values. This phase (referred to as "dampened Newton phase") cannot take more than $\frac{f(x_0) - f(x^\star)}{\gamma}$ iterations. Once the norm of the gradient becomes sufficiently small, no dampening of the Newton step will required in the rest of the algorithm, and quadratic convergence will be observed, thus making it the "quadratically convergence phase."

Note that it is not necessary to know the values of η and γ to apply this version of the algorithm! The two-phase Newton's method is globally convergent; however, to ensure global convergence, the function being minimized needs to posses particularly nice *global* properties.

8.2.2 Other modifications of the Newton's method

We have seen that if Newton's method is initialized sufficiently close to the point $\bar{x}$ such that $\nabla f(\bar{x}) = 0$ and $H(\bar{x})$ is positive definite (i.e., $\bar{x}$ is a local minimizer), then it will converge quadratically, using stepsizes of $\lambda = 1$. There are three issues in the above statement that we should be concerned with:

- What if $H(\bar{x})$ is singular, or nearly-singular?
- How do we know if we are "close enough," and what to do if we are not?
- Can we modify Newton's method to guarantee global convergence?

In the previous subsection we "assumed away" the first issue, and, under an additional assumption, showed how to address the other two. What if the function f is not strongly convex, and $H(x)$ may approach singularity?

There are two popular approaches (which are actually closely related) to address these issues. The first approach ensures that the method always uses a descent direction. For example, instead of the direction $-H(x_k)^{-1}\nabla f(x_k)$, use the direction $-(H(x_k)+\epsilon_k I)^{-1}\nabla f(x_k)$, where $\epsilon_k \geq 0$ is chosen so that the smallest eigenvalue of $H(x_k) + \epsilon_k I$ is bounded below by a fixed number $\delta > 0$. It is important to choose the value of δ appropriately — if it is chosen to be too small, the matrix employed in computing the direction can become ill-conditioned if $H(\bar{x})$ is nearly singular; if it is chosen to be too large, the direction becomes nearly that of the steepest descent algorithm, and hence only linear convergence can be guaranteed. Hence, the value of ϵ_k is often chosen dynamically.

The second approach is the so-called *trust region* method. Note that the main idea behind the Newton's method is to represent the function $f(x)$ by its quadratic approximation $q_k(x) = f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$ around the current iterate, and then minimize that approximation. While locally the approximation is a good one, this may no longer be the case when a large step is taken. The trust region methods hence find the next iterate by solving the following *constrained* optimization problem:

$$\min q_k(x) \text{ s.t. } \|x - x_k\| \leq \Delta_k$$

(as it turns out, this problem is not much harder to solve than the unconstrained minimization of $q_k(s)$).

The value of Δ_k is set to represent the size of the region in which we can "trust" $q_k(x)$ to provide a good approximation of $f(x)$. Smaller values of Δ_k ensure that we are working with an accurate representation of $f(x)$, but result in conservative steps. Larger values of Δ_k allow for larger steps, but may lead to inaccurate estimation of the objective function. To account for this, the value if Δ_k is updated dynamically throughout the algorithm, namely, it is increased if it is observed that $q_k(x)$ provided an exceptionally good approximation of $f(x)$ at the previous iteration, and decreased is the approximation was exceptionally bad.

8.3 Quasi-Newton (secant) methods

Quasi-Newton methods are probably the most popular general-purpose algorithms for unconstrained optimization. The basic idea behind quasi-Newton methods is quite simple. A typical iteration of the method is

$$x_{k+1} = x_k + \lambda_k d_k, \text{ where } d_k = -D_k \nabla f(x_k),$$

where D_k is a positive definite matrix (which is adjusted from iteration to iteration) chosen so that the directions d_k tend to approximate Newton's direction. The stepsize λ_k is usually chosen by a line search.

Many quasi-Newton methods are advantageous due to their fast convergence and absence of second-order derivative computation.

8.3.1 The Broyden family

Of course, what makes a quasi-Newton method work is the choice of the matrix D_k at each iteration. The important idea behind the methods is that two successive iterates x_k and x_{k+1} together with the gradients $\nabla f(x_k)$ and $\nabla f(x_{k+1})$ contain curvature (i.e., Hessian) information, in particular,

$$(\nabla f(x_{k+1}) - \nabla f(x_k)) \approx H(x_{k+1})(x_{k+1} - x_k)$$

(observe that the above approximation is an *equality* when the function in question is quadratic). Therefore, at every iteration we would like to choose D_{k+1} to satisfy

$$D_{k+1} q_k = p_k, \text{ where } p_k = x_{k+1} - x_k, \; q_k = \nabla f(x_{k+1}) - \nabla f(x_k). \tag{10}$$

Equation (10) is known as the quasi-Newton condition, or the secant equation.

Suppose that at every iteration we update the matrix D_{k+1} by taking the matrix D_k and adding a "correction" term C_k. Then the secant equation becomes

$$(D_k + C_k) q_k = p_k \;\Rightarrow\; C_k q_k = p_k - D_k q_k. \tag{11}$$

Note that equation (11) leaves us a lot of flexibility in selecting the correction matrix C_k. The most popular update methods come from the following (parametric) family of matrices (the subscript k is omitted in most of the following formulas for simplicity, here $D = D_k$):

$$C^{\mathrm{B}}(\phi) = \frac{pp^T}{p^T q} - \frac{Dqq^T D}{q^T D q} + \phi \tau v v^T, \text{ where } v = \frac{p}{p^T q} - \frac{Dq}{\tau}, \; \tau = q^T D q \tag{12}$$

(it is not hard to verify that these updates indeed satisfy the secant equation).

The choice of the scalar $\phi \in [0, 1]$, which parameterizes the family of matrices C, gives rise to several popular choices of updates. In particular:

- Setting $\phi = 0$ at each iteration, we obtain the so-called DFP (Davidson-Fletcher-Powell) update:
$$C^{\mathrm{DFP}} = C^{\mathrm{B}}(0) = \frac{pp^T}{p^T q} - \frac{Dqq^T D}{q^T D q},$$
which is historically the first quasi-Newton method.

- Setting $\phi = 1$ at each iteration, we obtain the BFGS (Broyden-Fletcher-Goldfarb-Shanno) update:
$$C^{\mathrm{BFGS}} = C^{\mathrm{B}}(1) = \frac{pp^T}{p^Tq}\left[1 + \frac{q^TDq}{p^Tq}\right] - \frac{Dqp^T + pq^TD}{p^Tq}.$$
The resulting method has been shown to be superior to other updating schemes in its overall performance.
- A general member of the Broyden family (12) can therefore be written as a convex combination of the two above updates:
$$C^{\mathrm{B}}(\phi) = (1-\phi)C^{\mathrm{DFP}} + \phi C^{\mathrm{BFGS}}$$

The following two results demonstrate that quasi-Newton methods generate descent search directions (as long as exact line searches are performed, and the initial approximation D_1 is positive definite), and, when applied to a quadratic function, converge in finite number of iterations.

Proposition 21 *If D_k is positive definite and the stepsize λ_k is chosen so that x_{k+1} satisfies*
$$(\nabla f(x_k) - \nabla f(x_{k+1})^T d_k < 0,$$
then D_{k+1} given by (12) is positive definite (and hence d_{k+1} is a descent direction).

Note that if exact line search is performed, $\nabla f(x_{k+1})^T d_k = 0$, so that condition above is satisfied.

Proposition 22 *If the quasi-Newton method with matrices D_k generated by (12) is applied to minimization of a positive-definite quadratic function $f(x) = \frac{1}{2}x^TQx - q^Tx$, then $D_{n+1} = Q^{-1}$.*

(In fact, for a quadratic function, the vectors d^i, $i = 1, \ldots, n$ are so-called Q-conjugate directions, and, if $D_1 = I$, the method actually coincides with the so-called conjugate gradient algorithm.)

8.3.2 BFGS method

An alternative to maintaining the matrix D_k above which approximates the inverse of the Hessian, one could maintain a positive definite matrix B_k which would approximate the Hessian itself. Viewed from this perspective, the secant equation can be written as
$$q_k = B_{k+1}p_k = (B_k + \tilde{C}_k)p_k \;\Rightarrow\; \tilde{C}_kp_k = q_k - B_kp_k,$$
where $\tilde{C}$ is the "correction" matrix in this setting. Analogously to $C^{\mathrm{B}}(\phi)$, one can construct a parametric family of update matrices
$$\tilde{C}(\phi) = \frac{qq^T}{q^Tp} - \frac{Bpp^TB}{p^TBp} + \phi\tau vv^T, \text{ where } v = \frac{q}{q^Tp} - \frac{Bp}{\tau},\ \tau = q^TBq.$$
Using $\phi = 0$, we obtain the update used in the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method:
$$\tilde{C}^{\mathrm{BFGS}} = \frac{qq^T}{q^Tp} - \frac{Bpp^TB}{p^TBp}.$$
(If it seems like we have two different objects referred to as "the BFGS," fear not — in fact, if $D = B^{-1}$, then $D + C^{\mathrm{BFGS}} = (B + \tilde{C}^{\mathrm{BFGS}})^{-1}$, so the two BFGS updates are consistent with each other.)

When using this method, the search direction at each iteration has to be obtained by solving the system of equations

$$B_{k+1}d_{k+1} = -\nabla f(x_{k+1}).$$

It may appear that this will require a lot of computation, and approximating the inverse of the Hessian directly (as was done in the previous subsection) is a better approach. However, obtaining solutions to these systems is implemented quite efficiently by maintaining the so-called Cholesky factorization of the matrix $B_{k+1} = L\Lambda L^T$ (here L is a lower triangular matrix, and Λ is a diagonal matrix). This factorization is easy to implement and update and is numerically stable. In addition, if the line searches are not performed to sufficient precision (in particular, the resulting iterates do not satisfy the conditions of Proposition 21), the matrices D_k are not guaranteed to be positive definite. This would be fairly hard to detect, and can lead to bad choice of direction d_k. On the other hand, maintaining the Cholesky factorization of the matrix B_k immediately allows us to check the signs of eigenvalues of B_k (just look at the elements of Λ), and if needed add a correction term to maintain positive definiteness.

8.3.3 A final note

Quasi-Newton methods (typically with BFGS update of one form of another) are usually the algorithms of choice in unconstrained optimization software.

The optimization toolbox in MATLAB implements quite a few gradient descent methods in its function `fminunc`. The default method for small-to-medium size problems is the BFGS method (with update $\tilde{C}^{\mathrm{BFGS}}$). The formula for gradient of the function f can be provided as a subroutine; if not available, the gradients will be approximated numerically.

The software allows you to change the algorithm used to DFP quasi-Newton method which approximates the inverse of the Hessian, or to steepest descent (the later, however, is "not recommended" by the manual).

For large-scaled problems, MATLAB implements a version of Newton's algorithms. An interesting aspect of the implementation (as is the case with many implementations of Newton's method) is that the computation of the direction $d = -H(x)^{-1}\nabla f(x)$ is often performed approximately by applying a few iterations of the above-mentioned conjugate gradient algorithm to solve the linear system $H(x)d = -\nabla f(x)$.

9 Constrained optimization — optimality conditions

9.1 Introduction

Recall that a constrained optimization problem is a problem of the form

$$\begin{array}{lll} \text{(P)} & \min & f(x) \\ & \text{s.t.} & g(x) \leq 0 \\ & & h(x) = 0 \\ & & x \in X, \end{array}$$

where X is an open set and $g(x) = (g_1(x), \ldots, g_m(x))^T : \mathbb{R}^n \to \mathbb{R}^m$, $h(x) = (h_1(x), \ldots, h_l(x))^T : \mathbb{R}^n \to \mathbb{R}^l$. Let S denote the *feasible set* of (P), i.e.,

$$S \stackrel{\triangle}{=} \{x \in X : g(x) \leq 0,\ h(x) = 0\}.$$

Then the problem (P) can be written as $\min_{x \in S} f(x)$. Recall that $\bar{x}$ is a local minimizer of (P) if $\exists \epsilon > 0$ such that $f(\bar{x}) \leq f(y)$ for all $y \in S \cap B_\epsilon(\bar{x})$. Local, global minimizers and maximizers, strict and non-strict, are defined analogously.

We will often use the following "shorthand" notation:

$$\nabla g(x) = [\nabla g_1(x), \quad \nabla g_2(x), \quad \cdots \quad \nabla g_m(x)] \text{ and } \nabla h(x) = [\nabla h_1(x), \quad \nabla h_2(x), \quad \cdots \quad \nabla g_l(x)],$$

i.e., $\nabla g(x)^T \in \mathbb{R}^{m \times n}$ and $\nabla h(x)^T \in \mathbb{R}^{l \times n}$ are Jacobian matrices, whose ith row is the transpose of the corresponding gradient.[6]

9.2 Necessary Optimality Conditions: Geometric view

We define a set $C \subseteq \mathbb{R}^n$ to be a *cone* if for every $x \in C$, $\alpha x \in C$ for any $\alpha > 0$. A set C is a *convex cone* if C is a cone and C is a convex set.

Suppose $\bar{x} \in S$. We make the following definitions:

$F_0 = \{d : \nabla f(\bar{x})^T d < 0\}$ — elements of this cone are improving directions of f at $\bar{x}$.

$I = \{i : g_i(\bar{x}) = 0\}$ — the indices of binding inequality constraints.

$G_0 = \{d : \nabla g_i(\bar{x})^T d < 0\ \forall i \in I\}$ — elements of this cone are inward directions of binding inequality constraints.

$H_0 = \{d : \nabla h_i(\bar{x})^T d = 0\ \forall i = 1, \ldots, l\}$ — the cone of tangent directions of equality constraints.

Note that, while all elements of F_0 are clearly improving directions, this cone does not necessarily contain *all* improving directions: indeed, we can have a direction d which is an improving direction, but $\nabla f(\bar{x})^T d = 0$. Same is true for G_0: all its elements are inward directions of the binding constraints, but it may not include all such directions.[7]

[6] BSS uses a different notation for the Jacobian — they denote the Jacobian of g by ∇g. I don't like their notation, because it is inconsistent when g returns values in $\mathbb{R}^1$.

[7] For a more detailed discussion of this point, see BSS section 4.2.

Theorem 23 (cf. BSS 4.3.1, linear equalities) *Assume that $h(x)$ is a linear function, i.e., $h(x) = Ax - b$ for $A \in \mathbb{R}^{l\times n}$. If $\bar{x}$ is a local minimizer of (P), then $F_0 \cap G_0 \cap H_0 = \emptyset$.*

Proof: Let

$$A = \begin{bmatrix} a_1^T \\ \vdots \\ a_l^T \end{bmatrix}, \text{ where } a_i \in \mathbb{R}^n,\ i = 1, \ldots, l.$$

Then $h_i(x) = a_i^T x - b_i$ and $\nabla h_i(x) = a_i$, i.e., $H_0 = \{d : a_i^T d = 0,\ i = 1, \ldots, l\} = \{d : Ad = 0\}$.

Suppose $d \in F_0 \cap G_0 \cap H_0$. Then for all $\lambda > 0$ sufficiently small $g_i(\bar{x} + \lambda d) < g_i(\bar{x}) = 0\ \forall i \in I$ (for $i \notin I$, since λ is small, $g_i(\bar{x} + \lambda d) < 0$ by continuity), and $h(\bar{x} + \lambda d) = (A\bar{x} - b) + \lambda Ad = 0$. Therefore, $\bar{x} + \lambda d \in S$ for all $\lambda > 0$ sufficiently small. On the other hand, for all sufficiently small $\lambda > 0$, $f(\bar{x} + \lambda d) < f(\bar{x})$. This contradicts the assumption that $\bar{x}$ is a local minimizer of (P). ∎

To extend the above theorem to include general equality functions, we need the following tool, known as the Implicit Function Theorem.

Example: Let $h(x) = Ax - b$, where $A \in \mathbb{R}^{l\times n}$ has full row rank (i.e., its rows are linearly independent). Notice that $\nabla h(x) = A^T$, and the Jacobian of $h(\cdot)$ is equal to A.

We can partition columns of A and elements of x as follows: $A = [B, N]$, $x = (y; z)$, so that $B \in \mathbb{R}^{l\times l}$ is non-singular, and $h(x) = By + Nz - b$, $\nabla_y h(x)^T = B$ and $\nabla_z h(x)^T = N$.

Let $s(z) = B^{-1}b - B^{-1}Nz$. Then for any z, $h(s(z), z) = Bs(z) + Nz - b = 0$, i.e., $x = (s(z), z)$ solves $h(x) = 0$. Moreover, $\nabla s(z) = -N^T(B^T)^{-1}$. This idea of "invertability" of a system of equations is generalized (although only locally) by the following theorem (we will preserve the notation used in the example):

Theorem 24 (IFT) *Let $h(x) : \mathbb{R}^n \to \mathbb{R}^l$ and $\bar{x} = (\bar{y}_1, \ldots, \bar{y}_l, \bar{z}_1, \ldots, \bar{z}_{n-l}) = (\bar{y}, \bar{z})$ satisfy:*

1. *$h(\bar{x}) = 0$*
2. *$h(x)$ is continuously differentiable in a neighborhood of $\bar{x}$*
3. *The $l \times l$ Jacobian matrix*

$$\nabla_y h(\bar{y}, \bar{z})^T = \begin{bmatrix} \frac{\partial h_1(\bar{x})}{\partial y_1} & \cdots & \frac{\partial h_1(\bar{x})}{\partial y_l} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_l(\bar{x})}{\partial y_1} & \cdots & \frac{\partial h_l(\bar{x})}{\partial y_l} \end{bmatrix}$$

is non-singular.

Then there exists $\epsilon > 0$ along with functions $s(z) = (s_1(z), \ldots, s_l(z))^T$ such that $s(\bar{z}) = \bar{y}$ and $\forall z \in N_\epsilon(\bar{z})$, $h(s(z), z) = 0$. Moreover, for all $z \in N_\epsilon(\bar{z})$ $s_k(z)$ are continuously differentiable and

$$\nabla s(z) = -\nabla_z h(s(z), z) \cdot (\nabla_y h(s(z), z))^{-1}.$$

Example Consider $h(x) = x_1^2 + x_2^2 - 1$. Then $\nabla h(x) = (2x_1, 2x_2)^T$. Let also $\bar{x} = \frac{1}{\sqrt{2}}e$. We will use the IFT to formalize the fact that, locally, the implicit function $h(x) = 0$ can be written as an explicit function $x_2 = \sqrt{1 - x_1^2}$.

Let $y = x_2$ (notice that $\frac{\partial h(\bar{x})}{\partial x_2} = \sqrt{2} \neq 0$, so this is a valid choice). Then $z = x_1$, and the desired function is $s(z) = \sqrt{1-z^2}$. It is easy to verify that $h(s(z), z) = 0$ in a small neighborhood of $\bar{z} = 1/\sqrt{2}$. In particular, the neighborhood has to be small enough to be contained in the interval $(-1, 1)$. Moreover,

$$\frac{\partial s(z)}{\partial z} = -\nabla_z h(s(z), z) \cdot \frac{1}{\nabla_y h(s(z), z)} = -2z \cdot \frac{1}{2\sqrt{1-z^2}} = -\frac{z}{\sqrt{1-z^2}}.$$

Notice that the explicit function we derived only works locally; in the neighborhood of the point $-\bar{x}$, for example, the explicit form of the function is different. Moreover, in a neighborhood of the point $(1, 0)$, x_2 cannot be written as a function of x_1 — indeed, the corresponding submatrix of the Jacobian is singluar. However, x_1 can be expressed as a function of x_2 around that point.

Theorem 25 (cf. BSS 4.3.1) *If $\bar{x}$ is a local minimizer of (P) and the gradient vectors $\nabla h_i(\bar{x})$, $i = 1, \ldots, l$ are linearly independent, then $F_0 \cap G_0 \cap H_0 = \emptyset$.*

Proof: Since, by assumption, $\nabla h(\bar{x})^T \in \mathbb{R}^{l \times n}$ has full row rank, from the IFT, elements of $\bar{x}$ can be re-arranged so that $\bar{x} = (\bar{y}; \bar{z})$ and there exists $s(z)$ such that $s(\bar{z}) = \bar{y}$ and $h(s(z), z) = 0$ for z in a small neighborhood of $\bar{z}$.

Suppose $d = F_0 \cap G_0 \cap H_0$. Let $d = (q; p)$, where the partition is done in the same way as above. Let $z(\lambda) = \bar{z} + \lambda p$, $y(\lambda) = s(z(\lambda)) = s(\bar{z} + \lambda p)$, and $x(\lambda) = (y(\lambda), z(\lambda))$. We will derive a contradiction by showing that for small $\lambda > 0$, $x(\lambda)$ is feasible and $f(x(\lambda)) < f(\bar{x})$.

To show feasibility, first note that for all λ sufficiently close to 0 (positive and negative), by IFT,

$$h(x(\lambda)) = h(s(z(\lambda)), z(\lambda)) = 0. \tag{13}$$

We can show that $\left.\frac{\partial x_k(\lambda)}{\partial \lambda}\right|_{\lambda=0} = d_k$ for $k = 1, \ldots, n$ (or $\left.\frac{\partial x(\lambda)}{\partial \lambda}\right|_{\lambda=0} = d$) — see Proposition 26.

Using Taylor's expansion around $\lambda = 0$, we have for $i \in I$,

$$\begin{aligned} g_i(x(\lambda)) &= g_i(\bar{x}) + \lambda \left.\frac{\partial g_i(x(\lambda))}{\partial \lambda}\right|_{\lambda=0} + |\lambda|\alpha_i(\lambda) \\ &= \lambda \sum_{k=1}^{n} \frac{\partial g_i(x(\lambda))}{\partial x_k} \cdot \left.\frac{\partial x_k(\lambda)}{\partial \lambda}\right|_{\lambda=0} + |\lambda|\alpha_i(\lambda) \\ &= \lambda \nabla g_i(\bar{x})^T d + |\lambda|\alpha_i(\lambda), \end{aligned}$$

where $\alpha_i(\lambda) \to 0$ as $\lambda \to 0$. Hence $g_i(x(\lambda)) < 0$ for all $i = 1, \ldots, m$ for $\lambda > 0$ sufficiently small, and therefore, $x(\lambda)$ is feasible for any $\lambda > 0$ sufficiently small.

On the other hand,

$$f(x(\lambda)) = f(\bar{x}) + \lambda \nabla f(\bar{x})^T d + |\lambda|\alpha(\lambda) < f(\bar{x})$$

for $\lambda > 0$ sufficiently small, which contradicts local optimality of $\bar{x}$. ∎

Proposition 26 *Let $x(\lambda)$ and d be as defined in Theorem 25. Then $\left.\frac{\partial x(\lambda)}{\partial \lambda}\right|_{\lambda=0} = d$.*

Proof: We will continue with the notation and definitions in the statement and proof of Theorem 25. Recall that $x(\lambda) = (y(\lambda), z(\lambda))$, where $z(\lambda) = \bar{z} + \lambda p$, so, $\left.\frac{\partial z(\lambda)}{\partial \lambda}\right|_{\lambda=0} = p$. It remains to show that $\left.\frac{\partial y(\lambda)}{\partial \lambda}\right|_{\lambda=0} = q$.

Let $A = \nabla h(\bar{x})^T \in \mathbb{R}^{l \times n}$. Then A has full row rank. To use the IFT, the elements of $\bar{x}$ (and d) were re-arranged so that, after the corresponding re-arrangement of columns of A, we have $A = [B; N]$, where B is non-singular. Then $0 = Ad = Bq + Np$, or $q = -B^{-1}Np$.

Since (13) holds for all λ (positive and negative) sufficiently close to 0, we have $\frac{\partial h(x(\lambda))}{\partial \lambda} = 0$, or

$$0 = \frac{\partial h(x(\lambda))}{\partial \lambda} = \nabla_y h(x(\lambda))^T \cdot \frac{\partial y(\lambda)}{\partial \lambda} + \nabla_z h(x(\lambda))^T \cdot \frac{\partial z(\lambda)}{\partial \lambda}$$

for all λ sufficiently close to 0. In particular, for $\lambda = 0$,

$$\begin{aligned} 0 = & \nabla_y h(x(\lambda))^T\big|_{\lambda=0} \cdot \frac{\partial y(\lambda)}{\partial \lambda}\bigg|_{\lambda=0} + \nabla_z h(x(\lambda))^T\big|_{\lambda=0} \cdot \frac{\partial z(\lambda)}{\partial \lambda}\bigg|_{\lambda=0} \\ = & B \cdot \frac{\partial y(\lambda)}{\partial \lambda}\bigg|_{\lambda=0} + Np. \end{aligned}$$

Therefore,

$$\frac{\partial y(\lambda)}{\partial \lambda}\bigg|_{\lambda=0} = -B^{-1}Np = q.$$

Incidentally, this proof technique is quite similar to how the Jacobian of $s(z)$ is derived in the IFT. ∎

Note that Theorem 25 is essentially saying that if a point $\bar{x}$ is (locally) optimal, there is no direction d which is an *improving* direction (i.e., such that $f(\bar{x}+\lambda d) < f(\bar{x})$ for small $\lambda > 0$), and at the same time a *feasible direction* (i.e., such that $g_i(\bar{x} + \lambda d) \leq g_i(\bar{x}) = 0$ for $i \in I$ and $h(\bar{x} + \lambda d) = 0$), which makes sense intuitively. However, that the condition in Theorem 25 is somewhat weaker than the above intuitive explanation due to the fact that F_0 and G_0 might not contain the complete set of improving/inward directions, as discussed above.

9.3 Separation of convex sets

We will shortly attempt to reduce the geometric necessary local optimality conditions ($F_0 \cap G_0 \cap H_0 = \emptyset$) to a statement in terms of the gradients of the objective and constraints functions. For this we need the mathematical tools developed in this section.

First, some definitions:

- If $p \neq 0$ is a vector in $\mathbb{R}^n$ and α is a scalar, $H = \{x \in \mathbb{R}^n : p^T x = \alpha\}$ is a *hyperplane*, and $H^+ = \{x \in \mathbb{R}^n : p^T x \geq \alpha\}$, $H^- = \{x \in \mathbb{R}^n : p^T x \leq \alpha\}$ are *half-spaces*.
- Let S and T be two non-empty sets in $\mathbb{R}^n$. A hyperplane $H = \{x : p^T x = \alpha\}$ is said to *separate* S and T if $p^T x \geq \alpha$ for all $x \in S$ and $p^T x \leq \alpha$ for all $x \in T$, i.e., if $S \subseteq H^+$ and $T \subseteq H^-$. If, in addition, $S \cup T \not\subset H$, then H is said to *properly* separate S and T.
- H is said to *strictly* separate S and T if $p^T x > \alpha$ for all $x \in S$ and $p^T x < \alpha$ for all $x \in T$.
- H is said to *strongly* separate S and T if for some $\epsilon > 0$, $p^T x > \alpha + \epsilon$ for all $x \in S$ and $p^T x < \alpha - \epsilon$ for all $x \in T$.

Theorem 27 (BSS 2.4.4) *Let S be a nonempty closed convex set in $\mathbb{R}^n$, and $y \notin S$. Then $\exists p \neq 0$ and α such that $H = \{x : p^T x = \alpha\}$ strongly separates S and $\{y\}$.*

To prove the theorem, we need the following result:

Theorem 28 (BSS 2.4.1) *Let S be a nonempty closed convex set in $\mathbb{R}^n$, and $y \notin S$. Then there exists a unique point $\bar{x} \in S$ with minimum distance from y. Furthermore, $\bar{x}$ is the minimizing point if and only if $(y - \bar{x})^T(x - \bar{x}) \leq 0$ for all $x \in S$.*

Proof: Let $\hat{x}$ be an arbitrary point in S, and let $\bar{S} = S \cap \{x : \|x - y\| \leq \|\hat{x} - y\|\}$. Then $\bar{S}$ is a compact set. Let $f(x) = \|x - y\|$. Then $f(x)$ attains its minimum over the set $\bar{S}$ at some point $\bar{x} \in \bar{S}$ (note: $\bar{x} \neq y$), and by construction $\bar{S}$

To show uniqueness, suppose that $x' \in S$ is such that $\|y - \bar{x}\| = \|y - x'\|$. By convexity of S, $\frac{1}{2}(\bar{x} + x') \in S$. But by the triangle inequality, we get

$$\left\| y - \frac{1}{2}(\bar{x} + x') \right\| \leq \frac{1}{2}\|y - \bar{x}\| + \frac{1}{2}\|y - x'\|.$$

If strict inequality holds, we have a contradiction. Therefore, equality holds, and we must have $y - \bar{x} = \lambda(y - x')$ for some λ. Since $\|y - \bar{x}\| = \|y - x'\|$, $|\lambda| = 1$. If $\lambda = -1$, then $y = \frac{1}{2}(\bar{x} + x') \in S$, contradicting the assumption. Hence, $\lambda = 1$, i.e., $x' = \bar{x}$.

Finally we need to establish that $\bar{x}$ is the minimizing point if and only if $(y - \bar{x})^T(x - \bar{x}) \leq 0$ for all $x \in S$. To establish sufficiency, note that for any $x \in S$,

$$\|x - y\|^2 = \|(x - \bar{x}) - (y - \bar{x})\|^2 = \|x - \bar{x}\|^2 + \|y - \bar{x}\|^2 - 2(x - \bar{x})^T(y - \bar{x}) \geq \|\bar{x} - y\|^2.$$

Conversely, assume that $\bar{x}$ is the minimizing point. For any $x \in S$, $\lambda x + (1 - \lambda)\bar{x} \in S$ for any $\lambda \in [0, 1]$. Also, $\|\lambda x + (1 - \lambda)\bar{x} - y\| \geq \|\bar{x} - y\|$. Thus,

$$\|\bar{x} - y\|^2 \leq \|\lambda x + (1 - \lambda)\bar{x} - y\|^2 = \lambda^2\|x - \bar{x}\|^2 + 2\lambda(x - \bar{x})^T(\bar{x} - y) + \|\bar{x} - y\|^2,$$

or $\lambda^2\|x - \bar{x}\|^2 \geq 2\lambda(y - \bar{x})^T(x - \bar{x})$. This implies that $(y - \bar{x})^T(x - \bar{x}) \leq 0$ for any $x \in S$, since otherwise the above expression can be invalidated by choosing $\lambda > 0$ small. ∎

Proof of Theorem 27: Let $\bar{x} \in S$ be the point minimizing the distance from the point y to the set S. Note that $\bar{x} \neq y$. Let $p = y - \bar{x} \neq 0$, $\alpha = \frac{1}{2}(y - \bar{x})^T(y + \bar{x})$, and $\epsilon = \frac{1}{2}\|y - \bar{x}\|^2 > 0$. Then for any $x \in S$, $(x - \bar{x})^T(y - \bar{x}) \leq 0$, and so

$$p^T x = (y - \bar{x})^T x \leq \bar{x}^T(y - \bar{x}) = \bar{x}^T(y - \bar{x}) + \frac{1}{2}\|y - \bar{x}\|^2 - \epsilon = \alpha - \epsilon.$$

That is, $p^T x \leq \alpha - \epsilon$ for all $x \in S$. On the other hand, $p^T y = (y - \bar{x})^T y = \alpha + \epsilon$, establishing the result. ∎

Corollary 29 *If S is a closed, convex set in $\mathbb{R}^n$, then S is the intersection of all half-spaces that contain it.*

Theorem 30 *Let $S \subset \mathbb{R}^n$ and let C be the intersection of all half-spaces containing S. Then C is the smallest closed convex set containing S.*

Theorem 31 (BSS 2.4.5, Farkas' Lemma) *Exactly one of the following two systems has a solution:*

(i) $Ax \leq 0,\ c^T x > 0$

(ii) $A^T y = c,\ y \geq 0.$

Proof: First note that both systems cannot have a solution, since then we would have $0 < c^T x = y^T Ax \leq 0$.

Suppose the system (ii) has no solution. Let $S = \{x : x = A^T y \text{ for some } y \geq 0\}$. Then $c \notin S$. S is easily seen to be a convex set. Also, it can be shown that S is a closed set (see, for example, Appendix B.3 of "Nonlinear Programming" by Bertsekas). Therefore, there exist p and α such that $c^T p > \alpha$ and $(Ap)^T y \leq \alpha$ for all $y \geq 0$.

If $(Ap)_i > 0$, one could set y_i sufficiently large so that $(Ap)^T y > \alpha$, a contradiction. Thus $Ap \leq 0$. Taking $y = 0$, we also have that $\alpha \geq 0$, and so $c^T p > 0$, and p is a solution of (i). ∎

Lemma 32 (Key Lemma) *Exactly one of the two following systems has a solution:*

(i) $\bar{A}x < 0,\ Bx \leq 0,\ Hx = 0$

(ii) $\bar{A}^T u + B^T v + H^T w = 0,\ u \geq 0,\ v \geq 0,\ e^T u = 1.$

Proof: It is easy to show that both (i) and (ii) cannot have a solution. Suppose (i) does not have a solution. Then the system

$$\begin{gathered} \bar{A}x + e\theta \leq 0, \quad \theta > 0 \\ Bx \leq 0 \\ Hx \leq 0 \\ -Hx \leq 0 \end{gathered}$$

has no solution. This system can be re-written in the form

$$\begin{bmatrix} \bar{A} & e \\ B & 0 \\ H & 0 \\ -H & 0 \end{bmatrix} \cdot \begin{pmatrix} x \\ \theta \end{pmatrix} \leq 0,\ (0, \ldots, 0, 1) \cdot \begin{pmatrix} x \\ \theta \end{pmatrix} > 0.$$

From Farkas' Lemma, there exists a vector $(u; v; w^1; w^2) \geq 0$ such that

$$\begin{bmatrix} \bar{A} & e \\ B & 0 \\ H & 0 \\ -H & 0 \end{bmatrix}^T \cdot \begin{pmatrix} u \\ v \\ w^1 \\ w^2 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix},$$

or $\bar{A}^T u + B^T v + H^T(w^1 - w^2) = 0,\ e^T u = 1$. Letting $w = w^1 - w^2$ completes the proof of the lemma. ∎

9.4 First order optimality conditions

9.4.1 "Algebraic" necessary conditions

Theorem 33 (BSS 4.3.2, Fritz John Necessary Conditions) *Let $\bar{x}$ be a feasible solution of (P). If $\bar{x}$ is a local minimizer, then there exists (u_0, u, v) such that*

$$u_0 \nabla f(\bar{x}) + \sum_{i=1}^{m} u_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} v_i \nabla h_i(\bar{x}) = 0,$$

$$u_0, u \geq 0,\ (u_0, u, v) \neq 0,$$

$$u_i g_i(\bar{x}) = 0,\ i = 1, \ldots, m.$$

(Note that the fist equation can be rewritten as $u_0 \nabla f(\bar{x}) + \nabla g(\bar{x}) u + \nabla h(\bar{x}) v = 0$*.)*

Proof: If the vectors $\nabla h_i(\bar{x})$ are linearly dependent, then there exists $v \neq 0$ such that $\nabla h(\bar{x}) v = 0$. Setting $(u_0, u) = 0$ establishes the result.

Suppose now that the vectors $\nabla h_i(\bar{x})$ are linearly independent. Then Theorem 4.3.1 applies, i.e., $F_0 \cap G_0 \cap H_0 = \emptyset$. Assume for simplicity that $I = \{1, \ldots, p\}$. Let

$$\bar{A} = \begin{bmatrix} \nabla f(\bar{x})^T \\ \nabla g_1(\bar{x})^T \\ \vdots \\ \nabla g_p(\bar{x})^T \end{bmatrix},\ H = \begin{bmatrix} \nabla h_1(\bar{x})^T \\ \vdots \\ \nabla h_l(\bar{x})^T \end{bmatrix}.$$

Then there is no d that satisfies $\bar{A} d < 0$, $Hd = 0$. From the Key Lemma there exists $(u_0, u_1, \ldots, u_p)$ and $(v_1, \ldots, v_l)$ such that

$$u_0 \nabla f(\bar{x}) + \sum_{i=1}^{p} u_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} v_i \nabla h_i(\bar{x}) = 0,$$

with $u_0 + u_1 + \cdots + u_p = 1$ and $(u_0, u_1, \ldots, u_p) \geq 0$. Define $u_{p+1}, \ldots, u_m = 0$. Then $(u_0, u) \geq 0$, $(u_0, u) \neq 0$, and for any i, either $g_i(\bar{x}) = 0$, or $u_i = 0$. Finally,

$$u_0 \nabla f(\bar{x}) + \nabla g(\bar{x}) u + \nabla h(\bar{x}) v = 0. \ \blacksquare$$

Theorem 34 (BSS 4.3.7, KKT Necessary Conditions) *Let* $\bar{x}$ *be a feasible solution of (P) and let* $I = \{i : g_i(\bar{x}) = 0\}$*. Further, suppose that* $\nabla g_i(\bar{x})$ *for* $i \in I$ *and* $\nabla h_i(\bar{x})$ *for* $i = 1, \ldots, l$ *are linearly independent. If* $\bar{x}$ *is a local minimizer, there exists* (u, v) *such that*

$$\nabla f(\bar{x}) + \nabla g(\bar{x}) u + \nabla h(\bar{x}) v = 0, \tag{14}$$

$$u \geq 0,\ u_i g_i(\bar{x}) = 0\ i = 1, \ldots, m. \tag{15}$$

Proof: $\bar{x}$ must satisfy the Fritz John conditions. If $u_0 > 0$, we can redefine $u \leftarrow u/u_0$ and $v \leftarrow v/u_0$. If $u_0 = 0$, it would imply that $\sum_{i \in I} u_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} v_i \nabla h_i(\bar{x}) = 0$, i.e., the above gradients are linearly dependent. This contradicts the assumptions of the theorem. $\blacksquare$

A point $\bar{x}$ that together with some multiplier vectors u and v satisfies conditions (14) and (15) is referred to as a *KKT point*.

9.4.2 Generalizations of convexity and first order necessary conditions

Just like for unconstrained optimization, convexity of a problem plays a significant role in our ability to identify local and global minimizers by first order conditions. In fact, we can consider somewhat more "relaxed" notion of convexity for this purpose.

Suppose X is a convex set in $\mathbb{R}^n$. A function $f : X \to \mathbb{R}$ is *quasiconvex* if $\forall x, y \in X$ and $\forall \lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda) y) \leq \max\{f(x), f(y)\}.$$

If $f : X \to \mathbb{R}$, then the *level sets* of f are the sets

$$S_\alpha = \{x \in X : f(x) \leq \alpha\}$$

for each $\alpha \in \mathbb{R}$.

Proposition 35 *If f is convex, then f is quasiconvex.*

Proof: If f is convex, for $\lambda \in [0, 1]$,

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \leq \max\{f(x), f(y)\}. \blacksquare$$

Theorem 36 *A function f is quasiconvex on X if and only if S_α is a convex set for every $\alpha \in \mathbb{R}$.*

Proof: Suppose that f is quasiconvex. For any given value of α, suppose that $x, y \in S_\alpha$.

Let $z = \lambda x + (1-\lambda)y$ for some $\lambda \in [0, 1]$. Then $f(z) \leq \max\{f(x), f(y)\} \leq \alpha$, so $z \in S_\alpha$, which shows that S_α is convex for every α.

Conversely, suppose S_α is convex for every α. Let x and y be given, and let $\alpha = \max\{f(x),\ f(y)\}$, and hence $x, y \in S_\alpha$. Then for any $\lambda \in [0, 1]$, $f(\lambda x + (1-\lambda)y) \leq \alpha = \max\{f(x),\ f(y)\}$. Thus f is quasi-convex. $\blacksquare$

Corollary 37 *If f is a convex function, its level sets are convex.*

Suppose X is a convex set in $\mathbb{R}^n$. A differentiable function $f : X \to \mathbb{R}$ is *pseudoconvex* if for every $x, y \in X$, $\nabla f(x)^T (y - x) \geq 0$ implies $f(y) \geq f(x)$. (An equivalent way to define a pseudoconvex function is: if $f(y) < f(x)$, then $\nabla f(x)^T (y - x) < 0$.)

Theorem 38
1. A differentiable convex function is pseudoconvex.
2. A pseudoconvex function is quasiconvex.

Proof: To prove the first claim, we use the gradient inequality: if f is convex and differentiable, then $f(y) \geq f(x) + \nabla f(x)^T (y - x)$. Hence, if $\nabla f(x)^T (y - x) \geq 0$, then $f(y) \geq f(x)$, so f is pseudoconvex.

To show the second part of the theorem, suppose f is pseudoconvex. Let x, y and $\lambda \in [0, 1]$ be given, and let $z = \lambda x + (1-\lambda)y$. If $\lambda = 0$ or $\lambda = 1$, then $f(z) \leq \max\{f(x), f(y)\}$ trivially; therefore, assume $0 < \lambda < 1$. Let $d = y - x$.

We can assume without loss of generality that $\nabla f(z)^T d \geq 0$ (otherwise, reverse the roles of x and y). Then

$$\nabla f(z)^T (y - z) = \nabla f(z)^T (\lambda (y - x)) = \lambda \nabla f(z)^T d \geq 0,$$

so $f(z) \leq f(y) \leq \max\{f(x), f(y)\}$. Thus f is quasiconvex. $\blacksquare$

By extending the intuition on the relationship between convex and concave functions, a function $f(x)$ is quasiconcave (pseudoconcave) if its negative $(-f(x))$ is quasiconvex (pseudoconvex).

Theorem 39 (BSS 4.3.8 - modified, KKT First-order Sufficient Conditions) *Let $\bar{x}$ be a feasible point of (P), and suppose it satisfies*

$$\nabla f(\bar{x}) + \nabla g(\bar{x}) u + \nabla h(\bar{x}) v = 0,$$

$$u \geq 0,\ u_i g_i(\bar{x}) = 0,\ i = 1, \ldots, m.$$

If X is a convex set, $f(x)$ is pseudoconvex, $g_i(x)$'s are quasiconvex, and $h_i(x)$'s are linear, then $\bar{x}$ is a global optimal solution of (P).

Proof: Because each g_i is quasiconvex, the level sets

$$C_i = \{x \in X : g_i(x) \leq 0\},\ i = 1, \ldots, m$$

are convex sets. Also, because each h_i is linear, the sets

$$D_i = \{x \in X : h_i(x) = 0\},\ i = 1, \ldots, m$$

are convex. Thus, since the intersection of convex sets is convex, the feasible region

$$S = \{x \in X : g(x) \leq 0,\ h(x) = 0\}$$

is a convex set.

Let $x \in S$ be any point different from $\bar{x}$. Then $\lambda x + (1-\lambda)\bar{x}$ is feasible for all $\lambda \in (0, 1)$. Thus

$$\forall i \in I,\ g_i(\lambda x + (1-\lambda)\bar{x}) = g_i(\bar{x} + \lambda(x - \bar{x})) \leq 0 = g_i(\bar{x})$$

for any $\lambda \in (0, 1)$, i.e., direction $x - \bar{x}$ is not an ascent direction[8] of g_i at $\bar{x}$, we must have $\nabla g_i(\bar{x})^T(x - \bar{x}) \leq 0$ for all $i \in I$.

Similarly, $h_i(\bar{x} + \lambda(x - \bar{x})) = 0$, and so $\nabla h_i(\bar{x})^T(x - \bar{x}) = 0$ for all $i = 1, \ldots, l$.

Thus, from the KKT conditions,

$$\nabla f(\bar{x})^T(x - \bar{x}) = -\left(\nabla g(\bar{x})u + \nabla h(\bar{x})v\right)^T(x - \bar{x}) \geq 0,$$

and by pseudoconvexity, $f(x) \geq f(\bar{x})$ for any feasible x. ∎

The program

$$\begin{array}{ll} \text{(P)} & \min f(x) \\ \text{s.t.} & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in X \end{array}$$

is a convex program if f, $g_i,\ i = 1, \ldots, m$ are convex functions, $h_i,\ i = 1 \ldots, l$ are linear functions, and X is an open convex set.

Corollary 40 *The first order KKT conditions are sufficient for optimality in a convex program.*

9.4.3 Constraint qualifications, or when are necessary conditions really necessary?

Recall that the statement of the KKT necessary conditions established above has the form "if $\bar{x}$ is a local minimizer of (P) and *some requirement for the constraints* then KKT conditions must hold at $\bar{x}$." This additional requirement for the constraints that enables us to proceed with the proof of the KKT conditions is called a *constraint qualification*.

[8] An ascent direction is defined analogously to a descent direction: a direction d is an ascent direction of function f at point $\bar{x}$ if $f(\bar{x} + \lambda d) > f(\bar{x})$ for all $\lambda > 0$ sufficiently small.

We have already established (Theorem 34) that the following is a constraint qualification:
Linear Independence Constraint Qualification The gradients $\nabla g_i(\bar{x}),\ i \in I,\ \nabla h_i(\bar{x}),\ i = 1,\ldots,l$ are linearly independent.

We will now establish two other useful constraint qualifications.

Theorem 41 (Slater's condition) *Suppose the problem (P) satisfies Slater condition, i.e., $g_i,\ i = 1,\ldots,m$ are pseudoconvex, $h_i, i = 1,\ldots,l$ are linear, and $\nabla h_i(x),\ i = 1,\ldots,l$ are linearly independent, and there exists a point $x^0 \in X$ which satisfies $h(x^0) = 0$ and $g(x^0) < 0$. Then the KKT conditions are necessary to characterize an optimal solution.*

Proof: Let $\bar{x}$ be a local minimizer. Fritz-John conditions are necessary for this problem, i.e., there must exist $(u_0, u, v) \neq 0$ such that $(u_0, u) \geq 0$ and

$$u_0 \nabla f(\bar{x}) + \nabla g(\bar{x})u + \nabla h(\bar{x})v = 0,\ u_i g_i(\bar{x}) = 0.$$

If $u_0 > 0$, dividing through by u_0 demonstrates KKT conditions. Now suppose $u_0 = 0$. Let $d = x^0 - \bar{x}$. Then for each $i \in I$, $0 = g_i(\bar{x}) > g_i(x^0)$, and by pseudo-convexity of g_i, $\nabla g_i(\bar{x})^T d < 0$. Also, since $h(x)$ are linear, $\nabla h(\bar{x})^T d = 0$. Thus,

$$0 = 0^T d = (\nabla g(\bar{x})u + \nabla h(\bar{x})v)^T d < 0,$$

unless $u_i = 0$ for all $i \in I$. Therefore, $v \neq 0$ and $\nabla h(\bar{x})v = 0$, violating the linear independence assumption. This is a contradiction, and so $u_0 > 0$. ∎

The Slater condition here is stated in a less general form than in the textbook. The major difference is that, as stated in the book, this constraint qualification (as most others) has a "local" flavor, i.e., all conditions depend on the particular point $\bar{x}$ we are considering to be the candidate for optimality. Therefore, we can't really verify that the conditions hold until we have a particular point in mind (and once we have a point, it might very well turn out that it does not satisfy any local constraint qualification, and so we don't know if it needs to satisfy the KKT conditions!). It simplifies our task tremendously if our problem satisfies a global version of some constraint qualification, such as the Slater condition as stated in Theorem 41. Then we know that every candidate must satisfy KKT conditions! Of course, a global constraint qualification is a stronger condition than an analogous local qualification, so fewer problem will satisfy them.

The next constraint qualification is also of a "global" nature:

Theorem 42 (Linear constraints) *If all constraints are linear, the KKT conditions are necessary to characterize an optimal solution.*

Proof: Our problem is $\min\{f(x) : Ax \leq b,\ Mx = g\}$. Suppose $\bar{x}$ is a local minimizer. Without loss of generality, we can partition the constraints $Ax \leq b$ into groups $A_I x \leq b_I$ and $A_{\bar{I}} x \leq b_{\bar{I}}$ such that $A_I \bar{x} = b_I$ and $A_{\bar{I}} \bar{x} < b_{\bar{I}}$. Then at $\bar{x}$, the set $\{d : Md = 0,\ A_I d \leq 0\}$ is precisely the set of feasible directions. Thus, in particular, for every d as above, $\nabla f(\bar{x})^T d \geq 0$, for otherwise d would be a feasible descent direction at $\bar{x}$, violating its local optimality. Therefore, the linear system

$$\begin{bmatrix} A_I \\ M \\ -M \end{bmatrix} d \leq 0,\ -\nabla f(\bar{x})^T d > 0$$

has no solution.

From Farkas' lemma, there exists $(u, v^1, v^2) \geq 0$ such that $A_I^T u + M^T v^1 - M^T v^2 = -\nabla f(\bar{x})$. Taking $v = v^1 - v^2$, we obtain the KKT conditions. ∎

9.5 Second order conditions

To describe the second order conditions for optimality, we will define the following function, known as the *Lagrangian function*, or simply the Lagrangian:

$$L(x,u,v) = f(x) + \sum_{i=1}^{m} u_i g_i(x) + \sum_{i=1}^{l} v_i h_i(x).$$

Using the Lagrangian, we can, for example, re-write the gradient conditions of the KKT necessary conditions as follows:

$$\nabla_x L(\bar{x},u,v) = 0, \tag{16}$$

since $\nabla_x L(x,u,v) = \nabla f(x) + \nabla g(x)u + \nabla h(x)v$.

Also, note that $\nabla^2_{xx} L(x,u,v) = \nabla^2 f(x) + \sum_{i=1}^m u_i \nabla^2 g_i(x) + \sum_{i=1}^l v_i \nabla^2 h_i(x)$. Here we use the standard notation: $\nabla^2 q(x)$ denotes the Hessian of the function $q(x)$, and $\nabla^2_{xx} L(x,u,v)$ denotes the submatrix of the Hessian of $L(x,u,v)$ corresponding to the partial derivatives with respect to the x variables only.

Theorem 43 (BSS 4.4.3, KKT second order necessary conditions) *Suppose $\bar{x}$ is a local minimizer of (P), and $\nabla g_i(\bar{x})$, $i \in I$ and $\nabla h_i(\bar{x})$, $i = 1,\ldots,l$ are linearly independent. Then $\bar{x}$ is a KKT point, and, in addition,*

$$d^T \nabla^2_{xx} L(\bar{x},u,v) d \geq 0$$

for all $d \neq 0$ such that $\nabla g_i(\bar{x})^T d \leq 0$, $i \in I$, $\nabla h_i(\bar{x})^T d = 0$, $i = 1\ldots,l$.

Theorem 44 (BSS 4.4.2, KKT second order sufficient conditions) *Suppose the point $\bar{x} \in S$ together with multipliers (u,v) satisfies the first order KKT conditions. Let $I^+ = \{i \in I : u_i > 0\}$ and $I^0 = \{i \in I : u_i = 0\}$. If in addition,*

$$d^T \nabla^2_{xx} L(\bar{x},u,v) d > 0$$

for all $d \neq 0$ such that $\nabla g_i(\bar{x})^T d \leq 0$, $i \in I^0$, $\nabla g_i(\bar{x})^T d = 0$, $i \in I^+$, $\nabla h_i(\bar{x})^T d = 0$, $i = 1\ldots,l$. Then $\bar{x}$ is a (strict) local minimizer.

10 Linearly constrained problems and quadratic programming

10.1 The gradient projection method for linear equality constrained problems

10.1.1 Optimization over linear equality constraints

Suppose we want to solve

$$\begin{array}{lll} \text{(P)} & \min & f(x) \\ & \text{s.t.} & Ax = b. \end{array}$$

Assume that the problem is feasible and hence, without loss of generality, that matrix A has full row rank. The KKT conditions are necessary for this problem (because it satisfies a number of constraint qualifications) and are as follows:

$$\begin{array}{lll} A\bar{x} & & = b \\ \nabla f(\bar{x}) & +A^T\bar{v} & = 0. \end{array}$$

We therefore wish to find such a KKT point $(\bar{x}, \bar{v})$.

Suppose we are at an iterate x where $Ax = b$, i.e., x is a feasible point. Just like in the steepest descent algorithm, we wish to find a direction d which is a direction of steepest descent of the objective function, but in order to stay in the feasible region, we also need to have $Ad = 0$. Therefore, the direction-finding problem takes form

$$\begin{array}{ll} \min & \nabla f(x)^T d \\ \text{s.t.} & d^T I d \leq 1 \\ & Ad = 0. \end{array}$$

The first constraint of the problem requires that the search direction d has length 1 in the Euclidean norm. We can, however, adapt a more general approach and replace the Euclidean norm $\|d\| = \sqrt{d^T I d} = \sqrt{d^T d}$ with a more general norm $\|d\|_Q = \sqrt{d^T Q d}$, where Q is an arbitrary symmetric positive definite matrix. Using this general norm in the direction-finding problem, we can state the **projected steepest descent algorithm:**

Step 0 Given x_0, set $k \leftarrow 0$

Step 1 Solve the direction-finding problem defined at point x_k:

$$\begin{array}{lll} (\text{DFP}_{x_k}) & d_k = \text{argmin} & \nabla f(x_k)^T d \\ & \text{s.t.} & d^T Q d \leq 1 \\ & & Ad = 0. \end{array}$$

If $\nabla f(x_k)^T d_k = 0$, stop. x_k is a KKT point.

Step 2 Choose stepsize λ_k by performing an exact (or inexact) line search.

Step 3 Set $x_{k+1} \leftarrow x_k + \lambda_k d_k$, $k \leftarrow k+1$. Go to **Step 1**.

Notice that if $Q = I$ and the equality constraints are absent, the above is just the steepest descent algorithm. The choice of name "projected" steepest descent may not be apparent at this point, but will be clarified later.

10.1.2 Analysis of (DFP)

Note that (DFP_{x_k}) above is a convex program, and $d = 0$ is a Slater point. Therefore, the KKT conditions are necessary and sufficient for optimality. These conditions are (we omit the superscript k for simplicity):

$$\begin{aligned} Ad &= 0 \\ d^T Qd &\leq 1 \\ \nabla f(x) + A^T\pi + 2\beta Qd &= 0 \\ \beta &\geq 0 \\ \beta(1 - d^T Qd) &= 0. \end{aligned} \tag{17}$$

Let d solve these equations together with multipliers β and π.

Proposition 45 *x is a KKT point of (P) if and only if $\nabla f(x)^T d = 0$, where d is a KKT point of (DFP_x).*

Proof: First, suppose x is a KKT point of (P). Then there exists v such that

$$Ax = b,\ \nabla f(x) + A^T v = 0.$$

Let d be any KKT point of (DFP_x) together with multipliers π and β. Then, in particular, $Ad = 0$. Therefore,

$$\nabla f(x)^T d = -(A^T v)^T d = v^T Ad = 0.$$

To prove the converse, suppose that d (together with multipliers π and β) is a KKT point of (DFP_x), and $\nabla f(x)^T d = 0$. Then

$$0 = (\nabla f(x) + A^T\pi + 2\beta Qd)^T d = \nabla f(x)^T d + \pi^T Ad + 2\beta d^T Qd = 2\beta,$$

and so $\beta = 0$. Therefore,

$$Ax = b \text{ and } \nabla f(x) + A^T\pi = 0,$$

i.e., the point x (together with multiplier vector π) is a KKT point of (P). ∎

Proposition 46 *x is a KKT point of (P) if and only if $\beta = 0$, where β is the appropriate KKT multiplier of (DFP_x).*

Proposition 47 *If x is not a KKT point of (P), then d is a descent direction, where d is the KKT point of (DFP_x).*

Proposition 48 *The projected steepest descent algorithm has the same convergence properties and the same linear convergence as the steepest descent algorithm. Under the same conditions as in the steepest descent algorithm, the iterates converge to a KKT point of (P), and the convergence rate is linear, with a convergence constant that is bounded in terms of eigenvalues identically as in the steepest descent algorithm.*

10.1.3 Solving (DFP_x)

Approach 1 to solving DFP: solving linear equations

Create the system of linear equations:

$$\begin{array}{lll} Q\tilde{d} & +A^T\tilde{\pi} & = -\nabla f(x) \\ A\tilde{d} & & = 0 \end{array} \tag{18}$$

and solve this linear system for $(\tilde{d}, \tilde{\pi})$ by any method at your disposal.

If $Q\tilde{d} = 0$, then $\nabla f(x) + A^T\tilde{\pi} = 0$ and so x is a KKT point of (P).

If $Q\tilde{d} \neq 0$, then rescale the solution as follows:

$$d = \frac{\tilde{d}}{\sqrt{\tilde{d}^T Q \tilde{d}}},$$

$$\pi = \tilde{\pi},$$

$$\beta = \frac{1}{2\sqrt{\tilde{d}^T Q \tilde{d}}}.$$

Proposition 49 (d, π, β) *defined above satisfy* (17).

Note that the rescaling step is not necessary in practice, since we use a line-search.

Approach 2 to solving DFP: Formulae

Let

$$P_Q = Q^{-1} - Q^{-1}A^T(AQ^{-1}A^T)^{-1}AQ^{-1}$$

$$\beta = \frac{\sqrt{(\nabla f(x))^T P_Q (\nabla f(x))}}{2}$$

$$\pi = -(AQ^{-1}A^T)^{-1}AQ^{-1}(\nabla f(x))$$

If $\beta > 0$, let

$$d = \frac{-P_Q \nabla f(x)}{\sqrt{\nabla f(x)^T P_Q \nabla f(x)}}.$$

If $\beta = 0$, let $\bar{d} = 0$.

Proposition 50 *P_Q is symmetric and positive semi-definite. Hence β is well-defined.*

Proposition 51 (d, π, β) *defined above satisfy* (17).

10.1.4 The Variable Metric Method

In the projected steepest descent algorithm, the direction d must be contained in the ellipsoid $E_Q = \{d \in \mathbb{R}^n : d^T Q d \leq 1\}$, where Q is fixed for all iterations. In a variable metric method, Q can vary at every iteration. The **variable metric algorithm** is:

Step 0 Given x_0, set $k \leftarrow 0$

Step 1 Choose a symmetric positive definite matrix Q. (Perhaps $Q = Q(\bar{x})$, i.e., the choice of Q may depend on the current point.) Solve the direction-finding problem defined at point x_k:

$$(\text{DFP}_{x_k}) \quad d_k = \begin{array}{ll} \text{argmin} & \nabla f(x_k)^T d \\ \text{s.t.} & d^T Q d \leq 1 \\ & Ad = 0. \end{array}$$

If $\nabla f(x_k)^T d_k = 0$, stop. x_k is a KKT point.

Step 2 Choose stepsize λ_k by performing an exact (or inexact) line search.

Step 3 Set $x_{k+1} \leftarrow x_k + \lambda_k d_k$, $k \leftarrow k+1$. Go to **Step 1**.

All properties of the projected steepest descent algorithm still hold here.

Some strategies for choosing Q at each iteration are:

- $Q = I$
- Q is a given matrix held constant over all iterations
- $Q = H(x_k)$ where $H(x)$ is the Hessian of $f(x)$. It is easy to show that in this case, the variable metric algorithm is equivalent to Newton's method with a line-search, see the proposition below.
- $Q = H(x_k) + \delta I$, where δ is chosen to be large for early iterations, but δ is chosen to be small for later iterations. One can think of this strategy as approximating the projected steepest descent algorithm in early iterations, followed by approximating Newton's method in later iterations.

Proposition 52 *Suppose that $Q = H(x_k)$ in the variable metric algorithm. Then the direction $\bar{d}$ in the variable metric method is a positive scalar times the Newton direction.*

Proof: If $Q = H(x_k)$, then the vector $\bar{d}$ of the variable metric method is the optimal solution of (DFP_{x_k}):

$$(\text{DFP}_{x_k}) \quad \bar{d} = \begin{array}{ll} \text{argmin} & \nabla f(x_k)^T d \\ \text{s.t.} & Ad = 0 \\ & d^T H(x_k) d \leq 1. \end{array}$$

The Newton direction $\tilde{d}$ for (P) at the point x_k is the solution of the following problem:

$$(\text{NDP}_{x_k}): \quad \hat{d} = \begin{array}{ll} \text{argmin} & \nabla f(x_k)^T d + \frac{1}{2} d^T H(x_k) d \\ \text{s.t.} & Ad = 0. \end{array} \tag{19}$$

If you write down the KKT conditions for each of these two problems, you then can easily verify that $\bar{d} = \gamma \hat{d}$ for some scalar $\gamma > 0$. ∎

10.2 Linear inequality constraints: manifold suboptimization methods

Suppose we want to solve

$$(\text{P}) \quad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax \leq b. \end{array}$$

The problem might also have linear equality constraints, but we will assume we can handle them in the spirit of the previous subsection.

The algorithm described here[9] can be viewed as a variant of gradient projection method above; the difference is that here the search direction needs to be in the null space of active constraints rather than the entire constraint set. Once the set of constraints that are active at a solution is identified, the method will be identical to the algorithm for equality-constrained problems. At the early phase of the algorithm we maintain the set of active constraints which is our current "guess" of the "correct" set of active constraints, and the algorithm proceeds essentially by searching through a sequence of manifolds, each defined by a set of constraints (the successive manifolds typically differ by one constraint).[10]

We will denote by $A(x)$ the set of indices of active constraints at a feasible point x. We will assume for simplicity that the set of vectors $\{a_i : i \in A(x)\}$ is linearly independent for every feasible point x.

A typical iteration of the method proceeds as follows: let x_k be an iterate. Solve the direction finding problem at x_k:

$$\begin{array}{rcll} d_k = & \text{argmin} & \nabla f(x_k)^T d + \frac{1}{2} d^T H_k d \\ & \text{s.t.} & a_i^T d = 0, \ i \in A(x_k) \end{array} \tag{20}$$

(here H_k is some symmetric positive definite matrix). If a non-zero descent direction is found, i.e., $\nabla f(x_k)^T d_k < 0$, the next iterate is given by $x_{k+1} = x_k + \lambda_k d_k$, where λ_k is chosen by some rule from the range $\{\lambda : A(x_k + \lambda d_k) \leq b\}$.

If no feasible descent direction can be found by solving the problem (20), there are two possibilities: either the current point x_k is optimal over the entire set of constraints, and the algorithm is terminated, or the current manifold needs to be updated.

We now analyze the problem (20) in further detail. First, note that since $d = 0$ is feasible, we must have

$$\nabla f(x_k)^T d_k + \frac{1}{2} (d_k)^T H_k d_k \leq 0.$$

If $d_k \neq 0$, then the above equation, together with positive-definiteness of H_k implies that d_k is a feasible descent direction.

What happens if $d_k = 0$? The optimal solution of the direction-finding problem can be computed from the KKT conditions:

$$d_k = -(H_k)^{-1}(\nabla f(x_k) + (A_k)^T v),$$

where v is the KKT multiplier

$$v = -(A_k (H_k)^{-1} A_k)^{-1} A_k (H_k)^{-1} \nabla f(x_k)$$

(here A_k is the submatrix of A consisting of rows of active constraints). If $d_k = 0$, then we have

$$\nabla f(x_k) + \sum_{i \in A(x_k)} v_i a_i = 0$$

If all $v_i \geq 0 : \ i \in A(x_k)$, then the current point is a KKT point for the problem (P), and the algorithm can be terminated. Suppose, on the other hand, that $v_j < 0$ for some $j \in A(x_k)$. We

[9] Based on Bertsekas, Section 2.5

[10] A *manifold* is formally defined as a topological space that is locally Euclidean. Here we are dealing with spaces that are intersections of several hyperplanes, which are, in a sense, the simplest kinds of manifolds.

proceed by deleting the constraint $a_j^T x = b_j$ from the manifold of active constraints and solving the new direction-finding problem

$$\begin{array}{rcll} \bar{d}_k = & \text{argmin} & \nabla f(x_k)^T d + \frac{1}{2} d^T H_k d \\ & \text{s.t.} & a_i^T d = 0 \; i \in A(x_k), \; i \neq j. \end{array}$$

Proposition 53 *$\bar{d}_k$ is a feasible descent direction.*

Proof: First we show that $\bar{d}_k \neq 0$. If $\bar{d}_k = 0$, then

$$\nabla f(x_k) + \sum_{i \in A(x_k), \; i \neq j} \bar{v}_i a_i = 0$$

for some vector of multipliers $\bar{v}$. Then

$$\sum_{i \in A(x_k), \; i \neq j} (v_i - \bar{v}_i) a_i + v_j a_j = 0,$$

which contradicts the linear independence assumption, since $v_j \neq 0$. Therefore $\bar{d}_k \neq 0$, and hence is a descent direction.

To show that $\bar{d}_k$ is a feasible direction, we need to show that $a_j^T \bar{d}_k \leq 0$. We have:

$$0 = (\nabla f(x_k) + \sum_{i \in A(x_k)} v_i a_i)^T \bar{d}_k = \nabla f(x_k)^T \bar{d}_k + \sum_{i \in A(x_k)} v_i a_i^T \bar{d}_k = \nabla f(x_k)^T \bar{d}_k + v_j a_j^T \bar{d}_k < v_j a_j^T \bar{d}_k.$$

Since $v_j < 0$, $a_j^T \bar{d}_k < 0$. This implies in particular that once a step in direction d_k is taken, the constraint $a_j^T x \leq b_j$ will no longer be active. ∎

The above discussion provides a general idea of the manifold suboptimization methods. Note that the initial stage of the algorithm will require at least as many manifold changes as the number of constraints whose status differs at the initial point and the solution. Therefore these methods are most efficient when the number of inequality constraints is relatively small.

10.3 Quadratic Programming

Quadratic programming (QP) problems involve minimization of a quadratic function $\frac{1}{2} x^T Q x + q^T x$ subject to linear equality and inequality constraints. The two algorithms above can be easily re-stated for the case of quadratic programming.

In general, if the matrix Q is positive semidefinite, then the corresponding QP is convex and its optimal solutions are completely characterized by the KKT conditions. Therefore, for such problems the algorithms described above converge to an optimal solution. The computation involved in solving convex QP problems is often simplified. For example, if Q is positive definite, in the projected Newton's method outlined above, matrix P_Q needs to be computed only once, since the Hessian of the function $f(x)$ remains constant. Also, in the manifold suboptimization method, it makes sense to use $H_k = Q$ (provided that Q is positive definite), also simplifying the computation.

If, however, the matrix Q is not positive semidefinite, QP can potentially have many local minimizers, and the algorithms applied to such problems will typically find only a local minimizer.

11 Introduction to penalty methods for constrained optimization

Consider the constrained optimization problem (P):

$$\begin{array}{lll} \text{(P)} & \min & f(x) \\ & \text{s.t.} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & & h_i(x) = 0, \quad i = 1, \ldots, l \\ & & x \in X \subseteq \mathbb{R}^n, \end{array}$$

whose feasible region we denote by $S = \{x \in X : g_i(x) \leq 0, i = 1, \ldots, m, h_i(x) = 0, i = 1, \ldots, l\}$. Here, the set X represents either an open set, or a set defined by constraints that can be easily incorporated into the optimization (e.g., linear equality constraints).

Penalty methods are designed to solve (P) by instead solving a specially constructed unconstrained optimization problem, or a sequence of such problems. In particular the feasible region of (P) is expanded from S to all of X, but a large cost or "penalty" is added to the objective function for points that lie outside of the original feasible region S.

We will often write $g(x) = (g_1(x), \ldots, g_m(x))^T$ and $h(x) = (h_1(x), \ldots, h_l(x))^T$ for convenience.

Definition: A function $p(x) : \mathbb{R}^n \to \mathbb{R}$ is called a *penalty function* for (P) if $p(x)$ satisfies:

- $p(x) = 0$ if $g(x) \leq 0,\ h(x) = 0$ and
- $p(x) > 0$ if $g(x) \not\leq 0$ or $h(x) \neq 0$.

Penalty functions are typically defined by

$$p(x) = \sum_{i=1}^{m} \phi(g_i(x)) + \sum_{i=1}^{l} \psi(h_i(x)), \tag{21}$$

where

- $\phi(y) = 0$ if $y \leq 0$ and $\phi(y) > 0$ if $y > 0$
- $\psi(y) = 0$ if $y = 0$ and $\psi(y) > 0$ if $y \neq 0$,

although more general functions satisfying the definition can conceptually be used.

Example:

$$p(x) = \sum_{i=1}^{m} (\max\{0, g_i(x)\})^2 + \sum_{i=1}^{l} h_i(x)^2.$$

We then consider solving the following *penalty program*:

$$\begin{array}{lll} \text{P}(c): & \min & f(x) + cp(x) \\ & \text{s.t.} & x \in X \end{array}$$

for an increasing sequence of constants c as $c \to +\infty$. Note that in the program P(c) we are assigning a penalty to the violated constraints. The scalar quantity c is called the *penalty parameter*.

Let $c_k \geq 0,\ k = 1, \ldots, \infty$, be a sequence of penalty parameters that satisfies $c_{k+1} > c_k$ for all k and $\lim_{k\to\infty} c_k = +\infty$. Let $q(c, x) = f(x) + cp(x)$, and let x_k be the exact solution to the program P(c_k), and let $x^\star$ denote any optimal solution of (P).

The following Lemma presents some basic properties of penalty methods:

Lemma 54 (Penalty Lemma, 9.2.1)

1. $q(c_k, x_k) \leq q(c_{k+1}, x_{k+1})$
2. $p(x_k) \geq p(x_{k+1})$
3. $f(x_k) \leq f(x_{k+1})$
4. $f(x^\star) \geq q(c_k, x_k) \geq f(x_k)$

Proof:

1. We have

$$q(c_{k+1}, x_{k+1}) = f(x_{k+1}) + c_{k+1}p(x_{k+1}) \geq f(x_{k+1}) + c_k p(x_{k+1}) \geq f(x_k) + c_k p(x_k) = q(c_k, x_k)$$

2. $$f(x_k) + c_k p(x_k) \leq f(x_{k+1}) + c_k p(x_{k+1})$$

   and

$$f(x_{k+1}) + c_{k+1}p(x_{k+1}) \leq f(x_k) + c_{k+1}p(x_k)$$

   Thus $(c_{k+1} - c_k)p(x_k) \geq (c_{k+1} - c_k)p(x_{k+1})$, whereby $p(x_k) \geq p(x_{k+1})$.

3. From the proof of (1.),

$$f(x_{k+1}) + c_k p(x_{k+1}) \geq f(x_k) + c_k p(x_k).$$

   But $p(x_k) \geq p(x_{k+1})$, which implies that $f(x_{k+1}) \geq f(x_k)$.

4. $f(x_k) \leq f(x_k) + c_k p(x_k) \leq f(x^\star) + c_k p(x^\star) = f(x^\star)$. ∎

The next result concerns convergence of the penalty method.

Theorem 55 (Penalty Convergence Theorem, 9.2.2) *Suppose that $S \neq \emptyset$ and $f(x)$, $g(x)$, $h(x)$, and $p(x)$ are continuous functions. Let $\{x_k\}$, $k = 1, \ldots, \infty$, be a sequence of solutions to $P(c_k)$, and suppose the sequence $\{x_k\}$ is contained in a compact set. Then any limit point $\bar{x}$ of $\{x_k\}$ solves (P).*

Proof: Let $\bar{x}$ be a limit point of $\{x_k\}$. From the continuity of the functions involved, $\lim_{k\to\infty} f(x_k) = f(\bar{x})$. Also, from the Penalty Lemma,

$$q^\star = \lim_{k\to\infty} q(c_k, x_k) \leq f(x^\star).$$

Thus

$$\lim_{k\to\infty} c_k p(x_k) = \lim_{k\to\infty} [q(c_k, x_k) - f(x_k)] = q^\star - f(\bar{x}).$$

But $c_k \to \infty$, which implies from the above that

$$\lim_{k\to\infty} p(x_k) = 0.$$

Therefore, from the continuity of $p(x)$, $g(x)$ and $h(x)$, $p(\bar{x}) = 0$, and so $g(\bar{x}) \leq 0$ and $h(\bar{x}) = 0$, that is, $\bar{x}$ is a feasible solution of (P). Finally, from the Penalty Lemma, $f(x_k) \leq f(x^\star)$ for all k, and so $f(\bar{x}) \leq f(x^\star)$, which implies that $\bar{x}$ is an optimal solution of (P). ∎

An often-used class of penalty functions is:

$$p(x) = \sum_{i=1}^{m} [\max\{0, g_i(x)\}]^q + \sum_{i=1}^{l} |h_i(x)|^q, \quad \text{where} \quad q \geq 1. \tag{22}$$

We note the following:

- If $q = 1$, $p(x)$ in (22) is called the "linear penalty function." This function may not be differentiable at points where $g_i(x) = 0$ or $h_i(x) = 0$ for some i.
- Setting $q = 2$ is the most common form of (22) that is used in practice, and is called the "quadratic penalty function" (for obvious reasons). If we denote

  $$g^+(x) = (\max\{0, g_i(x)\}, \ldots, \max\{0, g_m(x)\})^T,$$

  then the quadratic penalty function can be written as

  $$p(x) = (g^+(x))^T (g^+(x)) + h(x)^T h(x).$$

11.1 Karush-Kuhn-Tucker multipliers in penalty methods

Suppose the penalty function $p(x)$ is defined as

$$p(x) = \sum_{i=1}^{m} \phi(g_i(x)) + \sum_{i=1}^{l} \psi(h_i(x)),$$

where $\phi(y)$ and $\psi(y)$ are as above.

Note that $p(x)$ might not be continuously differentiable, since the functions $g_i^+(x)$ are not differentiable at points x where $g_i(x) = 0$. However, if we assume that the functions $\phi(y)$ and $\psi(y)$ are continuously differentiable and

$$\phi'(0) = 0, \tag{23}$$

then $p(x)$ is differentiable whenever the functions $g(x)$, and $h(x)$ are differentiable, and we can write

$$\nabla p(x) = \sum_{i=1}^{m} \phi'(g_i(x)) \nabla g_i(x) + \sum_{i=1}^{l} \psi'(h_i(x)) \nabla h_i(x). \tag{24}$$

Now let x_k solve $P(c_k)$. Then x_k will satisfy

$$\nabla f(x_k) + c_k \nabla p(x_k) = 0,$$

that is,

$$\nabla f(x_k) + c_k \sum_{i=1}^{m} \phi'(g_i(x_k)) \nabla g_i(x_k) + c_k \sum_{i=1}^{l} \psi'(h_i(x_k)) \nabla h_i(x_k) = 0.$$

Let us define

$$[u_k]_i = c_k \phi'(g_i(x_k)), \ [v_k]_i = c_k \psi'(h_i(x_k)). \tag{25}$$

Then

$$\nabla f(x_k) + \sum_{i=1}^{m} [u_k]_i \nabla g_i(x_k) + \sum_{i=1}^{l} [v_k]_i \nabla h_i(x_k) = 0,$$

and so we can interpret (u_k, v_k) as a sort of vector of Karush-Kuhn-Tucker multipliers. In fact, we have:

Lemma 56 *Suppose $\phi(y)$ and $\psi(y)$ are continuously differentiable and satisfy (23), and that $f(x)$, $g(x)$, and $h(x)$ are differentiable. Let (u_k, v_k) be defined by (25). Then if $x_k \to \bar{x}$, and $\bar{x}$ satisfies the linear independence condition for gradient vectors of active constraints, then $(u_k, v_k) \to (\bar{u}, \bar{v})$, where $(\bar{u}, \bar{v})$ is a vector of Karush-Kuhn-Tucker multipliers for the optimal solution $\bar{x}$ of (P).*

Proof: From the Penalty Convergence Theorem, $\bar{x}$ is an optimal solution of (P). Let $I = \{i \mid g_i(\bar{x}) = 0\}$ and $N = \{i : g_i(\bar{x}) < 0\}$. For $i \in N$, $g_i(x_k) < 0$ for all k sufficiently large, so $[u_k]_i = 0$ for all k sufficiently large, whereby $\bar{u}_i = 0$ for $i \in N$.

From (25) and the definition of a penalty function, it follows that $[u_k]_i \geq 0$ for $i \in I$, for all k sufficiently large.

Suppose $(u_k, v_k) \to (\bar{u}, \bar{v})$ as $k \to \infty$. Then $\bar{u}_i = 0$ for $i \in N$. From the continuity of all functions involved,

$$\nabla f(x_k) + \sum_{i=1}^{m} [u_k]_i \nabla g_i(x_k) + \sum_{i=1}^{l} [v_k]_i \nabla h_i(x_k) = 0$$

implies

$$\nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{u}_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} \bar{v}_i \nabla h_i(\bar{x}) = 0.$$

From the above remarks, we also have $\bar{u} \geq 0$ and $\bar{u}_i = 0$ for all $i \in N$. Thus $(\bar{u}, \bar{v})$ is a vector of Karush-Kuhn-Tucker multipliers. It therefore remains to show that $(u_k, v_k) \to (\bar{u}, \bar{v})$ for some unique $(\bar{u}, \bar{v})$.

Suppose $\{(u_k, v_k)\}_{k=1}^{\infty}$ has no accumulation point. Then $\|(u_k, v_k)\| \to \infty$. But then define

$$(\lambda_k, \mu_k) = \frac{(u_k, v_k)}{\|(u_k, v_k)\|},$$

and then $\|(\lambda_k, \mu_k)\| = 1$ for all k, and so the sequence $\{(\lambda_k, \mu_k)\}_{k=1}^{\infty}$ has some accumulation point $(\bar{\lambda}, \bar{\mu})$. For all $i \in N$, $[\lambda_k]_i = 0$ for k large, whereby $\bar{\lambda}_i = 0$ for $i \in N$, and

$$\begin{aligned}
\sum_{i \in I} [\lambda_k]_i \nabla g_i(x_k) + \sum_{i=1}^{l} [\mu_k]_i \nabla h_i(x_k) &= \sum_{i=1}^{m} [\lambda_k]_i \nabla g_i(x_k) + \sum_{i=1}^{l} [\mu_k]_i \nabla h_i(x_k) \\
&= \sum_{i=1}^{m} \left(\frac{[u_k]_i}{\|(u_k, v_k)\|} \right) \nabla g_i(x_k) + \sum_{i=1}^{l} \left(\frac{[v_k]_i}{\|(u_k, v_k)\|} \right) \nabla h_i(x_k) \\
&= \frac{-\nabla f(x_k)}{\|(u_k, v_k)\|}
\end{aligned}$$

for k large. As $k \to \infty$, we have $x_k \to \bar{x}$, $(\lambda_k, \mu_k) \to (\bar{\lambda}, \bar{\mu})$, and $\|(u_k, v_k)\| \to \infty$, and so the above becomes

$$\sum_{i \in I} \bar{\lambda}_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} \bar{\mu}_i \nabla h_i(\bar{x}) = 0,$$

and $\|(\bar{\lambda}, \bar{\mu})\| = 1$, which violates the linear independence condition. Therefore $\{(u_k, v_k)\}$ is a bounded sequence, and so has at least one accumulation point.

Now suppose that $\{(u_k, v_k)\}$ has two accumulation points, $(\tilde{u}, \tilde{v})$ and $(\bar{u}, \bar{v})$. Note $\bar{u}_i = 0$ and $\tilde{u}_i = 0$ for $i \in N$, and so

$$\sum_{i \in I} \bar{u}_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} \bar{v}_i \nabla h_i(\bar{x}) = -\nabla f(\bar{x}) = \sum_{i \in I} \tilde{u}_i \nabla g_i(\bar{x}) + \sum_{i=1}^{l} \tilde{v}_i \nabla h_i(\bar{x}),$$

so that

$$\sum_{i \in I} (\bar{u}_i - \tilde{u}_i) \nabla g_i(\bar{x}) + \sum_{i=1}^{l} (\bar{v}_i - \tilde{v}_i) \nabla h_i(\bar{x}) = 0.$$

But by the linear independence condition, $\bar{u}_i - \tilde{u}_i = 0$ for all $i \in I$, and $\bar{v} - \tilde{v} = 0$. This then implies that $(\bar{u}, \bar{v}) = (\tilde{u}, \tilde{v})$. ∎

Remark. The quadratic penalty function satisfies the condition (23), but that the linear penalty function does not satisfy (23).

11.2 Exact penalty methods

The idea in an *exact penalty method* is to choose a penalty function $p(x)$ and a constant c so that the optimal solution $\tilde{x}$ of P(c) is also an optimal solution of the original problem (P).

Theorem 57 (9.3.1) *Suppose (P) is a convex program for which the Karush-Kuhn-Tucker conditions are necessary.*

Suppose that

$$p(x) := \sum_{i=1}^{m} g_i(x)^+ + \sum_{i=1}^{l} |h_i(x)|.$$

Then as long as c is chosen sufficiently large, the sets of optimal solutions of $P(c)$ and (P) coincide. In fact, it suffices to choose $c > \max\{u_i^\star,\ i = 1, \ldots, m;\ |v_i^\star|,\ i = 1, \ldots, l\}$, where $(u^\star, v^\star)$ is a vector of Karush-Kuhn-Tucker multipliers.

Proof: Suppose $\hat{x}$ solves (P). For any $x \in \mathbb{R}^n$ we have:

$$\begin{aligned}
q(c,x) &= f(x) + c\left(\sum_{i=1}^{m} g_i(x)^+ + \sum_{i=1}^{l} |h_i(x)|\right) \\
&\geq f(x) + \sum_{i=1}^{m} u_i^\star g_i(x)^+ + \sum_{i=1}^{l} |v_i h_i(x)| \\
&\geq f(x) + \sum_{i=1}^{m} u_i^\star g_i(x) + \sum_{i=1}^{l} v_i h_i(x) \\
&\geq f(x) + \sum_{i=1}^{m} u_i^\star (g_i(\hat{x}) + \nabla g_i(\hat{x})^T(x-\hat{x})) + \sum_{i=1}^{l} v_i^\star (h_i(\hat{x}) + \nabla h_i(\hat{x})^T(x-\hat{x})) \\
&= f(x) + \left(\sum_{i=1}^{m} u_i^\star \nabla g_i(\hat{x}) + \sum_{i=1}^{l} v_i^\star \nabla h_i(\hat{x})\right)^T (x-\hat{x}) \\
&= f(x) - \nabla f(\hat{x})^T(x-\hat{x}) \geq f(\hat{x}) \\
&= f(\hat{x}) + c\left(\sum_{i=1}^{m} g_i(\hat{x})^+ + \sum_{i=1}^{l} |h_i(\hat{x})|\right) = q(c,\hat{x}).
\end{aligned}$$

Thus $q(c,\hat{x}) \leq q(c,x)$ for all x, and therefore $\hat{x}$ solves $P(c)$.

Next suppose that $\bar{x}$ solves $P(c)$. Then if $\hat{x}$ solves (P), we have:

$$f(\bar{x}) + c\left(\sum_{i=1}^{m} g_i(\bar{x})^+ + \sum_{i=1}^{l} |h_i(\bar{x})|\right) \leq f(\hat{x}) + c\left(\sum_{i=1}^{m} g_i(\hat{x})^+ + \sum_{i=1}^{l} |h_i(\hat{x})|\right) = f(\hat{x}),$$

and so

$$f(\bar{x}) \leq f(\hat{x}) - c\left(\sum_{i=1}^{m} g_i(\bar{x})^+ + \sum_{i=1}^{l} |h_i(\bar{x})|\right). \tag{26}$$

However, if $\bar{x}$ is not feasible for (P), then

$$\begin{aligned}
f(\bar{x}) &\geq f(\hat{x}) + \nabla f(\hat{x})^T(\bar{x}-\hat{x}) \\
&= f(\hat{x}) - \sum_{i=1}^{m} u_i^\star \nabla g_i(\hat{x})^T(\bar{x}-\hat{x}) - \sum_{i=1}^{l} v_i^\star \nabla h_i(\hat{x})^T(\bar{x}-\hat{x}) \\
&\geq f(\hat{x}) + \sum_{i=1}^{m} u_i^\star (g_i(\hat{x}) - g_i(\bar{x})) + \sum_{i=1}^{l} v_i^\star (h_i(\hat{x}) - h_i(\bar{x})) \\
&= f(\hat{x}) - \sum_{i=1}^{m} u_i^\star g_i(\bar{x}) - \sum_{i=1}^{l} v_i^\star h_i(\bar{x}) \\
&> f(\hat{x}) - c\left(\sum_{i=1}^{m} g_i(\bar{x})^+ + \sum_{i=1}^{l} |h_i(\bar{x})|\right),
\end{aligned}$$

which contradicts (26). Thus $\bar{x}$ is feasible for (P). That being the case,

$$f(\bar{x}) \leq f(\hat{x}) - c\left(\sum_{i=1}^{m} g_i(\bar{x})^+ + \sum_{i=1}^{l} |h_i(x)|\right) = f(\hat{x})$$

from (26), and so $\bar{x}$ solves (P). ∎

11.3 Augmented Lagrangian penalty function

As we have seen in the above discussion, most "smooth" penalty functions (such as quadratic penalty function) never generate exact solutions to the constrained minimization problem. Therefore, we would need to solve the (penalized) unconstrained problems with very large values of the constant c in order to obtain solutions that are close to being feasible and optimal. (In theory, we need to let $c \to \infty$ to obtain a solution.) This is unfortunate, since the unconstrained optimization problems one encounters in implementing penalty methods tend to become ill-conditioned when c increases, and therefore, it will be hard to solve each of the unconstrained subproblems required by the algorithm.

Alternatively, one could employ an exact penalty method, i.e., a method that guarantees termination at an optimal solution provided that the value of c is sufficiently large (but finite). As we have established, linear penalty function is an exact penalty function; unfortunately, it is not differentiable at points at the boundary of the feasible region, and therefore poses difficulties in solving corresponding unconstrained problems. Below we attempt to investigate the existence of differentiable exact penalty methods.

For simplicity we will consider problems (P) with only equality constraints (there are techniques that extend the discussion below to more general problems). Consider the following penalty function, known as the *augmented Lagrangian penalty function* or the *multiplier penalty function*:

$$L_{\text{ALAG}}(x, v) = f(x) + \sum_{i=1}^{l} v_i h_i(x) + c \sum_{i=1}^{l} h_i(x)^2,$$

where $v \in \mathbb{R}^l$ is some vector of multipliers, that can be either kept constant or updated as we proceed with the penalty algorithm. (Compare this to the usual Lagrangian function $L(x, v) = f(x) + \sum_{i=1}^{l} v_i h_i(x)$.) The usage of this function as a penalty function can be partially motivated by the following observation: suppose that $\bar{x}$ is the optimal solution of (P), and $\bar{v}$ is the vector of corresponding multipliers. Taking the (partial) gradient of the function L_{ALAG} at $(\bar{x}, \bar{v})$, we obtain

$$\nabla_x L_{\text{ALAG}}(\bar{x}, \bar{v}) = \left[\nabla f(\bar{x}) + \sum_{i=1}^{l} \bar{v}_i \nabla h_i(\bar{x})\right] + 2c \sum_{i=1}^{l} h_i(\bar{x}) \nabla h_i(\bar{x}) = 0$$

for all values of c (this is not necessarily true for the simple quadratic penalty function!). Hence, if the vector of multipliers $\bar{v}$ is known, one can hope that under some regularity assumptions, the point $\bar{x}$ is the local minimizer of $L_{\text{ALAG}}(x, \bar{v})$ for large (but finite) values of c. Indeed, we have the following

Theorem 58 (9.3.3) *Let $(\bar{x}, \bar{v})$ be a KKT solution of (P) satisfying the second order sufficiency conditions for a local minimum, i.e.,*

$$d^T \nabla^2_{xx} L(\bar{x}, v) d > 0$$

for all $d \neq 0$ such that $\nabla h_i(\bar{x})^T d = 0,\ i = 1 \ldots, l$. Then there exists $\bar{c}$ such that $\forall c \geq \bar{c}$, the function $L_{\text{ALAG}}(x, \bar{v})$ achieves a strict local minimum at $\bar{x}$.

In particular, if f is convex and h is linear, then any minimizing solution $\bar{x}$ for (P) also minimizes $L_{\text{ALAG}}(x, \bar{v})$ for all $c \geq 0$.

In practice, of course, the exact values of the KKT multipliers are not known in advance. Therefore, to make use of the above result, one attempts to estimate the multipliers by updating the vector v_k after solving each (or some) unconstrained minimizations of L_{ALAG}. The outline of such an algorithm is as follows:

Initialization Select the initial multipliers v_0 and penalty weight $c_0 > 0$. Set $k \leftarrow 0$

Iteration k, inner loop Solve the unconstrained problem to minimize $L_{\text{ALAG}}(x, v_k)$ and let x_k denote the optimal solution obtained. If termination criteria are satisfied, stop.

Iteration k, outer loop Obtain the updated multipliers v_{k+1} according to appropriate formulas, increase k, repeat iteration.

As you can see, the above description is extremely generic. In particular, the multipliers can be updated in numerous ways. Some of them are:

- *Constant:* Keep the multipliers constant. This version of the method is not significantly different from the usual quadratic penalty method.
- *Method of multipliers:* Let
$$v_{k+1} = v_k + 2c_k h(x_k).$$
The rationale for this method is provided by the following fact. Suppose the multipliers v_k in the augmented Lagrangian function are updated according to any rule such that the sequence $\{v_k\}$ is bounded. Suppose further that $c_k \to +\infty$, and $x_k \to x^\star$, where $x^\star$ is a KKT point with multipliers $v^\star$ (of course, we need some regularity conditions to hold for this to happen). Then
$$v_k + 2c_k h(x_k) \to v^\star$$
(this result is very similar to Lemma 56).
- Other multiplier update methods – second order, exponential, etc.

The study of convergence of such methods, and especially rates of convergence is fairly complicated. The text by Bertsekas contains some fairly detailed analysis and provides many references.

Augmented Lagrangian methods are implemented in several popular optimization codes.

12 Successive quadratic programming (SQP)

In this section we will describe some basic ideas of the Sequential Quadratic Programming (SQP) method for solving nonlinearly constrained optimization problems.[11] The problem we will consider is, as usual,

$$\begin{array}{lll} \text{(P)} & \min & f(x) \\ & \text{s.t.} & g(x) \leq 0 \\ & & h(x) = 0, \end{array}$$

where at least one of the constraints is nonlinear.

One of the ways to motivate SQP is to employ Newton's (or quasi-Newton) method to directly solve the KKT conditions for the problem (P). An alternative way to view the basic idea of SQP it to model (P) at a given approximate solution, say x_k, by a quadratic programming (QP) subproblem, and then use the solution to the subproblem to construct a better approximation x_{k+1}. This process is iterated to create a sequence of approximations that, it is hoped, will converge to a solution $x^\star$ of (P). We will attempt to demonstrate in this presentation that SQP methods can be viewed as the natural extension of Newton (or quasi-Newton) methods to constrained optimization setting, as mentioned above.

It should be noted that in general SQP is not a feasible-point method, i.e., its iterates need not be feasible (although the method can be easily modified so that if any *linear* constraints are present, they are always satisfied).

12.1 The basic SQP method

As usual, we define the Lagrangian function associated with problem (P) by

$$L(x, u, v) = f(x) + g(x)^T u + h(x)^T v.$$

For any feasible point x, let $G(x)$ denote the Jacobian of the function corresponding to active constraints (i.e., the rows of $G(x)$ are the (transposed) gradients of all $h_i(x)$'s and active $g_i(x)$'s). We will denote by $x^\star$ any particular local solution. We assume that the following conditions hold for any such solution:

A1 The first order necessary conditions are satisfied, i.e., there exists an optimal multiplier vector $(u^\star, v^\star)$ that together with $x^\star$ satisfies first order KKT conditions.

A2 $G(x^\star)$ has full row rank.

A3 Strict complementarity holds at $x^\star$ (i.e., if $g_i(x^\star) = 0$, then $u_i^\star > 0$).

A4 The Hessian of the Lagrangian function with respect to x is p.d. on the null space of $G(x^\star)$ (this is the second order sufficient condition for a strict local minimum).

At a current iterate x_k, we seek to (locally) approximate the problem (P) by a quadratic subproblem, i.e., an optimization problem with a quadratic objective function and linear constraints. We thus

[11] This discussion is based primarily on the review paper "Sequential Quadratic Programming" by Paul T. Boggs and Jon W. Tolle, *Acta Numerica*, 1995, 1–51.

will construct the subproblem by linearizing the constraints of (P) around x_k:

$$\begin{array}{ll} \min\limits_{d_x} & (r_k)^T d_x + \frac{1}{2} d_x^T B_k d_x \\ \text{s.t.} & \nabla g(x_k)^T d_x + g(x_k) \leq 0, \\ & \nabla h(x_k)^T d_x + h(x_k) = 0 \end{array}$$

where $d_x = x - x_k$. It remains to specify the vector and symmetric matrix to form the objective function. The most obvious choice would be the local quadratic approximation to f at x_k. However, the following choice allows us to take the nonlinearity of the constraints into account. Observe that conditions **A1**–**A4** imply that $x^\star$ is a local minimizer of the problem $\min_x\{L(x, u^\star, v^\star) : g(x) \leq 0,\ h(x) = 0\}$. Although the optimal multiplier vector is not known, an approximation (u_k, v_k) can be maintained as part of the algorithm. Given the current iterate (x_k, u_k, v_k), the quadratic approximation in x for the Lagrangian is

$$L(x_k, u_k, v_k) + \nabla L(x_k, u_k, v_k)^T d_x + \frac{1}{2} d_x^T \nabla^2 L(x_k, u_k, v_k) d_x,$$

and we can construct the quadratic subproblem by letting B_k to be an approximation of $\nabla^2 L(x_k, u_k, v_k)$, and $r_k = \nabla L(x_k, u_k, v_k)$. Another variation, most often used in literature, is

$$\begin{array}{lll} \text{(QP)} & \min\limits_{d_x} & \nabla f(x_k)^T d_x + \frac{1}{2} d_x^T B_k d_x \\ & \text{s.t.} & \nabla g(x_k)^T d_x + g(x_k) \leq 0, \\ & & \nabla h(x_k)^T d_x + h(x_k) = 0 \end{array} \tag{27}$$

(the above two choices of r_k are equivalent when only equality constraints are present in the problem (P)).

The solution d_x of (QP) can be used as a search direction for the next iterate x_{k+1}. We also need to update the estimates of the multipliers, which is done as follows: let (u^{QP}, v^{QP}) be the vector of optimal multipliers for (QP). Then we will define the corresponding search directions as $(d_u, d_v) = (u^{QP} - u_k, v^{QP} - v_k)$. We then choose a step-size α and define the next iterate to be $(x_{k+1}, u_{k+1}, v_{k+1}) = (x_k, u_k, v_k) + \alpha(d_x, d_u, d_v)$.

There are several issues of potential concern in the algorithm as described so far. First, the subproblems (QP) need to be feasible. While as a result of **A2** this will be the case at points x_k which are close to $x^\star$, but it will not necessarily be true at "non-local" points. Secondly, the subproblems need to have an optimal solution. This can be assured by the appropriate choice of matrices B_k. Finally, the algorithm has to be designed in such a way that its iterates converge to a desirable point. There are two aspects of convergence to consider: local and global. Local convergence concerns the behavior of iterates if the algorithm is started (or has found a point) near the optimum $x^\star$, and typically addresses rates of convergence. Global convergence analysis attempts to describe when the iterates of the algorithm will converge to an optimum solution starting from any point. This analysis is typically performed by the means of a *merit function* ϕ, i.e., a function whose reduction implies progress towards a solution.

With these considerations in mind, we now state the template for a basic SQP algorithm:

SQP Algorithm

Initialization: given (x_0, u_0, v_0), B_0 and a merit function ϕ, set $k = 0$.

1. Form and solve (QP) to obtain (d_x, d_u, d_v).
2. Choose α so that $\phi(x_k+\alpha d_x) < \phi(x_k)$ (alternatively, choose $\alpha \geq 0$ that minimizes $\phi(x_k+\alpha d_x)$).
3. Set $(x_{k+1}, u_{k+1}, v_{k+1}) = (x_k, u_k, v_k) + \alpha(d_x, d_u, d_v)$.
4. Stop if converged.
5. Compute B_{k+1}.
6. Set $k \leftarrow k+1$, go to 1.

12.2 Local convergence

The local convergence analysis establishes conditions under which the iterates of the algorithm converge to a solution and at what rate, provided that the starting data (e.g., $x_0,\ u_0,\ v_0,\ B_0$) are sufficiently close to the corresponding data at a solution $x^\star$.

We will make two assumptions to simplify the presentation. First, we assume that the active inequality constraints of (P) at $x^\star$ are known. (This assumption can be justified since it can be shown that under the assumption **A3** the problem (QP) at x_k will have the same active constraints as (P) at $x^\star$ if x_k is close to $x^\star$.) Under this assumption, only equality-constrained problems need be considered for local analysis, and we will eliminate the terms referring to $g(x)$ from the notation under this assumption. We can now rewrite the quadratic problem with equality constraints as

$$\begin{array}{lll} \text{(ECQP)} & \min & \nabla f(x_k)^T d_x + \frac{1}{2} d_x^T B_k d_x \\ & \text{s.t.} & \nabla h(x_k)^T d_x + h(x_k) = 0. \end{array} \tag{28}$$

The second assumption arises from the fact that we will use the properties of the (pure) Newton's method to analyze the iterates of the algorithm. We therefore assume that the merit function $\phi(x)$ is such that it allows $\alpha = 1$ to be used as the stepsize.

For an equality constrained problem, the KKT conditions (under appropriate assumptions) lead to the following formulas for the optimal multipliers:

$$v^\star = -\left[\nabla h(x^\star)^T \nabla h(x^\star)\right]^{-1} \nabla h(x^\star)^T \nabla f(x^\star).$$

By the smoothness assumption, the vector

$$v_0 = -\left[\nabla h(x_0)^T \nabla h(x_0)\right]^{-1} \nabla h(x_0)^T \nabla f(x_0) \tag{29}$$

can be made arbitrarily close to $v^\star$ by choosing x_0 close to $x^\star$.

From the first order optimality conditions for (ECQP), we obtain the equations for computing the update direction (d_x, d_v):

$$\begin{array}{rcl} B_k d_x + \nabla h(x_k) d_v &=& -\nabla L(x_k, v_k) \\ \nabla h(x_k)^T d_x &=& -h(x_k) \end{array} \tag{30}$$

(here, $d_v = v_{k+1} - v_k = v^{QP} - v_k$).

12.2.1 The Newton SQP method

The most straightforward method is obtained by setting $B_k = \nabla^2 L(x_k, v_k)$. Then the SQP method can be viewed as an application of Newton's method to solve the system of nonlinear equations

obtained from the KKT conditions:

$$\Psi(x,v) = \begin{bmatrix} \nabla L(x,v) \\ h(x) \end{bmatrix} = 0.$$

Assuming that x_0 is close to $x^\star$ it follows that v_0 is close to $v^\star$, and hence $\nabla^2 L(x_0, v_0)$ is close to $\nabla^2 L(x^\star, v^\star)$. From assumptions **A1** and **A4**, the Jacobian of this system, which is given by

$$\nabla\Psi(x,v) = \begin{bmatrix} \nabla^2 L(x,v) & \nabla h(x) \\ \nabla h(x)^T & 0 \end{bmatrix}$$

is nonsingular at $(x^\star, v^\star)$, and we can apply the Newton iteration scheme (whose iterates are identical to those generated by the Newton SQP method). The following theorem is an immediate result of the local analysis of Newton's method:

Theorem 59 *Let x_0 be an initial estimate of the solution to (P) and let v_0 be given by (29). Then if $\|x_0 - x^\star\|$ is sufficiently small, the sequence of iterates (x_k, v_k) generated by the Newton SQP method is well-defined and converges quadratically to the pair $(x^\star, v^\star)$.*

12.2.2 Quasi-Newton approximations

The SQP method as presented above has several disadvantages. First, it requires evaluation of second-order derivatives at every iteration, which can be computationally expensive. Secondly, the Hessians $\nabla^2 L(x_k, v_k)$ might not be positive definite, and as a result, the corresponding quadratic problem will be hard to solve. Both these issues can be overcome by using positive definite approximations B_k for $\nabla^2 L(x_k, v_k)$ which are easier to compute and update.

There is a lot of theoretical work that goes into designing an update scheme for the matrices B_k. We would like these matrices to be positive definite (at least on the null space of $\nabla h(x_k)$). It is also desirable for these matrices to be good approximations of $\nabla L(x^\star, v^\star)$ in the limit (at least, again, when acting on vectors in the null space of $\nabla h(x_k)$).

One of the ways to generate B_k's is to use the quasi-Newton methods which we discussed in the framework of unconstrained optimization. The main difference is the usage of gradients of L (instead of simply f) to construct the updates of the matrices. For example, we could use the BFGS method and update the matrices as follows:

$$B_{k+1} = B_k + \frac{qq^T}{q^T p} - \frac{B_k p p^T B_k}{p^T B_k p},$$

where

$$p = x_{k+1} - x_k,\ q = \nabla L(x_{k+1}, v_{k+1}) - \nabla L(x_k, v_{k+1}).$$

When using this formula, the matrices B_k remain positive definite so long as $p^T q > 0$. It can be shown that if $x^\star$ satisfies assumption **A4** then the algorithm SQP with this choice of matrices B_k and stepsizes $\alpha_k = 1$ will exhibit local superlinear convergence.

12.3 Global convergence

In this subsection we discuss how the *merit function* $\phi(x)$ assures that the iterates of the SQP algorithm eventually get close to $x^\star$, thus making the results of the previous discussion applicable.

The standard way to ensure that a reduction in ϕ indicates progress is to construct ϕ so that the solutions of (P) are unconstrained minimizers of ϕ. This brings us back to the idea of *penalty functions*, and indeed, some of the most popular merit functions are the augmented Lagrangian function and the linear penalty function. Another important attribute of the merit function is that it should guide the iterates of the algorithm and provide a measure of progress by means of exhibiting a descent.

12.3.1 l_1 (linear) penalty merit function

First, we consider the l_1 penalty merit function. We denote

$$\phi_1(x;\mu) = f(x) + \mu \left[\sum_{i=1}^{m} g_i(x)^+ + \sum_{i=1}^{l} |h_i(x)| \right].$$

The following lemma establishes that $\phi_1(x;\mu)$ can indeed be used as a merit function:

Lemma 60 (10.4.1) *Given an iterate x_k consider the quadratic subproblem (27), where B_k is any positive-definite matrix. Let d_x solve this problem. If $d \neq 0$ and $\mu \geq \max\left\{u_1^{QP}, \ldots, u_m^{QP}, v_1^{QP}, \ldots, v_l^{QP}\right\}$, then d_x is a descent direction at x_k for the function $\phi_1(x;\mu)$.*

Thus, the algorithm SQP can be applied using $\phi_1(x;\mu)$ as a merit function (of course, just as in the penalty method, we have to overcome the difficulty that comes with optimizing a non-differentiable function ϕ_1). The following theorem demonstrates its global convergence:

Theorem 61 (10.4.2) *Assume that the sequence of points $\{x_k\}$ generated by the algorithm SQP with $\phi = \phi_1$ is contained in a compact subset X of $\mathbb{R}^n$, and that for any point $x \in X$ and any $B \succ 0$ the corresponding QP has a unique solution and unique KKT multipliers (u^{QP}, v^{QP}) satisfying $\mu \geq \max\left\{u_1^{QP}, \ldots, u_m^{QP}, v_1^{QP}, \ldots, v_l^{QP}\right\}$. Furthermore, assume that all the matrices B_k are uniformly bounded and uniformly positive definite, i.e.,*

$$\exists \beta_1 > 0, \beta_2 > 0 : \ \|B_k\| \leq \beta_1, \ d^T B_k d \geq \beta_2 \|d\|^2 \ \forall k.$$

Then every limit point of $\{x_k\}$ is a KKT point of (P).

Recall that to obtain local convergence rate results for Newton and quasi-Newton versions of the SQP algorithm we had to assume that the stepsize of 1 is always used near the optimum. For the general implementation of the algorithm, however, the line searches continue throughout the algorithm. The merit function, therefore, must allow a steplength of 1 near the optimum when the matrices B_k are chosen appropriately. A significant disadvantage of the merit function $\phi_1(x;\mu)$ is that it may not allow the steplength $\alpha = 1$ near the solution. This phenomenon, known as the Maratos effect, precludes us from obtaining quadratic or superlinear convergence, unless additional modifications are made in the algorithm.

There are several ways to deal with this phenomenon. For example, one may require the merit function to exhibit a decrease over a sequence of several iterations, but not necessarily at every iteration (if the function decreases over a fixed number of iterations, convergence is still guaranteed). One of the ways of implementing it is to accept the steplength $\alpha = 1$ even if it leads to an increase in ϕ_1 if you "suspect" you are close to the solution. If taking such steps does not lead to a decrease in ϕ_1 after a small number of iterations, restore the original iterate and perform a line search.

12.3.2 Augmented Lagrangian merit function

To simplify the presentation, we will restrict our attention to the problems with only equality constraints. We will discuss the following (familiar) version of the augmented Lagrangian function (note the change of notation, however):

$$\phi_F(x;\eta) = f(x) + h(x)^T \bar{v}(x) + \frac{\eta}{2}\|h(x)\|_2^2,$$

where η is a constant to be determined, and

$$\bar{v}(x) = -\left[\nabla h(x)^T \nabla h(x)\right]^{-1} \nabla h(x)^T \nabla f(x).$$

$\bar{v}(x)$ is the estimate of the KKT multipliers; note that $\bar{v}(x^\star) = v^\star$.

The use of $\phi_F(x;\eta)$ as a merit function is justified by the following theorem:

Theorem 62 *Suppose that all the matrices B_k are uniformly bounded and uniformly positive definite. Suppose, furthermore, that the starting point x_0 and all the succeeding iterates lie in some compact set $\mathcal{C}$ such that the rows of $\nabla h(x)^T$ are linearly independent for all $x \in \mathcal{C}$. Then for η sufficiently large, we have:*

1. *$x^\star \in \mathcal{C}$ is a strong local minimum[12] of ϕ_F iff $x^\star$ is a strong local minimum of (P), and*
2. *if x is not a KKT point of (P), then d_x is a descent direction for ϕ_F.*

In fact, it can be also shown that when the line searches are performed appropriately (i.e., satisfy a condition similar to the Armijo rule) and η is sufficiently large, the iterates $\{x_k\}$ will converge to a KKT point of (P). Since in practice it is not known in advance how large η needs to be, it is necessary to employ a strategy for adjusting η (same is true for the merit function ϕ_1, where the strategy that should be used is fairly transparent).

In addition to differentiability, the augmented Lagrangian penalty function exhibits a decrease with the stepsize $\alpha = 1$ when the iterates converge superlinearly.

In order to extend the augmented Lagrangian approach to handle inequality constraints, one could employ a version of an *active set strategy*. Another option is to consider the following problem:

$$\begin{array}{ll} \min\limits_{x,t} & f(x) \\ \text{s.t.} & h(x) = 0 \\ & g_i(x) + t_i^2 = 0,\ i = 1,\ldots,m. \end{array}$$

where t is a vector of slack variables. This problem is equivalent to (P) in the sense that both have a strong local solution at $x^\star$.

12.4 Some final issues

First note of caution is to observe that all the global convergence results above only guarantee convergence to a KKT point of (P), which may not be a global, or even local, minimum.

[12] A local minimizer is strong, or isolated, if it is the only local minimizer in some ϵ-neighborhood.

Recall that we have always made the assumption (either implicitly or explicitly) that the problem (QP) always has a solution. This might not be the case: (QP) may be infeasible, or unbounded. When implementing a practical algorithm one must consider ways to deal with these difficulties when they arise.

One technique to avoid infeasibilities is to relax the constraints of the (QP) by using *trust region methods*. Another possibility is to take d_x to be some convenient direction when (QP) is infeasible, for example, the steepest descent direction of the merit function. Also, the output of the algorithm used to find a feasible point of (QP) (if found, used to initialize the algorithm for solving (QP)) can often be useful in determining a good direction d_x even if it demonstrates that (QP) is infeasible.

When the problem (QP) is unbounded, a change in B_k (for example, adding a positive multiple of the identity matrix) can be used to ensure that it is positive definite.

13 Barrier Methods

Consider the constrained optimization problem (P):

$$\begin{array}{lll} \text{(P)} & \min & f(x) \\ & \text{s.t.} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & & x \in \mathbb{R}^n, \end{array}$$

whose feasible region we denote by $S = \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, \ldots, m\}$. Barrier methods are also, like penalty methods, designed to solve (P) by instead solving a sequence of specially constructed unconstrained optimization problems.

In a barrier method, we presume that we are given a point x_0 that lies in the interior of the feasible region S, and we impose a very large cost on *feasible* points that lie ever closer to the boundary of S, thereby creating a "barrier" to exiting the feasible region.

Definition. A *barrier function* for (P) is any function $b(x) : \mathbb{R}^n \to \mathbb{R}$ that satisfies

- $b(x) \geq 0$ for all x that satisfy $g(x) < 0$, and
- $b(x) \to \infty$ as $\lim_x \max_i\{g_i(x)\} \to 0$.

The idea in a barrier method is to dissuade points x from ever approaching the boundary of the feasible region. We consider solving:

$$\begin{array}{lll} B(c) & \min & f(x) + \frac{1}{c} b(x) \\ & \text{s.t.} & g(x) < 0, \\ & & x \in \mathbb{R}^n. \end{array}$$

for a sequence of $c_k \to +\infty$. Note that the constraints "$g(x) < 0$" are effectively unimportant in $B(c)$, as they are never binding in $B(c)$.

Example:

$$b(x) = \sum_{i=1}^{m} \frac{1}{-g_i(x)}$$

Suppose $g(x) = (x - 4, 1 - x)^T, \quad x \in \mathbb{R}^1$. Then

$$b(x) = \frac{1}{4 - x} + \frac{1}{x - 1}.$$

Let $r(c, x) = f(x) + \frac{1}{c} b(x)$. Let the sequence $\{c_k\}$ satisfy $c_{k+1} > c_k$ and $c_k \to \infty$ as $k \to \infty$. Let x_k denote the exact solution to $B(c_k)$, and let $x^\star$ denote any optimal solution of (P).

The following Lemma presents some basic properties of barrier methods.

Lemma 63 (Barrier Lemma) *1.* $r(c_k, x_k) \geq r(c_{k+1}, x_{k+1})$

2. $b(x_k) \leq b(x_{k+1})$

3. $f(x_k) \geq f(x_{k+1})$

4. $f(x^\star) \leq f(x_k) \leq r(c_k, x_k)$.

Proof:

1.
$$r(c_k, x_k) = f(x_k) + \frac{1}{c_k} b(x_k) \geq f(x_k) + \frac{1}{c_{k+1}} b(x_k)$$
$$\geq f(x_{k+1}) + \frac{1}{c_{k+1}} b(x_{k+1}) = r(c_{k+1}, x_{k+1})$$

2.
$$f(x_k) + \frac{1}{c_k} b(x_k) \leq f(x_{k+1}) + \frac{1}{c_k} b(x_{k+1})$$
and
$$f(x_{k+1}) + \frac{1}{c_{k+1}} b(x_{k+1}) \leq f(x_k) + \frac{1}{c_{k+1}} b(x_k).$$
Summing and rearranging, we have
$$\left(\frac{1}{c_k} - \frac{1}{c_{k+1}} \right) b(x_k) \leq \left(\frac{1}{c_k} - \frac{1}{c_{k+1}} \right) b(x_{k+1}).$$
Since $c_k < c_{k+1}$, it follows that $b(x_{k+1}) \geq b(x_k)$.

3. From the proof of (1.),
$$f(x_k) + \frac{1}{c_{k+1}} b(x_k) \geq f(x_{k+1}) + \frac{1}{c_{k+1}} b(x_{k+1}).$$
But from (2.), $b(x_k) \leq b(x_{k+1})$. Thus $f(x_k) \geq f(x_{k+1})$.

4. $f(x^\star) \leq f(x_k) \leq f(x_k) + \frac{1}{c_k} b(x_k) = r(c_k, x_k)$.

∎

Let $N(x, \epsilon)$ denote the ball of radius ϵ centered at the point x. The next result concerns convergence of the barrier method.

Theorem 64 ***(Barrier Convergence Theorem).*** *Suppose $f(x)$, $g(x)$, and $b(x)$ are continuous functions. Let $\{x_k\}$, $k = 1, \ldots, \infty$, be a sequence of solutions of $B(c_k)$. Suppose there exists an optimal solution $x^\star$ of (P) for which $N(x^\star, \epsilon) \cap \{x \mid g(x) < 0\} \neq \emptyset$ for every $\epsilon > 0$. Then any limit point $\bar{x}$ of $\{x_k\}$ solves (P).*

Proof: Let $\bar{x}$ be any limit point of the sequence $\{x_k\}$. From the continuity of $f(x)$ and $g(x)$, $\lim_{k\to\infty} f(x_k) = f(\bar{x})$ and $\lim_{k\to\infty} g(x_k) = g(\bar{x}) \leq 0$. Thus $\bar{x}$ is feasible for (P).

For any $\epsilon > 0$, there exists $\tilde{x}$ such that $g(\tilde{x}) < 0$ and $f(\tilde{x}) \leq f(x^\star) + \epsilon$. For each k,

$$f(x^\star) + \epsilon + \frac{1}{c_k} b(\tilde{x}) \geq f(\tilde{x}) + \frac{1}{c_k} b(\tilde{x}) \geq r(c_k, x_k).$$

Therefore for k sufficiently large, $f(x^\star) + 2\epsilon \geq r(c_k, x_k)$, and since $r(c_k, x_k) \geq f(x^\star)$ from (4.) of the Barrier Lemma, then

$$f(x^\star) + 2\epsilon \geq \lim_{k\to\infty} r(c_k, x_k) \geq f(x^\star).$$

This implies that

$$\lim_{k\to\infty} r(c_k, x_k) = f(x^\star).$$

We also have

$$f(x^\star) \le f(x_k) \le f(x_k) + \frac{1}{c_k} b(x_k) = r(c_k, x_k).$$

Taking limits we obtain

$$f(x^\star) \le f(\bar{x}) \le f(x^\star),$$

whereby $\bar{x}$ is an optimal solution of (P). ∎

A typical class of barrier functions are:

$$b(x) = \sum_{i=1}^{m} (-g_i(x))^{-q}, \quad \text{where} \quad q > 0$$

along with

$$b(x) = -\sum_{i=1}^{m} \ln(\min\{1, -g_i(x)\}).$$

Note that the second barrier function is not differentiable. Actually, since the properties of the barrier function are only essential near the boundary of the feasible region, the following barrier function is the most commonly used one:

$$b(x) = -\sum_{i=1}^{m} \ln(1, -g_i(x)).$$

If can be shown that the above convergence properties apply to this function as well.

13.1 Karush-Kuhn-Tucker multipliers in barrier methods

Let

$$b(x) = \gamma(g(x)),$$

where $\gamma(y) : \mathbb{R}^m \to \mathbb{R}$, and assume that $\gamma(y)$ is continuously differentiable for all $y < 0$. Then

$$\nabla b(x) = \sum_{i=1}^{m} \frac{\partial \gamma(g(x))}{\partial y_i} \nabla g_i(x),$$

and if x_k solves $B(c_k)$, then $\nabla f(x_k) + \frac{1}{c_k} \nabla b(x_k) = 0$, that is,

$$\nabla f(x_k) + \frac{1}{c_k} \sum_{i=1}^{m} \frac{\partial \gamma(g(x_k))}{\partial y_i} \nabla g_i(x_k) = 0. \tag{31}$$

Let us define

$$[u_k]_i = \frac{1}{c_k} \frac{\partial \gamma(g(x_k))}{\partial y_i}. \tag{32}$$

Then (42) becomes:

$$\nabla f(x_k) + \sum_{i=1}^{m} [u_k]_i \nabla g_i(x_k) = 0. \tag{33}$$

Therefore we can interpret the u_k as a sort of vector of Karush-Kuhn-Tucker multipliers. In fact, we have:

Lemma 65 *Let (P) satisfy the conditions of the Barrier Convergence Theorem. Suppose $\gamma(y)$ is continuously differentiable and let u_k be defined by (32). Then if $x_k \to \bar{x}$, and $\bar{x}$ satisfies the linear independence condition for gradient vectors of active constraints, then $u_k \to \bar{u}$, where $\bar{u}$ is a vector of Karush-Kuhn-Tucker multipliers for the optimal solution $\bar{x}$ of (P).*

Proof: Let $x_k \to \bar{x}$ and let $I = \{i \mid g_i(\bar{x}) = 0\}$ and $N = \{i \mid g_i(\bar{x}) < 0\}$. For all $i \in N$,

$$[u_k]_i = \frac{1}{c_k} \frac{\partial \gamma(g(x_k))}{\partial y_i} \to 0 \quad \text{as} \quad k \to \infty,$$

since $c_k \to \infty$ and $g_i(x_k) \to g_i(\bar{x}) < 0$, and $\frac{\partial \gamma(g(\bar{x}))}{\partial y_i}$ is finite. Also $[u_k]_i \geq 0$ for all i, and k sufficiently large.

Suppose $u_k \to \bar{u}$ as $k \to \infty$. Then $\bar{u} \geq 0$, and $\bar{u}_i = 0$ for all $i \in N$. From the continuity of all functions involved, (33) implies that

$$\nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{u}_i \nabla g_i(\bar{x}) = 0, \quad \bar{u} \geq 0, \quad \bar{u}^T g(\bar{x}) = 0.$$

Thus $\bar{u}$ is a vector of Kuhn-Tucker multipliers. It remains to show that $u_k \to \bar{u}$ for some unique $\bar{u}$. The proof that $u_k \to \bar{u}$ for some unique $\bar{u}$ is exactly as in Lemma 56. ∎

14 Duality theory of nonlinear programming

14.1 The practical importance of duality

Duality arises in nonlinear (and linear) optimization models in a wide variety of settings. Some immediate examples of duality are in:

Models of electrical networks The current flows are "primal variables" and the voltage differences are the "dual variables" that arise in consideration of optimization (and equilibrium) in electrical networks.

Models of economic markets In these models, the "primal" variables are production levels and consumption levels, and the "dual" variables are prices of goods and services.

Structural design In these models, the tensions on the beams are "primal" variables, and the nodal displacements are the "dual" variables.

Nonlinear (and linear) duality is very useful. For example, dual problems and their solutions are used in connection with:

Identifying near-optimal solutions A good dual solution can be used to bound the values of primal solutions, and so can be used to actually identify when a primal solution is near-optimal.

Proving optimality Using a strong duality theorem, one can prove optimality of a primal solution by constructing a dual solution with the same objective function value.

Sensitivity analysis of the primal problem The dual variable on a constraint represents the incremental change in the optimal solution value per unit increase in the RHS of the constraint.

Karush-Kuhn-Tucker conditions The optimal solution to the dual problem is a vector of KKT multipliers.

Convergence of improvement algorithms The dual problem is often used in the convergence analysis of algorithms.

Good structure Quite often, the dual problem has some good mathematical, geometric, or computational structure that can exploited in computing solutions to both the primal and the dual problem.

Other uses, too...

14.2 Definition of the dual problem

We will consider the *primal* problem in the following form:

$$\begin{array}{lll} \text{(P)} & \inf & f(x) \\ & \text{s.t.} & g(x) \leq 0 \\ & & x \in X, \end{array}$$

where $g(x) : \mathbb{R}^n \to \mathbb{R}^m$. NOTE: unlike in our previous development, X no longer has to be an open set. Indeed, it can now be any set, including, potentially, the feasible region of any equality

constraints or other inequality constraints and restrictions that may be present in the problem.[13] For example, one could have

$$X = \{x \in \mathbb{R}^n : x \text{ is integer}\},$$

or

$$X = \{x \in \mathbb{R}^n : k_i(x) \leq 0,\ i = 1, \ldots, j;\ h_i(x) = 0,\ i = 1, \ldots, l\}.$$

Let $z^\star = \inf\{f(x) : g(x) \leq 0,\ x \in X\}$. (By convention, if (P) is infeasible, we set $z^\star = +\infty$, and if (P) is unbounded, $z^\star = -\infty$. Note that here we do not assume that, even if $z^\star$ is finite, the optimal value is attained.)

For a nonnegative m-vector u, the Lagrangian function is

$$L(x, u) = f(x) + u^T g(x).$$

Define

$$L^\star(u) = \inf_{x \in X} L(x, u) = \inf_{x \in X} (f(x) + u^T g(x)).$$

Notice that the optimization problem above may not attain its optimal value since X is not guaranteed to be a compact set; hence we use inf rather than min. If for some value of $u \geq 0$ the Lagrangian $L(x, u)$ is unbounded below over X, we have $L^\star(u) = -\infty$. The function $L^\star(u)$ is called the dual function. We presume that computing $L^\star(u)$ is an easy task for any u.

The dual problem (D) is defined to be:

$$(\text{D}) \sup_{u \geq 0} L^\star(u).$$

Let $v^\star$ be the optimal value (finite or infinite, attained or not) of the dual problem.

14.2.1 Problems with different formats of constraints

If the primal problem has the form

$$\begin{array}{lll} (\text{P}) & \inf & f(x) \\ & \text{s.t.} & g(x)_i \leq 0,\ i \in L \\ & & g(x)_i \geq 0,\ i \in G \\ & & g(x)_i = 0,\ i \in E \\ & & x \in X, \end{array}$$

we can still form the Lagrangian as

$$L(x, u) = f(x) + \sum_{i \in L} g_i(x) u_i + \sum_{i \in G} g_i(x) u_i + \sum_{i \in E} g_i(x) u_i$$

and construct the dual function $L^\star(u) = \inf_{x \in X} L(x, u)$. The only difference is the form of the dual problem:

$$\begin{array}{lll} (\text{D}) & \sup & L^\star(u) \\ & \text{s.t.} & u_i \geq 0,\ i \in L \\ & & u_i \leq 0,\ i \in G \\ & & u_i \text{ unrestricted},\ i \in E \end{array}$$

For simplicity, when studying the theory of duality, we will assume that the constraints of the primal problem are all in "$\leq$" form, but all results we develop apply to the general case as well.

[13] The text deals with equality constraints explicitly, but we will simplify the presentation.

14.3 Examples

14.3.1 The dual of a linear program

$$\begin{array}{lll} \text{(LP)} & \inf & c^T x \\ & \text{s.t.} & b - Ax \leq 0 \\ & & x \in \mathbb{R}^n. \end{array}$$

$$L(x,u) = c^T x + u^T(b - Ax) = u^T b + (c - A^T u)^T x.$$

$$L^\star(u) = \inf_{x \in \mathbb{R}^n} L(x,u) = \begin{cases} -\infty, & \text{if } A^T u \neq c \\ u^T b, & \text{if } A^T u = c. \end{cases}$$

The dual problem (D) is:

$$\text{(D)} \ \sup_{u \geq 0} L^\star(u) = \sup u^T b \text{ s.t. } A^T u = c, \ u \geq 0.$$

14.3.2 The dual of a binary integer program

$$\begin{array}{lll} \text{(IP)} & \inf & c^T x \\ & \text{s.t.} & b - Ax \leq 0 \\ & & x_j \in \{0,1\}, \ j = 1, \ldots, n. \end{array}$$

$$L(x,u) = c^T x + u^T(b - Ax) = u^T b + (c - A^T u)^T x.$$

$$L^\star(u) = \inf_{x \in \{0,1\}^n} L(x,u) = u^T b + \sum_{j:(c - A^T u)_j < 0} (c - A^T u)_j$$

The dual problem (D) is:

$$\text{(D)} \ \sup_{u \geq 0} L^\star(u) = \sup_{u \geq 0} u^T b + \sum_{j:(c - A^T u)_j < 0} (c - A^T u)_j.$$

14.3.3 The dual of a quadratic problem

$$\begin{array}{lll} \text{(LP)} & \inf & \frac{1}{2} x^T Q x + c^T x \\ & \text{s.t.} & b - Ax \leq 0 \\ & & x \in \mathbb{R}^n. \end{array}$$

$$L(x,u) = \frac{1}{2} x^T Q x + c^T x + u^T(b - Ax) = u^T b + (c - A^T u)^T x + \frac{1}{2} x^T Q x.$$

$$L^\star(u) = \inf_{x \in \mathbb{R}^n} L(x,u).$$

Assuming that Q is positive definite, we solve: $Q\tilde{x} = -(c - A^T u)$, i.e., $\tilde{x} = -Q^{-1}(c - A^T u)$ and

$$L^\star(u) = L(\tilde{x}, u) = -\frac{1}{2}(c - A^T u)^T Q^{-1}(c - A^T u) + u^T b.$$

The dual problem (D) is:

$$\text{(D)} \ \sup_{u \geq 0} L^\star(u) = \sup_{u \geq 0} -\frac{1}{2}(c - A^T u)^T Q^{-1}(c - A^T u) + u^T b.$$

14.3.4 Dual of a log-barrier problem

Consider the following example of a log-barrier problem:

$$\begin{array}{lll} \text{(BP)} & \inf & 5x_1 + 7x_2 - 4x_3 - \sum_{j=1}^3 \ln(x_j) \\ & \text{s.t.} & x_1 + 3x_2 + 12x_3 = 37 \\ & & x > 0 \end{array}$$

What is the dual of this problem?

14.4 Geometry of the dual

Let $I = \{(s, z) \in \mathbb{R}^{m+1} : g(x) \leq s \text{ and } f(x) \leq z \text{ for some } x \in X\}$.

Let $H(u, b) = \{(s, z) \in \mathbb{R}^{m+1} : u^T s + z = b\}$. $H(u, b)$ is a hyperplane, and it is a *lower support* of I if $u^T s + z \geq b$ for all $(s, z) \in I$.

Consider now the following optimization problem, in which we determine a hyperplane $H(u, b)$ that is a lower support of I, and whose intersection with the $m + 1$st axis (i.e., intercept) is the highest:

$$\begin{aligned} &\sup_{u,b}\{b : H(u, b) \text{ is a lower support of } I\} \\ &= \sup_{u,b}\{b : b \leq u^T s + z \ \forall (s, z) \in I\} \\ &= \sup_{u \geq 0, b}\{b : b \leq u^T s + z \ \forall (s, z) \in I\} \\ &= \sup_{u \geq 0, b}\{b : b \leq u^T g(x) + f(x) \ \forall x \in X\} \\ &= \sup_{u \geq 0, b}\{b : b \leq L(x, u) \ \forall x \in X\} \\ &= \sup_{u \geq 0, b}\{b : b \leq \inf_{x \in X} L(x, u)\} = \sup_{u \geq 0} L^\star(u). \end{aligned}$$

The last expression is exactly the dual problem. Therefore, the dual problem can be geometrically interpreted as finding a hyperplane $H(u, b)$ that is a lower support of I and whose intercept is the highest. This highest value is exactly the optimal value of the dual problem.

Note that if X is a convex set and f and g_i's are convex, then I is a convex set.

14.5 Properties of the dual and weak duality

Proposition 66 *The dual is a concave maximization problem.*

Proof: It suffices to show that $L^\star(u)$ is a concave function on the nonnegative orthant. Let $u^1 \geq 0$ and $u^2 \geq 0$ be given, and let $u^3 = \lambda u^1 + (1 - \lambda)u^2$, $\lambda \in [0, 1]$. Then

$$L(u^3) = \inf_{x \in X} f(x) + (u^3)^T g(x) = \inf_{x \in X} (\lambda(f(x) + (u^1)^T g(x)) + (1 - \lambda)(f(x) + (u^2)^T g(x))$$

$$\geq \lambda \inf_{x\in X}(f(x) + (u^1)^T g(x)) + (1-\lambda)\inf_{x\in X}(f(x) + (u^2)^T g(x))$$
$$= \lambda L^\star(u^1) + (1-\lambda)L^\star(u^2). \ \blacksquare$$

Theorem 67 (Weak Duality Theorem) *If $\bar{x}$ is feasible in (P) and $\bar{u}$ is feasible in (D), then $f(\bar{x}) \geq L^\star(\bar{u})$.*

Proof:
$$f(\bar{x}) \geq f(\bar{x}) + \bar{u}^T g(\bar{x}) = L(\bar{x},\bar{u}) \geq \inf_{x\in X} L(x,\bar{u}) = L^\star(\bar{u}). \ \blacksquare$$

Corollary 68
$$\inf\{f(x) : g(x) \leq 0,\ x \in X\} \geq \sup\{L^\star(u) : u \geq 0\}$$

Corollary 69 *If $\bar{x} \in X$ satisfying $g(\bar{x}) \leq 0$ and $\bar{u} \geq 0$ are such that $f(\bar{x}) = L^\star(\bar{u})$, then $\bar{x}$ and $\bar{u}$ solve the primal and the dual problem, respectively.*

Corollary 70 *If $\inf\{f(x) : g(x) \leq 0,\ x \in X\} = -\infty$, then $L^\star(u) = -\infty$ for any $u \geq 0$.*

Corollary 71 *If $\sup\{L^\star(u) : u \geq 0\} = \infty$, then the primal problem is infeasible.*

Unlike in Linear Programming theory, the strong duality theorem cannot always be established for general optimization problems.

14.6 Saddlepoint optimality criteria

The pair $(\bar{x},\bar{u}) \in X \times \mathbb{R}^m_+$ is called a *saddle point* of the Lagrangian if

$$L(\bar{x},u) \leq L(\bar{x},\bar{u}) \leq L(x,\bar{u}) \text{ for all } x \in X,\ u \geq 0.$$

Theorem 72 (BSS 6.2.5) *Let $(\bar{x},\bar{u}) \in X \times \mathbb{R}^m_+$. Then the following three conditions are equivalent:*

(a) $(\bar{x},\bar{u})$ is a saddle point of the Lagrangian

(b) $(\bar{x},\bar{u})$ satifies

1. *$L(\bar{x},\bar{u}) = L^\star(\bar{u})$*
2. *$g(\bar{x}) \leq 0$, and*
3. *$\bar{u}^T g(\bar{x}) = 0$.*

(c) $\bar{x}$ and $\bar{u}$ are, respectively, optimal solutions to the primal and dual problems with no duality gap*, that is, with $f(\bar{x}) = L^\star(\bar{u})$.*

Proof: Suppose that $(\bar{x},\bar{u})$ is a saddle point. By definition, condition *1* must be true. Also, for any $u \geq 0$,
$$f(\bar{x}) + \bar{u}^T g(\bar{x}) \geq f(\bar{x}) + u^T g(\bar{x}).$$
This implies that $g(\bar{x}) \leq 0$, since otherwise the above inequality can be violated by picking $u \geq 0$ appropriately — this establishes *2*. Taking $u = 0$, we get $\bar{u}^T g(\bar{x}) \geq 0$; hence, $\bar{u}^T g(\bar{x}) = 0$, establishing *3*. Therefore, (a) implies (b).

If (b) holds, then $\bar{x}$ and $\bar{u}$ are feasible for their respective problems, and $f(\bar{x}) = L(\bar{x}, \bar{u}) = L^\star(\bar{u})$, which implies that they are optimal solutions of the respective problems, and there is no duality gap. That is, (b) implies (c).

Suppose now that (b) is satisfied. Then $L(\bar{x}, \bar{u}) \leq L(x, \bar{u})$ for all $x \in X$ by *1*. Furthermore,

$$L(\bar{x}, \bar{u}) = f(\bar{x}) \geq L(\bar{x}, u) \ \forall u \geq 0.$$

Thus $(\bar{x}, \bar{u})$ is a saddle point.

Finally, suppose (c) holds, i.e., $\bar{x}$ and $\bar{u}$ are optimal with no duality gap. Primal-dual feasibility implies that

$$L^\star(\bar{u}) \leq f(\bar{x}) + \bar{u}^T g(\bar{x}) \leq f(\bar{x}).$$

Since there is no duality gap, equality holds throughout, implying conditions *1–3*, and hence $(\bar{x}, \bar{u})$ is a saddle point. ∎

If strong duality holds and a dual optimal solution $\bar{u}$ exists, then any primal optimal point is also a minimizer of $L(x, \bar{u})$. This fact sometimes allows us to compute a primal optimal solution from a dual optimal solution.

Suppose that strong duality holds and $\bar{u}$ is known. Suppose further that $L(x, \bar{u})$ has a unique minimizer over the set X (this occurs, for instance, if X is a convex open set and $L(x, \bar{u})$ is a strictly convex function of x). Then if solution of $\min_{x \in X} L(x, \bar{u})$ is primal feasible, it must be primal optimal; if it is not primal feasible, then no primal optimal solution point can exists, i.e., $z^\star$ is only an inf, not a min. This observation is useful when the dual problem is easier to solve then the primal — we'll see some examples shortly.

14.7 Strong duality for convex optimization problems

Note that up until this point we made no assumptions about the functions $f(x)$ and $g(x)$, or the structure of the set X. We do need to impose some assumptions for the following result:

Theorem 73 (KKT and saddle point optimality criteria) *Suppose X is an open convex set, and f and g_i's are convex and differentiable. Then KKT conditions and saddle point optimality conditions are equivalent.*

Proof: Suppose $\bar{x}$ and $\bar{u}$ together satisfy the KKT optimality conditions for (P). Then *1* and *2* in Theorem 72 are satisfied, and

$$\nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{u}_i \nabla g_i(\bar{x}) = 0,$$

or

$$\nabla_x L(\bar{x}, \bar{u}) = 0.$$

Since $L(x, u)$ is convex in x for any u, this means that $L^\star(\bar{u}) = L(\bar{x}, \bar{u})$, establishing *3*. Thus $(\bar{x}, \bar{u})$ is a saddle point.

Conversely, if $(\bar{x}, \bar{u})$ is a saddle point, then *1–3* hold by Theorem 72. As we discussed above, these can be seen, for convex problem (P), as equivalent to the KKT conditions. ∎

The above theorem shows that if, in a convex problem (P), the optimal solution satisfies KKT conditions, then (P) has a strong dual, i.e., there is no duality gap between (P) and (D). However,

as we know, KKT conditions may not hold at the optimal solution if the problem does not satisfy a constraint qualification of some sort. For example, we can use the "linear constraint" CQ to prove the following result from linear programming:

Corollary 74 (Strong duality in linear programming) *Suppose (P) is an LP. Then (D) is also an LP, and exactly one of the following holds:*

(i) (P) and (D) are both infeasible (i.e., $z^\star = +\infty$ and $v^\star = -infty$)

(ii) (P) is infeasible and (D) is unbounded (i.e., $z^\star = v^\star = -infty$)

(iii) (P) is unbounded and (D) is infeasible (i.e., $z^\star == v^\star = -infty$)

(iv) (P) and (D) both have finite optimal values with $p^\star = v^\star$ and optimal solutions $(\bar{x}, \bar{u})$ that attain those values.

Also, if (P) is a convex problem and satisfies the Slater condition (saying that $\exists x_0 : g(x_0) < 0$) then, again, it has a strong dual. In the homework, we will see an example of a (convex) problem in which strong duality fails.

14.8 Perturbation and sensitivity analysis

Consider the following perturbed version of the original problem:

$$\begin{array}{lll} (\mathrm{P}_\lambda) & \inf & f(x) \\ & \text{s.t.} & g(x) \leq \lambda \\ & & x \in X, \end{array}$$

where $\lambda \in \mathbb{R}^m$. The problem coincides with the original problem when $\lambda = 0$. We define the *perturbation function* $z(\lambda)$ as the optimal value of the perturbed problem. Notice that $z(0) = z^\star$.

When the problem (P) is convex, then the function $z(\lambda)$ is a convex function of λ (the proof is left as a homework exercise). Moreover, the dual variables provide important information about how rapidly $z(\lambda)$ changes:

Theorem 75 *Suppose that strong duality for (P) holds and that the dual optimum is attained. If $\bar{u}$ is the optimal solution of the dual of the unperturbed problem, then for any λ*

$$z(\lambda) \geq z(0) - \bar{u}^T \lambda.$$

Proof: Suppose that x is any feasible point of the perturbed problem, i.e., $g(x) \leq \lambda$. Then, by strong duality,

$$z(0) = L^\star(\bar{u}) \leq f(x) + \bar{u}^T g(x) \leq f(x) + \bar{u}^T \lambda.$$

We conclude that for any x feasible for the perturbed problem,

$$f(x) \geq z(0) - \bar{u}^T \lambda,$$

from which the conclusion of the theorem follows. ∎

If can furthermore be shown that, if the conditions of the above theorem hold and $z(\lambda)$ is differentiable at $\lambda = 0$, then $-\bar{u} = \nabla z(0)$.

14.9 Duality strategies

14.9.1 Dualizing "bad" constraints

Suppose we wish to solve:

$$\begin{array}{lllll} \text{(P)} & \min_x & c^T x & & \\ & \text{s.t.} & Ax & \leq & b \\ & & Nx & \leq & g. \end{array}$$

Suppose that optimization over the constraints "$Nx \leq g$" is easy, but that the addition of the constraints "$Ax \leq b$" makes the problem much more difficult. This can happen, for example, when the constraints "$Nx \leq g$" are network constraints, and when "$Ax \leq b$" are non-network constraints in the model.

Let

$$X = \{x \mid Nx \leq g\}$$

and re-write (P) as:

$$\begin{array}{lllll} \text{(P)}: & \min_x & c^T x & & \\ & \text{s.t.} & Ax & \leq & b \\ & & x & \in & X. \end{array}$$

The Lagrangian is:

$$L(x,u) = c^T x + u^T(Ax - b) = -u^T b + (c^T + u^T A)x,$$

and the dual function is:

$$\begin{array}{lll} L^\star(u) := & \min_x & -u^T b + (c^T + u^T A)x \\ & \text{s.t.} & x \in X. \end{array}$$

Notice that $L^\star(u)$ is easy to evaluate for any value of u, and so we can attempt to solve (P) by designing an algorithm to solve the dual problem:

$$\begin{array}{lll} \text{(D)}: & \text{maximum}_u & L^\star(u) \\ & \text{s.t.} & u \geq 0. \end{array}$$

14.9.2 Dualizing a large problem into many small problems

Suppose we wish to solve:

$$\begin{array}{llllll} \text{(P)} & \min_{x^1,x^2} & (c^1)^T x^1 & +(c^2)^T x^2 & & \\ & \text{s.t.} & B^1 x^1 & +B^2 x^2 & \leq & d \\ & & A^1 x^1 & & \leq & b^1 \\ & & & A^2 x^2 & \leq & b^2 \end{array}$$

Notice here that if it were not for the constraints "$B^1x^1 + B^2x^2 \leq d$", that we would be able to separate the problem into two separate problems. Let us dualize on these constraints. Let:

$$X = \left\{(x^1, x^2) \mid A^1x^1 \leq b^1,\ A^2x^2 \leq b^2\right\}$$

and re-write (P) as:

$$\begin{array}{lccccc} \text{(P)} & \min_{x^1,x^2} & (c^1)^T x^1 & +(c^2)^T x^2 & & \\ & \text{s.t.} & B^1 x^1 & +B^2 x^2 & \leq & d \\ & & & (x^1, x^2) & \in & X \end{array}$$

The Lagrangian is:

$$\begin{array}{rcl} L(x,u) & = & (c^1)^T x^1 + (c^2)^T x^2 + u^T (B^1 x^1 + B^2 x^2 - d) \\ & = & -u^T d + ((c^1)^T + u^T B^1) x^1 + ((c^2)^T + u^T B^2) x^2, \end{array}$$

and the dual function is:

$$\begin{array}{rccc} L^\star(u) = & \min_{x^1,x^2} & -u^T d + ((c^1)^T + u^T B^1) x^1 + ((c^2)^T + u^T B^2) x^2 \\ & \text{s.t.} & (x^1, x^2) \in X \end{array}$$

which can be re-written as:

$$\begin{array}{rcl} L^\star(u) = & - & u^T d \\ & + & \min\limits_{A^1 x^1 \leq b^1} ((c^1)^T + u^T B^1) x^1 \\ & + & \min\limits_{A^2 x^2 \leq b^2} ((c^2)^T + u^T B^2) x^2 \end{array}$$

Notice once again that $L^\star(u)$ is easy to evaluate for any value of u, and so we can attempt to solve P by designing an algorithm to solve the dual problem:

$$\begin{array}{lcc} \text{(D)} & \text{maximum}_u & L^\star(u) \\ & \text{s.t.} & u \geq 0 \end{array}$$

14.10 A slight detour: subgradient optimization

14.10.1 Review: separating hyperplane theorems

Recall that in Theorem 27 we established that, if S be a nonempty closed convex set in $\mathbb{R}^n$, and $y \notin S$, then $\exists p \neq 0$ and α such that $H = \{x : p^T x = \alpha\}$ strongly separates S and $\{y\}$.

This result can be extended to prove the following two theorems:

Theorem 76 (BSS 2.4.7) *If S is a nonempty convex set in $\mathbb{R}^n$, and let $\bar{x}$ be contained in the boundary of S. Then there exists a hyperplane that supports S at $\bar{x}$, i.e., there exists $p \in \mathbb{R}^n$ such that $p^T(x - \bar{x}) \leq 0$ for each x in the closure of S.*

Theorem 77 (BSS 2.4.8) *Suppose S_1 and S_2 are two nonempty convex sets in $\mathbb{R}^m$, and suppose they do not intersect. Then there exists a hyperplane that separates S_1 and S_2, i.e., there exists $p \in \mathbb{R}^n$ such that*

$$\inf_{x \in S_1} p^T x \geq \sup_{x \in S_2} p^T x.$$

14.10.2 Subgradients of convex functions

Let $S \in \mathbb{R}^n$ and $f : S \to \mathbb{R}$ be given. The *epigraph* of f, denoted by epi$f$, is a set defined as

$$\text{epi}f = \{(x, \alpha) \in \mathbb{R}^{n+1} : x \in S,\ \alpha \in \mathbb{R},\ f(x) \leq \alpha\}$$

Theorem 78 *Let S be a nonempty convex set. Then $f : S \to \mathbb{R}$ is a convex function if and only if epi$f$ is a convex set.*

Proof: Left as an exercise. ∎

Suppose that $f(x)$ is a convex function. If $f(x)$ is differentiable, we have the gradient inequality:

$$f(x) \geq f(\bar{x}) + \nabla f(\bar{x})^T (x - \bar{x}) \text{ for any } x \in S\ ,$$

where typically we think of $S = \mathbb{R}^n$. There are many important convex functions that are not differentiable. The notion of the gradient generalizes to the concept of a *subgradient* of a convex function. A vector $g \in \mathbb{R}^n$ is called a *subgradient* of the convex function $f(x)$ at $x = \bar{x}$ if the following *subgradient inequality* is satisfied:

$$f(x) \geq f(\bar{x}) + g^T (x - \bar{x}) \text{ for all } x \in S.$$

Theorem 79 *Supose $f : S \to \mathbb{R}$ is a convex function. If $\bar{x} \in intS$, then there exists a vector g such that the hyperplane*

$$H = \{(x, y) \in \mathbb{R}^{n+1} : y = f(\bar{x}) + g^T (x - \bar{x})\}$$

supports epi$f$ at $(\bar{x}, f(\bar{x}))$. In particular, g is a subgradient of f at $\bar{x}$.

Proof: Because epi$f$ is a convex set, and $(\bar{x}, f(\bar{x}))$ belongs to the boundary of epi$f$, here exists a supporting hyperplane to epi$f$ at $(\bar{x}, f(\bar{x}))$. Thus, there exists a nonzero vector $(g, u) \in \mathbb{R}^{n+1}$ such that

$$g^T x + u\alpha \leq g^T \bar{x} + f(\bar{x})u \text{ for all } (x, \alpha) \in \text{epi} f.$$

Let α be any scalar larger than $f(x)$. Then as we make α arbitrarily large, the inequality must still hold. Thus $u \leq 0$. If $u < 0$, we can re-scale such that $u = -1$. Then $g^T x - \alpha \leq g^T \bar{x} - f(\bar{x})$, which upon rearranging terms yields:

$$\alpha \geq f(\bar{x}) + g^T(x - \bar{x}) \text{ for all } (x, \alpha) \in \text{epi} f.$$

In particular, with $\alpha = f(x)$, $f(x) \geq f(\bar{x}) + g^T(x - \bar{x})$, proving the theorem.

It remains to show that $u = 0$ is impossible. If $u = 0$, then $g^T x \leq g^T \bar{x}$ for all $x \in S$. But since $\bar{x} \in \text{int} S$, we have $\bar{x} + \delta g \in S$ for $\delta > 0$ sufficiently small. Thus, $g^T(\bar{x} + \delta g) \leq g^T \bar{x}$, i.e., $\delta g^T g \leq 0$. But this is a contradiction, since $\delta > 0$ and $g \neq 0$, since $(g, u) = (g, 0) \neq 0$. ∎

For each x, let $\partial f(x)$ denote the set of all subgradients of $f(x)$ at x. We call $\partial f(x)$ the "subdifferential of $f(x)$." We write $g \in \partial f(x)$ if g is a subgradient of f at x. If $f(x)$ is differentiable, then $\partial f(x) = \{\nabla f(x)\}$

Theorem 80 *Suppose $f : S \to \mathbb{R}$ is a function defined on a convex set S. Suppose that for each $\bar{x} \in intS$ there exists a subgradient vector g. Then f is a convex function on $intS$.*

Theorem 81 *Let f be convex of $\mathbb{R}^n$, let S be a convex set, and consider the following optimization problem:*

$$\min_{x \in S} f(x)$$

Then $\bar{x}$ is a global minimizer if and only if there exists $g \in \partial f(\bar{x})$ such that $g^T(x - \bar{x})$ for any $x \in S$.

Proof: The "if" part follows easily by the subgradient inequality.

Conversely, suppose $\bar{x}$ is a global minimizer. Define the following sets:

$$A = \{(d, \alpha) \in \mathbb{R}^{n+1} : f(\bar{x} + d) < \alpha + f(\bar{x})\}$$

and

$$B = \{(d, \alpha) \in \mathbb{R}^{n+1} : \bar{x} + d \in S,\ \alpha \leq 0\}.$$

It is not hard to see that both A and B are convex sets, and $A \cap B = \emptyset$. (If not, $\bar{x}$ would not be a globally optimal solution.)

Therefore, A and B can be separated by a hyperplane $H = \{(d, \alpha) \in \mathbb{R}^{n+1} : g^T d + u\alpha = \beta\}$ where $(g, u) \neq 0$ and

- $f(\bar{x} + d) < \alpha + f(\bar{x}) \Rightarrow g^T d + u\alpha \leq \beta$
- $\bar{x} + d \in S,\ \alpha \leq 0 \Rightarrow g^T d + u\alpha \geq \beta$

In the first implication, α can be made arbitrarily large, which means $u \leq 0$. Also, setting $d = 0$ and $\alpha = \epsilon > 0$ implies that $\beta \geq \epsilon u$. In the second implication setting $d = 0$ and $\alpha = 0$ implies that $\beta \leq 0$. Thus, $\beta = 0$. In the second implication, setting $\alpha = 0$ we have $g^T d \geq 0$ whenever $\bar{x} + d \in S$, and so $g^T(\bar{x} + d - \bar{x}) \geq 0$ whenever $\bar{x} + d \in S$. Put another way, we have $x \in S$ implies that $g^T(x - \bar{x}) \geq 0$.

It only remains to show that g is a subgradient. Note that $u < 0$, for if $u = 0$, it would follow from the first implication that $g^T d \leq 0$ for any d, a contradiction. Since $u < 0$, we can re-scale so that $u = -1$.

Now let d be given so that $\bar{x} + d \in S$ and let $\alpha = f(\bar{x} + d) - f(\bar{x}) + \epsilon$ for some $\epsilon > 0$. Thus $f(\bar{x} + d) \geq f(\bar{x}) = g^T d$ for all $\bar{x} + d \in S$. Setting $x = \bar{x} + d$, we have that if $x \in S$,

$$f(x) \geq f(\bar{x}) + g^T(x - \bar{x}),$$

and so g is a subgradient of f at $\bar{x}$. ∎

14.10.3 Subgradient method for minimizing a convex function

Suppose that $f(x)$ is a convex function, and that we seek to solve:

$$\begin{array}{lll} \text{(P)} \ z^\star = & \min & f(x) \\ & \text{s.t.} & x \in \mathbb{R}^n. \end{array}$$

If $f(x)$ is differentiable and $d := -\nabla f(\bar{x})$ satisfies $d \neq 0$, then d is an *descent direction* at $\bar{x}$, namely

$$f(\bar{x} + \epsilon d) < f(\bar{x}) \ \text{ for all } \epsilon > 0 \ \text{ and sufficiently small.}$$

This is illustrated in Figure 1. However, if $f(x)$ is not differentiable and g is a subgradient of $f(x)$ at $x = \bar{x}$, then g is not necessarily an descent direction. This is illustrated in Figure 2.

The following algorithm generalizes the steepest descent algorithm and can be used to minimize a nondifferentiable convex function $f(x)$.

Subgradient method

Step 0: Initialization. Start with any point $x_1 \in \mathbb{R}^n$. Choose an infinite sequence of positive stepsize values $\{\lambda_k\}_{k=1}^\infty$. Set $k = 1$.

Step 1: Compute a subgradient. Compute $g \in \partial f(x_k)$. If $g = 0$, stop; x_k solves (P).

Step 2: Compute stepsize. Compute stepsize λ_k from stepsize series.

Step 3: Update Iterate. Set $x_{k+1} \leftarrow x_k - \lambda_k \frac{g}{\|g\|}$. Set $k \leftarrow k + 1$ and go to **Step 1**.

Note in this algorithm that the step-size λ_k at each iteration is determined without a line-search, and in fact is predetermined in Step 0. One reason for this is that a line-search might not be worthwhile, since $-g$ is not necessarily a descent direction for a non-differentiable function.

As it turns out, the viability of the subgradient method depends critically on the sequence of step-sizes:

Theorem 82 *Suppose that f is a convex function whose domain $D \subseteq \mathbb{R}^n$ satisfies* intD $\neq \emptyset$. *Suppose that $\{\lambda_k\}_{k=1}^\infty$ satisfies:*

$$\lim_{k \to \infty} \lambda_k = 0 \ \textit{and} \ \sum_{k=1}^{\infty} \lambda_k = \infty.$$

Let $\{x_k\}_{k=1}^\infty$ be the iterates generated by the subgradient method. Then

$$\inf_k f(x_k) = z^\star.$$

Proof: Suppose the result is not true. Then there exists $\epsilon > 0$ such that $f(x_k) \geq z^\star + \epsilon$ for all k. Let

$$T = \{x \in \mathbb{R}^n : f(x) \leq z^\star + \epsilon\}.$$

Then there exist $\hat{x}$ and $\rho > 0$ for which $B(\hat{x}, \rho) \subset T$. Let g_k be the subgradient chosen by the subgradient method at the iterate x_k. By the subgradient inequality we have for all k:

$$f(x_k) \geq z^\star + \epsilon \geq f\left(\hat{x} + \rho\frac{g_k}{\|g_k\|}\right) \geq f(x_k) + g_k^T\left(\hat{x} + \rho\frac{g_k}{\|g_k\|} - x_k\right),$$

which upon rearranging yields:

$$g_k^T(\hat{x} - x_k) \leq -\rho\frac{g_k^T g_k}{\|g_k\|} = -\rho\|g_k\|.$$

We also have, for each k,

$$\begin{aligned}
\|x_{k+1} - \hat{x}\|^2 &= \left\|x_k - \lambda_k\frac{g_k}{\|g_k\|} - \hat{x}\right\|^2 \\
&= \|x_k - \hat{x}\|^2 + \lambda_k^2 + 2\lambda_k\frac{g_k^T(\hat{x} - x_k)}{\|g_k\|} \\
&\leq \|x_k - \hat{x}\|^2 + \lambda_k^2 - 2\lambda_k\rho \\
&= \|x_k - \hat{x}\|^2 + \lambda_k(\lambda_k - 2\rho).
\end{aligned}$$

For k sufficiently large, say, $k \geq K$, we have $\lambda_k \leq \rho$, whereby:

$$\|x_{k+1} - \hat{x}\|^2 \leq \|x_k - \hat{x}\|^2 - \rho\lambda_k.$$

However, this implies by induction that for all $j \geq 1$ we have:

$$\|x_{K+j} - \hat{x}\|^2 \leq \|x_K - \hat{x}\|^2 - \rho\sum_{k=K+1}^{K+j}\lambda_k.$$

Now for j sufficiently large, the right hand side expression is negative, since $\sum_{k=1}^{\infty}\lambda_k = \infty$, which yields a contradiction, since the left hand side must be nonnegative. ∎

14.10.4 Subgradient method with projections

Problem (P) stated in the beginning of this subsection generalizes to the following (constrained) problem:

$$(\mathrm{P}_S)\ z^\star = \begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in S, \end{array}$$

where S is a closed convex set. We assume that S is a simple enough set that we can easily compute projections onto S. I.e., for any point $c \in \mathbb{R}^n$, we can easily compute

$$\Pi_S(c) = \mathrm{argmin}_{x \in S}\,\|c - x\|.$$

The following algorithm is a simple extension of the subgradient method for unconstrained minimization, but includes a projection computation so that all iterate values x_k satisfy $x_k \in S$.

Subgradient method with projections

Step 0: Initialization. Start with any point $x_1 \in S$. Choose an infinite sequence of positive stepsize values $\{\lambda_k\}_{k=1}^{\infty}$. Set $k = 1$.

Step 1: Compute a subgradient. Compute $g \in \partial f(x_k)$. If $g = 0$, stop; x_k solves (P).

Step 2: Compute stepsize. Compute stepsize λ_k from stepsize series.

Step 3: Update Iterate. Set $x_{k+1} \leftarrow \Pi_S\left(x_k - \lambda_k \frac{g}{\|g\|}\right)$. Set $k \leftarrow k+1$ and go to **Step 1**.

Similarly to Theorem 82, we have

Theorem 83 *Suppose that f is a convex function whose domain $D \subseteq \mathbb{R}^n$ satisfies* intD $\cap$ S $\neq \emptyset$. *Suppose that $\{\lambda_k\}_{k=1}^{\infty}$ satisfies:*

$$\lim_{k\to\infty} \lambda_k = 0 \text{ and } \sum_{k=1}^{\infty} \lambda_k = \infty.$$

Let $\{x_k\}_{k=1}^{\infty}$ be the iterates generated by the subgradient method. Then

$$\inf_k f(x_k) = z^\star.$$

The proof of Theorem 83 relies on the following "non-expansive" property of the projection operator Π_S:

Lemma 84 *Let S be a closed convex set and let Π_S be the projection operator onto S. Then for any two vectors c_1 and c_2 in $\mathbb{R}^n$,*

$$\|\Pi_S(c_1) - \Pi_S(c_2)\| \leq \|c_1 - c_2\|.$$

Proof: Let $b_1 = \Pi_S(c_1)$ and $b_2 = \Pi_S(c_2)$. Then from Theorem 28 we have:

$$(c_1 - b_1)^T(x - b_1) \leq 0 \ \forall x \in C, \text{ and } (c_2 - b_2)^T(x - b_2) \leq 0 \ \forall x \in C.$$

In particular, because $b_1, b_2 \in C$, it follows that

$$(c_1 - b_1)^T(b_2 - b_1) \leq 0, \text{ and } (c_2 - b_2)^T(b_1 - b_2) \leq 0.$$

Then note that

$$\begin{aligned}
\|c_1 - c_2\|^2 &= \|b_1 - b_2 + (c_1 - b_1 - c_2 + b_2)\|^2 \\
&= \|b_1 - b_2\|^2 + \|c_1 - b_1 - c_2 + b_2\|^2 + 2(b_1 - b_2)^T(c_1 - b_1 - c_2 + b_2) \\
&\geq \|b_1 - b_2\|^2 + 2(b_1 - b_2)^T(c_1 - b_1) + 2(b_1 - b_2)^T(-c_2 + b_2) \\
&\geq \|b_1 - b_2\|^2,
\end{aligned}$$

which proves the lemma. ∎

The proof of Theorem 83 can easily be constructed using Lemma 84 and by folowing the logic used in the proof of Theorem 82.

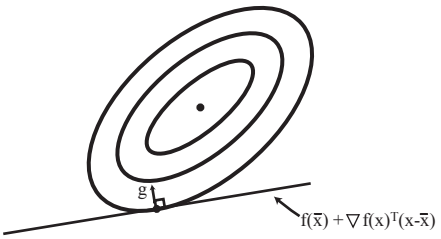


Figure 1: Contours of a **concave** functions. The gradient is an ascent direction.

14.11 Solution of the Lagrangian dual via subgradient optimization

We start with the primal problem:

$$\begin{array}{lll} \text{(P)} & \min_x & f(x) \\ & \text{s.t.} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & & x \in X. \end{array}$$

We create the Lagrangian:

$$L(x, u) := f(x) + u^T g(x)$$

and the dual function:

$$L^\star(u) := \min_{x \in X} f(x) + u^T g(x)$$

The dual problem then is:

$$\begin{array}{lll} \text{(D)} & \max_u & L^\star(u) \\ & \text{s.t.} & u \geq 0 \end{array}$$

Recall that $L^\star(u)$ is a concave function. For concave functions, we work with *supergradients*. If f is a concave function whose domain is a convex set S, then g is a supergradient of f at $\bar{x} \in S$ if it is a subgradient of $-f$ at $\bar{x}$, or if

$$f(x) \leq f(\bar{x}) + g^T(x - \bar{x}) \text{ for all } x \in S.$$

Here is an illustration of a situation in which a supergradient of a concave function is not necessarily an ascent direction: see figures 1–2.

The premise of Lagrangian duality is that it is "easy" to compute $L^\star(\bar{u})$ for any given $\bar{u}$. That is, it is easy to compute an optimal solution $\bar{x} \in X$ of

$$L^\star(\bar{u}) = \min_{x \in X} f(x) + \bar{u}^T g(x) = f(\bar{x}) + \bar{u}^T g(\bar{x})$$

for any given $\bar{u}$. It turns out that computing subgradients of $L^\star(u)$ is then also easy. We have:

Proposition 85 *Suppose that $\bar{u}$ is given and that $\bar{x} \in X$ is an optimal solution of $L^\star(\bar{u}) = \min_{x \in X} f(x) + \bar{u}^T g(x)$. Then $g := g(\bar{x})$ is a subgradient of $L^\star(u)$ at $u = \bar{u}$.*

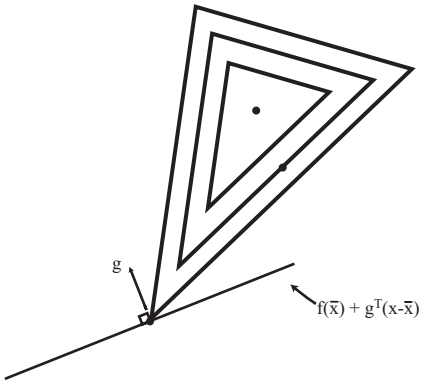


Figure 2: Contours of a **concave** functions. A supergradient is not necessarily an ascent direction.

Proof: For any $u \geq 0$ we have

$$\begin{aligned} L^\star(u) &= \min_{x \in X} f(x) + u^T g(x) \\ &\leq f(\bar{x}) + u^T g(\bar{x}) \\ &= f(\bar{x}) + \bar{u}^T g(\bar{x}) + (u - \bar{u})^T g(\bar{x}) \\ &= \min_{x \in X} f(x) + \bar{u}^T g(x) + g(\bar{x})^T (u - \bar{u}) \\ &= L^\star(\bar{u}) + g^T (u - \bar{u}). \end{aligned}$$

Therefore g is a subgradient of $L^\star(u)$ at $\bar{u}$. ∎

The Lagrangian dual problem (D) is in the same format as problem (P_S), with $S = \mathbb{R}^m_+$. In order to apply the projected subgradient method to this problem, we need to be able to conveniently compute the projection of any vectore $v \in \mathbb{R}^m$ onto $S = \mathbb{R}^m_+$. This indeed is easy: if v^+ is defined as the vector whose components are the positive parts of respective components of v, then it is easy to see that $\Pi_{\mathbb{R}^m_+}(v) = v^+$.

The subgradient method for solving the Lagrangian dual can now be stated:

Step 0: Initialization. Start with any point $u_1 \in \mathbb{R}^m_+$, $u^1 \geq 0$. Choose an infinite sequence of positive stepsize values $\{\lambda_k\}_{k=1}^\infty$. Set $k = 1$.

Step 1: Compute a subgradient. Solve for an optimal solution $\bar{x}$ of $L^\star(u_k) = \min_{x \in X} f(x) + u_k^T g(x)$. Set $g := g(\bar{x})$.

Step 2: Compute stepsize. Compute stepsize λ_k from stepsize series.

Step 3: Update Iterate. Set $u_{k+1} \leftarrow u_k + \lambda_k \frac{g}{\|g\|}$. If $u_{k+1} \not\geq 0$, re-set $u_{k+1} \leftarrow u^+_{k+1}$. Set $k \leftarrow k+1$ and go to **Step 1**.

15 Primal-dual interior point methods for linear programming

15.1 The problem

The logarithmic barrier approach to solving a linear program dates back to the work of Fiacco and McCormick in 1967 in their book *Sequential Unconstrained Minimization Techniques*, also known simply as SUMT. The method was not believed then to be either practically or theoretically interesting, when in fact today it is both! The method was re-born as a consequence of Karmarkar's interior-point method, and has been the subject of an enormous amount of research and computation, even to this day. In these notes we present the basic algorithm and a basic analysis of its performance.

Consider the linear programming problem in standard form:

$$\begin{array}{lll} \text{(P)} & \min & c^T x \\ & \text{s.t.} & Ax = b \\ & & x \geq 0, \end{array}$$

where x is a vector of n variables, whose standard linear programming dual problem is:

$$\begin{array}{lll} D: & \max & b^T \pi \\ & \text{s.t.} & A^T \pi + s = c \\ & & s \geq 0. \end{array}$$

Given a feasible solution x of P and a feasible solution (π, s) of D, the duality gap is simply

$$c^T x - b^T \pi = x^T s \geq 0.$$

We introduce the following notation which will be very convenient for manipulating equations, etc. Suppose that $x > 0$. Define the matrix X to be the $n \times n$ diagonal matrix whose diagonal entries are precisely the components of x. Then X looks like:

$$\begin{pmatrix} x_1 & 0 & \dots & 0 \\ 0 & x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n \end{pmatrix}.$$

Notice that X is positive definite, and so is X^2, which looks like:

$$\begin{pmatrix} x_1^2 & 0 & \dots & 0 \\ 0 & x_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n^2 \end{pmatrix}.$$

Similarly, the matrices X^{-1} and X^{-2} look like:

$$\begin{pmatrix} 1/x_1 & 0 & \dots & 0 \\ 0 & 1/x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1/x_n \end{pmatrix}$$

and

$$\begin{pmatrix} 1/x_1^2 & 0 & \dots & 0 \\ 0 & 1/x_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1/x_n^2 \end{pmatrix}.$$

Let us introduce a logarithmic barrier term for (P). We obtain:

$$\begin{array}{lll} P(\theta) & \min & c^T x - \theta \sum_{j=1}^{n} \ln(x_j) \\ & \text{s.t.} & Ax = b \\ & & x > 0. \end{array}$$

Because the gradient of the objective function of $P(\theta)$ is simply $c - \theta X^{-1} e$, (where e is the vector of ones, i.e., $e = (1, 1, 1, ..., 1)^T$), the Karush-Kuhn-Tucker conditions for $P(\theta)$ are:

$$\begin{aligned} Ax = b,\ x > 0 \\ c - \theta X^{-1} e = A^T \pi. \end{aligned} \tag{34}$$

If we define $s = \theta X^{-1} e$, then

$$\frac{1}{\theta} X s = e,$$

equivalently

$$\frac{1}{\theta} X S e = e,$$

and we can rewrite the Karush-Kuhn-Tucker conditions as:

$$\begin{aligned} Ax = b,\ x > 0 \\ A^T \pi + s = c \\ \frac{1}{\theta} X S e - e = 0. \end{aligned} \tag{35}$$

From the equations of (35) it follows that if (x, π, s) is a solution of (35), then x is feasible for P, (π, s) is feasible for D, and the resulting duality gap is:

$$x^T s = e^T X S e = \theta e^T e = \theta n.$$

This suggests that we try solving $P(\theta)$ for a variety of values of θ as $\theta \to 0$.

However, we cannot usually solve (35) exactly, because the third equation group is not linear in the variables. We will instead define a "β-approximate solution" of the Karush-Kuhn-Tucker conditions (35). A β-approximate solution of $P(\theta)$ is defined as any solution (x, π, s) of

$$\begin{aligned} Ax = b,\ x > 0 \\ A^T \pi + s = c \\ \left\| \frac{1}{\theta} X s - e \right\| \le \beta. \end{aligned} \tag{36}$$

Here the norm $\| \cdot \|$ is the Euclidean norm.

Lemma 86 *If $(\bar{x}, \bar{\pi}, \bar{s})$ is a β-approximate solution of $P(\theta)$ and $\beta < 1$, then $\bar{x}$ is feasible for P, $(\bar{\pi}, \bar{s})$ is feasible for D, and the duality gap satisfies:*

$$n\theta(1-\beta) \leq c^T\bar{x} - b^T\bar{\pi} = \bar{x}^T\bar{s} \leq n\theta(1+\beta). \tag{37}$$

Proof: Primal feasibility is obvious. To prove dual feasibility, we need to show that $\bar{s} \geq 0$. To see this, note that the third equation system of (36) implies that

$$-\beta \leq \frac{\bar{x}_j\bar{s}_j}{\theta} - 1 \leq \beta$$

which we can rearrange as:

$$\theta(1-\beta) \leq \bar{x}_j\bar{s}_j \leq \theta(1+\beta). \tag{38}$$

Therefore $\bar{x}_j\bar{s}_j \geq (1-\beta)\theta > 0$, which implies $\bar{x}_j\bar{s}_j > 0$, and so $\bar{s}_j > 0$. From (38) we have

$$n\theta(1-\beta) = \sum_{j=1}^{n}\theta(1-\beta) \leq \sum_{j=1}^{n}\bar{x}_j\bar{s}_j = \bar{x}^T\bar{s} \leq \sum_{j=1}^{n}\theta(1+\beta) = n\theta(1+\beta).$$

∎

15.2 The primal-dual algorithm

Based on the analysis just presented, we are motivated to develop the following algorithm:

Step 0: Initialization Data is $(x^0, \pi^0, s^0, \theta^0)$. $k = 0$. Assume that (x^0, π^0, s^0) is a β-approximate solution of $P(\theta^0)$ for some known value of β that satisfies $\beta < \frac{1}{2}$.

Step 1: Set current values $(\bar{x}, \bar{\pi}, \bar{s}) = (x^k, \pi^k, s^k)$, $\theta = \theta^k$.

Step 2: Shrink θ. Set $\theta' = \alpha\theta$ for some $\alpha \in (0, 1)$.

Step 3: Compute the primal-dual Newton direction. Compute the Newton step $(\Delta x, \Delta\pi, \Delta s)$ for the equation system (35) at $(x, \pi, s) = (\bar{x}, \bar{\pi}, \bar{s})$ for θ', by solving the following system of equations in the variables $(\Delta x, \Delta\pi, \Delta s)$:

$$\begin{aligned} A\Delta x &= 0 \\ A^T\Delta\pi + \Delta s &= 0 \\ \bar{S}\Delta x + \bar{X}\Delta s &= \bar{X}\bar{S}e - \theta' e. \end{aligned} \tag{39}$$

Denote the solution to this system by $(\Delta x, \Delta\pi, \Delta s)$.

Step 4: Update All Values.

$$(x', \pi', s') = (\bar{x}, \bar{\pi}, \bar{s}) + (\Delta x, \Delta\pi, \Delta s)$$

Step 5: Reset Counter and Continue. $(x^{k+1}, \pi^{k+1}, s^{k+1}) = (x', \pi', s')$. $\theta^{k+1} = \theta'$. $k \leftarrow k+1$. Go to Step 1.

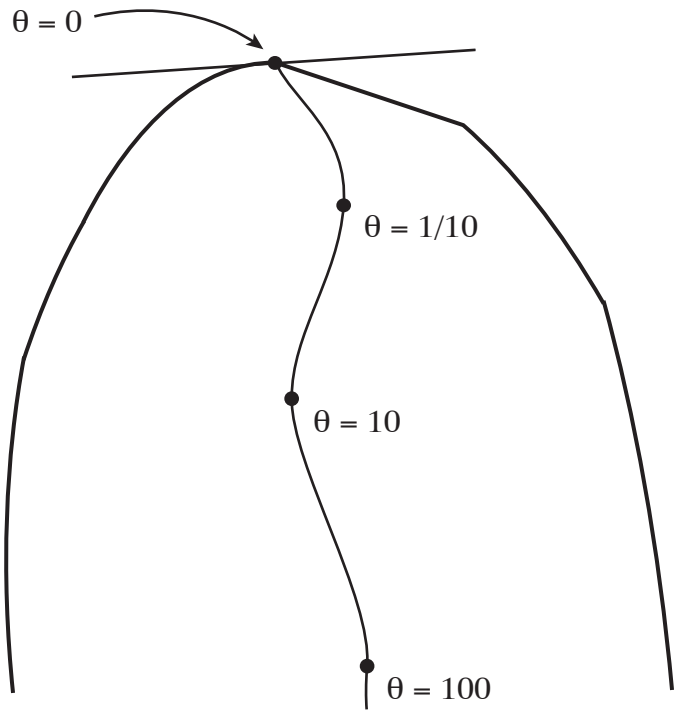


Figure 3: A conceptual picture of the interior-point algorithm.

Figure 3 shows a picture of the algorithm.

Some of the issues regarding this algorithm include:

- how to set the approximation constant β and the fractional decrease parameter α. We will see that it will be convenient to set $\beta = \frac{3}{40}$ and

$$\alpha = 1 - \frac{\frac{1}{8}}{\frac{1}{5} + \sqrt{n}}.$$

- the derivation of the primal-dual Newton equation system (39)
- whether or not successive iterative values (x^k, π^k, s^k) are β-approximate solutions to $P(\theta^k)$
- how to get the method started

15.3 The primal-dual Newton step

Recall that we introduced a logarithmic barrier term for P to obtain $P(\theta)$:

$$\begin{array}{rlll} P(\theta): & \min_x & c^T x - \theta \sum_{j=1}^{n} \ln(x_j) \\ & \text{s.t.} & Ax = b \\ & & x > 0, \end{array}$$

the Karush-Kuhn-Tucker conditions for which are:

$$\begin{aligned} Ax = b,\ x > 0 \\ c - \theta X^{-1} e = A^T \pi. \end{aligned} \tag{40}$$

We defined $s = \theta X^{-1} e$, and rewrote the Karush-Kuhn-Tucker conditions as:

$$\begin{aligned} Ax = b, x > 0 \\ A^T \pi + s = c \\ \frac{1}{\theta} XSe - e = 0. \end{aligned} \tag{41}$$

Let $(\bar{x}, \bar{\pi}, \bar{s})$ be our current iterate, which we assume is primal and dual feasible, namely:

$$A\bar{x} = b,\ \bar{x} > 0, \quad A^T \bar{\pi} + \bar{s} = c, \bar{s} > 0. \tag{42}$$

Introducing a direction $(\Delta x, \Delta \pi, \Delta s)$, the next iterate will be $(\bar{x}, \bar{\pi}, \bar{s}) + (\Delta x, \Delta \pi, \Delta s)$, and we want to solve:

$$\begin{aligned} A(\bar{x} + \Delta x) = b,\ \bar{x} + \Delta x > 0 \\ A^T(\bar{\pi} + \Delta \pi) + (\bar{s} + \Delta s) = c \\ \frac{1}{\theta}(\bar{X} + \Delta X)(\bar{S} + \Delta S)e - e = 0. \end{aligned}$$

Keeping in mind that $(\bar{x}, \bar{\pi}, \bar{s})$ is primal-dual feasible and so satisfies (42), we can rearrange the above to be:

$$\begin{aligned} A\Delta x = 0 \\ A^T \Delta \pi + \Delta s = 0 \\ \bar{S}\Delta x + \bar{X}\Delta s = \theta e - \bar{X}\bar{S}e - \Delta X \Delta S e. \end{aligned}$$

Notice that the only nonlinear term in the above system of equations in $(\Delta x, \Delta \pi, \Delta s)$ is term the term " $\Delta X \Delta S e$" in the last system. If we erase this term, which is the same as the linearized version of the equations, we obtain the following primal-dual Newton equation system:

$$\begin{aligned} A\Delta x = 0 \\ A^T \Delta \pi + \Delta s = 0 \\ \bar{S}\Delta x + \bar{X}\Delta s = \theta e - \bar{X}\bar{S}e. \end{aligned} \tag{43}$$

The solution $(\Delta x, \Delta \pi, \Delta s)$ of the system (43) is called the primal-dual Newton step. We can manipulate these equations to yield the following formulas for the solution:

$$\begin{aligned} \Delta x &\leftarrow \left[I - \bar{X}\bar{S}^{-1} A^T \left(A\bar{X}\bar{S}^{-1} A^T \right)^{-1} A \right] \left(-\bar{x} + \theta \bar{S}^{-1} e \right), \\ \Delta \pi &\leftarrow \left(A\bar{X}\bar{S}^{-1} A^T \right)^{-1} A \left(\bar{x} - \theta \bar{S}^{-1} e \right), \\ \Delta s &\leftarrow A^T \left(A\bar{X}\bar{S}^{-1} A^T \right)^{-1} A \left(-\bar{x} + \theta \bar{S}^{-1} e \right). \end{aligned} \tag{44}$$

Notice, by the way, that the computational effort in these equations lies primarily in solving a single equation:

$$\left(A\bar{X}\bar{S}^{-1} A^T \right) \Delta \pi = A \left(\bar{x} - \theta \bar{S}^{-1} e \right).$$

Once this system is solved, we can easily substitute:

$$\begin{aligned} \Delta s &\leftarrow -A^T \Delta \pi \\ \Delta x &\leftarrow -\bar{x} + \theta \bar{S}^{-1} e - \bar{S}^{-1} \bar{X} \Delta s. \end{aligned} \tag{45}$$

However, let us instead simply work with the primal-dual Newton system (43). Suppose that $(\Delta x, \Delta\pi, \Delta s)$ is the (unique) solution of the primal-dual Newton system (43). We obtain the new value of the variables (x, π, s) by taking the Newton step:

$$(x', \pi', s') = (\bar{x}, \bar{\pi}, \bar{s}) + (\Delta x, \Delta\pi, \Delta s).$$

We have the following very powerful convergence theorem which demonstrates the quadratic convergence of Newton's method for this problem, with an explicit guarantee of the range in which quadratic convergence takes place.

Theorem 87 (Explicit Quadratic Convergence of Newton's Method) *Suppose that $(\bar{x}, \bar{\pi}, \bar{s})$ is a β-approximate solution of $P(\theta)$ and $\beta < \frac{1}{3}$. Let $(\Delta x, \Delta\pi, \Delta s)$ be the solution to the primal-dual Newton equations (43), and let:*

$$(x', \pi', s') = (\bar{x}, \bar{\pi}, \bar{s}) + (\Delta x, \Delta\pi, \Delta s).$$

Then (x', π', s') is a $\left(\frac{1+\beta}{(1-\beta)^2}\right)\beta^2$-approximate solution of $P(\theta)$.

Proof: Our current point $(\bar{x}, \bar{\pi}, \bar{s})$ satisfies:

$$\begin{aligned} A\bar{x} = b, \bar{x} > 0 \\ A^T\bar{\pi} + \bar{s} = c \\ \left\| \frac{1}{\theta}\bar{X}\bar{S}e - e \right\| \leq \beta. \end{aligned} \tag{46}$$

Furthermore the primal-dual Newton step $(\Delta x, \Delta\pi, \Delta s)$ satisfies:

$$\begin{aligned} A\Delta x = 0 \\ A^T\Delta\pi + \Delta s = 0 \\ \bar{S}\Delta x + \bar{X}\Delta s = \theta e - \bar{X}\bar{S}e. \end{aligned} \tag{47}$$

Note from the first two equations of (47) that $\Delta x^T \Delta s = 0$. From the third equation of (46) we have

$$1 - \beta \leq \frac{\bar{s}_j \bar{x}_j}{\theta} \leq 1 + \beta, \ j = 1, \ldots, n, \tag{48}$$

which implies:

$$\bar{x}_j \geq \frac{(1-\beta)\theta}{\bar{s}_j} \text{ and } \bar{s}_j \geq \frac{(1-\beta)\theta}{\bar{x}_j}, \ j = 1, \ldots, n. \tag{49}$$

As a result of this we obtain:

$$\begin{aligned} \theta(1-\beta)\|\bar{X}^{-1}\Delta x\|^2 &= \theta(1-\beta)\Delta x^T \bar{X}^{-1}\bar{X}^{-1}\Delta x \\ &\leq \Delta x^T \bar{X}^{-1}\bar{S}\Delta x \\ &= \Delta x^T \bar{X}^{-1}\left(\theta e - \bar{X}\bar{S}e - \bar{X}\Delta s\right) \\ &= \Delta x^T \bar{X}^{-1}\left(\theta e - \bar{X}\bar{S}e\right) \\ &\leq \|\bar{X}^{-1}\Delta x\|\|\theta e - \bar{X}\bar{S}e\| \\ &\leq \|\bar{X}^{-1}\Delta x\|\beta\theta. \end{aligned}$$

From this it follows that

$$\|\bar{X}^{-1}\Delta x\| \leq \frac{\beta}{1-\beta} < 1.$$

Therefore

$$x' = \bar{x} + \Delta x = \bar{X}(e + \bar{X}^{-1}\Delta x) > 0.$$

We have the exact same chain of inequalities for the dual variables:

$$\begin{aligned}
\theta(1-\beta)\|\bar{S}^{-1}\Delta s\|^2 &= \theta(1-\beta)\Delta s^T \bar{S}^{-1}\bar{S}^{-1}\Delta s \\
&\leq \Delta s^T \bar{S}^{-1}\bar{X}\Delta s \\
&= \Delta s^T \bar{S}^{-1}\left(\theta e - \bar{X}\bar{S}e - \bar{S}\Delta x\right) \\
&= \Delta s^T \bar{S}^{-1}\left(\theta e - \bar{X}\bar{S}e\right) \\
&\leq \|\bar{S}^{-1}\Delta s\|\|\theta e - \bar{X}\bar{S}e\| \\
&\leq \|\bar{S}^{-1}\Delta s\|\beta\theta.
\end{aligned}$$

From this it follows that

$$\|\bar{S}^{-1}\Delta s\| \leq \frac{\beta}{1-\beta} < 1.$$

Therefore

$$s' = \bar{s} + \Delta s = \bar{S}(e + \bar{S}^{-1}\Delta s) > 0.$$

Next note from (47) that for $j = 1, \ldots, n$ we have:

$$x'_j s'_j = (\bar{x}_j + \Delta x_j)(\bar{s}_j + \Delta s_j) = \bar{x}_j\bar{s}_j + \bar{x}_j\Delta s_j + \Delta x_j\bar{s}_j + \Delta x_j\Delta s_j = \theta + \Delta x_j\Delta s_j.$$

Therefore

$$\left(e - \frac{1}{\theta}X'S'e\right)_j = -\frac{\Delta x_j \Delta s_j}{\theta}.$$

From this we obtain:

$$\begin{aligned}
\left\|e - \frac{1}{\theta}X'S'e\right\| &\leq \left\|e - \frac{1}{\theta}X'S'e\right\|_1 \\
&= \sum_{j=1}^{n} \frac{|\Delta x_j \Delta s_j|}{\theta} \\
&= \sum_{j=1}^{n} \frac{|\Delta x_j|}{\bar{x}_j}\frac{|\Delta s_j|}{\bar{s}_j}\frac{\bar{x}_j\bar{s}_j}{\theta} \\
&\leq \sum_{j=1}^{n} \frac{|\Delta x_j|}{\bar{x}_j}\frac{|\Delta s_j|}{\bar{s}_j}(1+\beta) \\
&\leq \|\bar{X}^{-1}\Delta x\|\|\bar{S}^{-1}\Delta s\|(1+\beta) \\
&\leq \left(\frac{\beta}{1-\beta}\right)^2(1+\beta).
\end{aligned}$$

■

15.4 Complexity analysis of the algorithm

Theorem 88 (Relaxation Theorem) *Suppose that* $(\bar{x}, \bar{\pi}, \bar{s})$ *is a* $\beta = \frac{3}{40}$*-approximate solution of* $P(\theta)$*. Let*

$$\alpha = 1 - \frac{\frac{1}{8}}{\frac{1}{5} + \sqrt{n}}$$

and let $\theta' = \alpha\theta$. Then $(\bar{x}, \bar{\pi}, \bar{s})$ is a $\beta = \frac{1}{5}$-approximate solution of $P(\theta')$.

Proof: The triplet $(\bar{x}, \bar{\pi}, \bar{s})$ satisfies $A\bar{x} = b, \bar{x} > 0$, and $A^T\bar{\pi} + \bar{s} = c$, and so it remains to show that

$$\left\| \frac{1}{\theta'} \bar{X}\bar{s} - e \right\| \leq \frac{1}{5}.$$

We have

$$\begin{aligned} \left\| \frac{1}{\theta'} \bar{X}\bar{s} - e \right\| = \left\| \frac{1}{\alpha\theta} \bar{X}\bar{s} - e \right\| &= \left\| \frac{1}{\alpha} \left(\frac{1}{\theta} \bar{X}\bar{s} - e \right) - \left(1 - \frac{1}{\alpha} \right) e \right\| \\ &\leq \left(\frac{1}{\alpha} \right) \left\| \frac{1}{\theta} \bar{X}\bar{s} - e \right\| + \left| \frac{1-\alpha}{\alpha} \right| \|e\| \\ \leq \frac{\frac{3}{40}}{\alpha} + \left(\frac{1-\alpha}{\alpha} \right) \sqrt{n} \quad &= \quad \frac{\frac{3}{40} + \sqrt{n}}{\alpha} - \sqrt{n} = \frac{1}{5}. \end{aligned}$$

■

Theorem 89 (Convergence Theorem). *Suppose that (x^0, π^0, s^0) is a $\beta = \frac{3}{40}$-approximate solution of $P(\theta^0)$. Then for all $k = 1, 2, 3, \ldots$, (x^k, π^k, s^k) is a $\beta = \frac{3}{40}$-approximate solution of $P(\theta^k)$.*

Proof: By induction, suppose that the theorem is true for iterates $0, 1, 2, ..., k$.

Then (x^k, π^k, s^k) is a $\beta = \frac{3}{40}$-approximate solution of $P(\theta^k)$. From the Relaxation Theorem, (x^k, π^k, s^k) is a $\frac{1}{5}$-approximate solution of $P(\theta^{k+1})$ where $\theta^{k+1} = \alpha\theta^k$.

From the Quadratic Convergence Theorem, $(x^{k+1}, \pi^{k+1}, s^{k+1})$ is a β-approximate solution of $P(\theta^{k+1})$ for

$$\beta = \frac{1 + \frac{1}{5}}{\left(1 - \frac{1}{5}\right)^2} \left(\frac{1}{5} \right)^2 = \frac{3}{40}.$$

Therefore, by induction, the theorem is true for all values of k. ■

Figure 4 shows a better picture of the algorithm.

Theorem 90 (Complexity Theorem) *Suppose that (x^0, π^0, s^0) is a $\beta = \frac{3}{40}$-approximate solution of $P(\theta^0)$. In order to obtain primal and dual feasible solutions (x^k, π^k, s^k) with a duality gap of at most ϵ, one needs to run the algorithm for at most*

$$k = \left\lceil 10\sqrt{n} \ln \left(\frac{43}{37} \frac{(x^0)^T s^0}{\epsilon} \right) \right\rceil$$

iterations.

Proof: Let k be as defined above. Note that

$$\alpha = 1 - \frac{\frac{1}{8}}{\frac{1}{5} + \sqrt{n}} = 1 - \frac{1}{\left(\frac{8}{5} + 8\sqrt{n}\right)} \leq 1 - \frac{1}{10\sqrt{n}}.$$

Therefore

$$\theta^k \leq \left(1 - \frac{1}{10\sqrt{n}} \right)^k \theta^0.$$

This implies that

$$c^T x^k - b^T \pi^k = (x^k)^T s^k \leq \theta^k n(1+\beta) \leq \left(1 - \frac{1}{10\sqrt{n}} \right)^k \left(\frac{43}{40} n \right) \theta^0$$

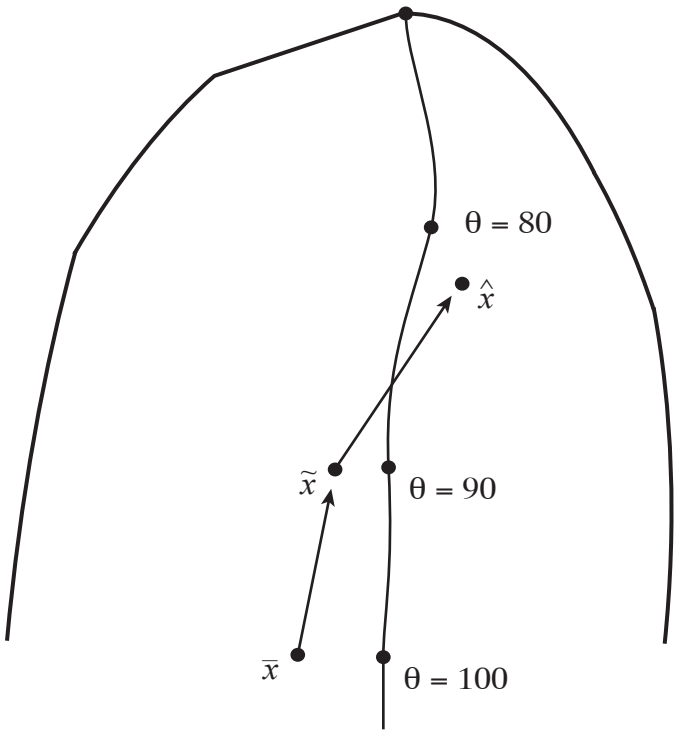


Figure 4: Another picture of the interior-point algorithm.

$$\leq \left(1 - \frac{1}{10\sqrt{n}}\right)^k \left(\frac{43}{40}n\right) \left(\frac{(x^0)^T s^0}{\frac{37}{40}n}\right),$$

from (37). Taking logarithms, we obtain

$$\ln(c^T x^k - b^T \pi^k) \leq k \ln\left(1 - \frac{1}{10\sqrt{n}}\right) + \ln\left(\frac{43}{37}(x^0)^T s^0\right)$$

$$\leq \frac{-k}{10\sqrt{n}} + \ln\left(\frac{43}{37}(x^0)^T s^0\right)$$

$$\leq -\ln\left(\frac{43}{37}\frac{(x^0)^T s^0}{\epsilon}\right) + \ln\left(\frac{43}{37}(x^0)^T s^0\right) = \ln(\epsilon).$$

The second inequality uses the fact that $\ln(1-t) \leq -t$ for all $t < 1$. Therefore $c^T x^k - b^T \pi^k \leq \epsilon$. ∎

15.5 An implementable primal-dual interior-point algorithm

Herein we describe a more implementable primal-dual interior-point algorithm. This algorithm differs from the previous method in the following respects:

- We do not assume that the current point is near the central path. In fact, we do not assume that the current point is even feasible.

- The fractional decrease parameter α is set to $\alpha = \frac{1}{10}$ rather than the conservative value of

$$\alpha = 1 - \frac{\frac{1}{8}}{\frac{1}{5} + \sqrt{n}}.$$

- We do not necessarily take the full Newton step at each iteration, and we take different step-sizes in the primal and dual.

Our current "point" is $(\bar{x}, \bar{\pi}, \bar{s})$ for which $\bar{x} > 0$ and $\bar{s} > 0$, but quite possibly $A\bar{x} \neq b$ and/or $A^T\bar{\pi} + \bar{s} \neq c$. We are given a value of the central path barrier parameter $\theta > 0$. We want to compute a direction $(\Delta x, \Delta\pi, \Delta s)$ so that $(\bar{x} + \Delta x, \bar{\pi} + \Delta\pi, \bar{s} + \Delta s)$ approximately solves the central path equations. We set up the system:

$$\begin{array}{ll} (1) & A(\bar{x} + \Delta x) = b \\ (2) & A^T(\bar{\pi} + \Delta\pi) + (\bar{s} + \Delta s) = c \\ (3) & (\bar{X} + \Delta X)(\bar{S} + \Delta S)e = \theta e. \end{array}$$

We linearize this system of equations and rearrange the terms to obtain the Newton equations for the current point $(\bar{x}, \bar{\pi}, \bar{s})$:

$$\begin{array}{ll} (1) & A\Delta x = b - A\bar{x} =: r_1 \\ (2) & A^T\Delta\pi + \Delta s = c - A^T\bar{\pi} - \bar{s} =: r_2 \\ (3) & \bar{S}\Delta x + \bar{X}\Delta s = \theta e - \bar{X}\bar{S}e =: r_3 \end{array}$$

We refer to the solution $(\Delta x, \Delta\pi, \Delta s)$ to the above system as the primal-dual Newton direction at the point $(\bar{x}, \bar{\pi}, \bar{s})$. It differs from that derived earlier only in that earlier it was assumed that $r_1 = 0$ and $r_2 = 0$.

Given our current point $(\bar{x}, \bar{\pi}, \bar{s})$ and a given value of θ, we compute the Newton direction $(\Delta x, \Delta\pi, \Delta s)$ and we update our variables by choosing primal and dual step-sizes α_P and α_D to obtain new values:

$$(\tilde{x}, \tilde{\pi}, \tilde{s}) \leftarrow (\bar{x} + \alpha_P\Delta x, \bar{\pi} + \alpha_D\Delta\pi, \bar{s} + \alpha_D\Delta s).$$

In order to ensure that $\tilde{x} > 0$ and $\tilde{s} > 0$, we choose a value of r satisfying $0 < r < 1$ ($r = 0.99$ is a common value in practice), and determine α_P and α_D as follows:

$$\alpha_P = \min\left\{1,\ r \min_{\Delta x_j < 0}\left\{\frac{\bar{x}_j}{-\Delta x_j}\right\}\right\}$$

$$\alpha_D = \min\left\{1,\ r \min_{\Delta s_j < 0}\left\{\frac{\bar{s}_j}{-\Delta s_j}\right\}\right\}.$$

These step-sizes ensure that the next iterate $(\tilde{x}, \tilde{\pi}, \tilde{s})$ satisfies $\tilde{x} > 0$ and $\tilde{s} > 0$.

15.5.1 Decreasing the Path Parameter θ

We also want to shrink θ at each iteration, in order to (hopefully) shrink the duality gap. The current iterate is $(\bar{x}, \bar{\pi}, \bar{s})$, and the current values satisfy:

$$\theta \approx \frac{\bar{x}^T \bar{s}}{n}$$

We then re-set θ to

$$\theta \leftarrow \left(\frac{1}{10}\right)\left(\frac{\bar{x}^T \bar{s}}{n}\right),$$

where the fractional decrease $\frac{1}{10}$ is user-specified.

15.5.2 The Stopping Criterion

We typically declare the problem solved when it is "almost" primal feasible, "almost" dual feasible, and there is "almost" no duality gap. We set our tolerance ϵ to be a small positive number, typically $\epsilon = 10^{-8}$, for example, and we stop when:

$$\begin{array}{ll} (1) & \|A\bar{x} - b\| \leq \epsilon \\ (2) & \|A^T\bar{\pi} + \bar{s} - c\| \leq \epsilon \\ (3) & \bar{s}^T\bar{x} \leq \epsilon \end{array}$$

15.5.3 The Full Interior-Point Algorithm

1. Given (x^0, π^0, s^0) satisfying $x^0 > 0$, $s^0 > 0$, and $\theta^0 > 0$, and r satisfying $0 < r < 1$, and $\epsilon > 0$. Set $k \leftarrow 0$.

2. Test stopping criterion. Check if:

$$\begin{array}{ll} (1) & \|Ax^k - b\| \leq \epsilon \\ (2) & \|A^T\pi^k + s^k - c\| \leq \epsilon \\ (3) & (s^k)^T x^k \leq \epsilon. \end{array}$$

   If so, STOP. If not, proceed.

3. Set $\theta \leftarrow \left(\frac{1}{10}\right)\left(\frac{(x^k)^T(s^k)}{n}\right)$

4. Solve the Newton equation system:

$$\begin{array}{ll} (1) & A\Delta x = b - Ax^k =: r_1 \\ (2) & A^T\Delta\pi + \Delta s = c - A^T\pi^k - s^k =: r_2 \\ (3) & S^k\Delta x + X^k\Delta s = \theta e - X^k S^k e =: r_3 \end{array}$$

5. Determine the step-sizes:

$$\begin{aligned} \theta_P &= \min\left\{1,\ r \min_{\Delta x_j < 0}\left\{\frac{x_j^k}{-\Delta x_j}\right\}\right\} \\ \theta_D &= \min\left\{1,\ r \min_{\Delta s_j < 0}\left\{\frac{s_j^k}{-\Delta s_j}\right\}\right\}. \end{aligned}$$

6. Update values:

$$(x^{k+1}, \pi^{k+1}, s^{k+1}) \quad \leftarrow \quad (x^k + \alpha_P \Delta x, \pi^k + \alpha_D \Delta \pi, s^k + \alpha_D \Delta s).$$

Re-set $k \leftarrow k + 1$ and return to (b).

15.5.4 Remarks on interior-point methods

- The algorithm just described is almost exactly what is used in commercial interior-point method software.
- A typical interior-point code will solve a linear or quadratic optimization problem in 25-80 iterations, regardless of the dimension of the problem.
- These days, interior-point methods have been extended to allow for the solution of a very large class of convex nonlinear optimization problems.

16 Introduction to Semidefinite Programming (SDP)

16.1 Introduction

Semidefinite programming (SDP) is probably the most exciting development in mathematical programming in the last ten years. SDP has applications in such diverse fields as traditional convex constrained optimization, control theory, and combinatorial optimization. Because SDP is solvable via interior-point methods (and usually requires about the same amount of computational resources as linear optimization), most of these applications can usually be solved fairly efficiently in practice as well as in theory.

16.2 A slightly different view of linear programming

Consider the linear programming problem in standard form:

$$\begin{array}{rlll} LP: & \text{minimize} & c \cdot x & \\ & \text{s.t.} & a_i \cdot x = b_i, & i = 1, \ldots, m \\ & & x \in \mathbb{R}^n_+. & \end{array}$$

Here x is a vector of n variables, and we write "$c \cdot x$" for the inner-product "$\sum_{j=1}^n c_j x_j$", etc.

Also, $\mathbb{R}^n_+ := \{x \in \mathbb{R}^n \mid x \geq 0\}$, and we call $\mathbb{R}^n_+$ the nonnegative orthant. In fact, $\mathbb{R}^n_+$ is a *closed convex cone*, where K is called a closed a convex cone if K satisfies the following two conditions:

- If $x, w \in K$, then $\alpha x + \beta w \in K$ for all nonnegative scalars α and β.
- K is a closed set.

In words, LP is the following problem:

"Minimize the linear function $c \cdot x$, subject to the condition that x must solve m given equations $a_i \cdot x = b_i,\ i = 1, \ldots, m$, and that x must lie in the closed convex cone $K = \mathbb{R}^n_+$."

We will write the standard linear programming dual problem as:

$$\begin{array}{rlll} LD: & \text{maximize} & \displaystyle\sum_{i=1}^m y_i b_i & \\ & \text{s.t.} & \displaystyle\sum_{i=1}^m y_i a_i + s = c & \\ & & s \in \mathbb{R}^n_+. & \end{array}$$

Given a feasible solution x of LP and a feasible solution (y, s) of LD, the duality gap is simply $c \cdot x - \sum_{i=1}^m y_i b_i = (c - \sum_{i=1}^m y_i a_i) \cdot x = s \cdot x \geq 0$, because $x \geq 0$ and $s \geq 0$. We know from LP duality theory that so long as the primal problem LP is feasible and has bounded optimal objective value, then the primal and the dual both attain their optima with no duality gap. That is, there exists x^* and (y^*, s^*) feasible for the primal and dual, respectively, for which $c \cdot x^* - \sum_{i=1}^m y_i^* b_i = s^* \cdot x^* = 0$.

16.3 Facts about matrices and the semidefinite cone

16.3.1 Facts about the semidefinite cone

If X is an $n \times n$ matrix, then X is a symmetric positive semidefinite (SPSD) matrix if $X = X^T$ and

$$v^T X v \geq 0 \text{ for any } v \in \mathbb{R}^n.$$

If X is an $n \times n$ matrix, then X is a symmetric positive definite (SPD) matrix if $X = X^T$ and

$$v^T X v > 0 \text{ for any } v \in \mathbb{R}^n, v \neq 0.$$

Let S^n denote the set of symmetric $n \times n$ matrices, and let S^n_+ denote the set of symmetric positive semidefinite (SPSD) $n \times n$ matrices. Similarly let S^n_{++} denote the set of symmetric positive definite (SPD) $n \times n$ matrices.

Let X and Y be any symmetric matrices. We write "$X \succeq 0$" to denote that X is SPSD, and we write "$X \succeq Y$" to denote that $X - Y \succeq 0$. We write "$X \succ 0$" to denote that X is SPD, etc.

$S^n_+ = \{X \in S^n \mid X \succeq 0\}$ is a closed convex cone in $\mathbb{R}^{n^2}$ of dimension $n \times (n+1)/2$.

To see why this remark is true, suppose that $X, W \in S^n_+$. Pick any scalars $\alpha, \beta \geq 0$. For any $v \in \mathbb{R}^n$, we have:

$$v^T(\alpha X + \beta W)v = \alpha v^T X v + \beta v^T W v \geq 0,$$

whereby $\alpha X + \beta W \in S^n_+$. This shows that S^n_+ is a cone. It is also straightforward to show that S^n_+ is a closed set.

16.3.2 Facts about eigenvalues and eigenvectors

If M is a square $n \times n$ matrix, then λ is an eigenvalue of M with corresponding eigenvector x if

$$Mx = \lambda x \text{ and } x \neq 0.$$

Note that λ is an eigenvalue of M if and only if λ is a root of the polynomial:

$$p(\lambda) := \det(M - \lambda I),$$

that is

$$p(\lambda) = \det(M - \lambda I) = 0.$$

This polynomial will have n roots counting multiplicities, that is, there exist $\lambda_1, \lambda_2, \ldots, \lambda_n$ for which:

$$p(\lambda) := \det(M - \lambda I) = \Pi_{i=1}^n (\lambda_i - \lambda).$$

If M is symmetric, then all eigenvalues λ of M must be real numbers, and these eigenvalues can be ordered so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ if we so choose.

The corresponding eigenvectores $q^1, \ldots, q^n$ of M can be chosen so that they are orthogonal, namely $\left(q^i\right)^T \left(q^j\right) = 0$ for $i \neq j$, and can be scaled so that $\left(q^i\right)^T \left(q^i\right) = 1$. This means that the matrix:

$$Q := \begin{bmatrix} q^1 & q^2 & \cdots & q^n \end{bmatrix}$$

satisfies:

$$Q^T Q = I,$$

or put another way:

$$Q^T = Q^{-1}.$$

We call such a matrix *orthonormal*.

Let us assemble the ordered eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ into a diagonal matrix D:

$$D := \begin{pmatrix} \lambda_1 & 0 & & 0 \\ 0 & \lambda_2 & & 0 \\ & & \ddots & \\ 0 & & & \lambda_n \end{pmatrix}.$$

Then we have:

Property: $M = QDQ^T$. To prove this, notice that $MQ = QD$, and so post-multiplying by Q^T yields: $M = MQQ^T = QDQ^T$.

The decomposition of M into $M = QDQ^T$ is called its *eigendecomposition*.

16.3.3 Facts about symmetric matrices

- If $X \in S^n$, then $X = QDQ^T$ for some orthonormal matrix Q and some diagonal matrix D. (Recall that Q is orthonormal means that $Q^{-1} = Q^T$, and that D is diagonal means that the off-diagonal entries of D are all zeros.)
- If $X = QDQ^T$ as above, then the columns of Q form a set of n orthogonal eigenvectors of X, whose eigenvalues are the corresponding entries of the diagonal matrix D.
- $X \succeq 0$ if and only if $X = QDQ^T$ where the eigenvalues (i.e., the diagonal entries of D) are all nonnegative.
- $X \succ 0$ if and only if $X = QDQ^T$ where the eigenvalues (i.e., the diagonal entries of D) are all positive.
- If M is symmetric, then $\det(M) = \Pi_{j=1}^n \lambda_j$.
- If $X \succeq 0$ then $X_{ii} \geq 0, i = 1, \ldots, n$.
- If $X \succeq 0$ and if $X_{ii} = 0$, then $X_{ij} = X_{ji} = 0$ for all $j = 1, \ldots, n$.
- Consider the matrix M defined as follows:

  $$M = \begin{pmatrix} P & v \\ v^T & d \end{pmatrix},$$

  where $P \succ 0$, v is a vector, and d is a scalar. Then $M \succeq 0$ if and only if $d - v^T P^{-1} v \geq 0$.
- For a given column vector a, the matrix $X := aa^T$ is SPSD, i.e., $X = aa^T \succeq 0$.

Also note the following:

- If $M \succeq 0$, then there is a matrix N for which $M = N^T N$. To see this, simply take $N = D^{\frac{1}{2}} Q^T$.
- If M is symmetric, then $\sum_{j=1}^n M_{jj} = \sum_{j=1}^n \lambda_j$

16.4 Semidefinite programming

Let $X \in S^n$. We can think of X as a matrix, or equivalently, as an array of n^2 components of the form $(x_{11}, \ldots, x_{nn})$. We can also just think of X as an object (a vector) in the space S^n. All three different equivalent ways of looking at X will be useful.

What will a linear function of X look like? If $C(X)$ is a linear function of X, then $C(X)$ can be written as $C \bullet X$, where

$$C \bullet X := \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}.$$

If X is a symmetric matrix, there is no loss of generality in assuming that the matrix C is also symmetric. With this notation, we are now ready to define a semidefinite program. A semidefinite program (SDP) is an optimization problem of the form:

$$\begin{array}{lll} SDP: & \text{minimize} & C \bullet X \\ & \text{s.t.} & A_i \bullet X = b_i,\ i = 1, \ldots, m, \\ & & X \succeq 0. \end{array}$$

Notice that in an SDP that the variable is the matrix X, but it might be helpful to think of X as an array of n^2 numbers or simply as a vector in S^n. The objective function is the linear function $C \bullet X$ and there are m linear equations that X must satisfy, namely $A_i \bullet X = b_i,\ i = 1, \ldots, m$. The variable X also must lie in the (closed convex) cone of positive semidefinite symmetric matrices S^n_+. Note that the data for SDP consists of the symmetric matrix C (which is the data for the objective function) and the m symmetric matrices $A_1, \ldots, A_m$, and the $m-$vector b, which form the m linear equations.

Let us see an example of an SDP for $n = 3$ and $m = 2$. Define the following matrices:

$$A_1 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 3 & 7 \\ 1 & 7 & 5 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 2 & 8 \\ 2 & 6 & 0 \\ 8 & 0 & 4 \end{pmatrix},\ b = \begin{pmatrix} 11 \\ 19 \end{pmatrix},\ \text{and } C = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 9 & 0 \\ 3 & 0 & 7 \end{pmatrix}.$$

Then the variable X will be the 3×3 symmetric matrix:

$$X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix},$$

and so, for example,

$$\begin{array}{rcl} C \bullet X & = & x_{11} + 2x_{12} + 3x_{13} + 2x_{21} + 9x_{22} + 0x_{23} + 3x_{31} + 0x_{32} + 7x_{33} \\ & = & x_{11} + 4x_{12} + 6x_{13} + 9x_{22} + 0x_{23} + 7x_{33}. \end{array}$$

since, in particular, X is symmetric. Therefore the SDP can be written as:

$$\begin{array}{lll} SDP: & \text{minimize} & x_{11} + 4x_{12} + 6x_{13} + 9x_{22} + 0x_{23} + 7x_{33} \\ & \text{s.t.} & x_{11} + 0x_{12} + 2x_{13} + 3x_{22} + 14x_{23} + 5x_{33} = 11 \\ & & 0x_{11} + 4x_{12} + 16x_{13} + 6x_{22} + 0x_{23} + 4x_{33} = 19 \\ & & X = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \succeq 0. \end{array}$$

Notice that SDP looks remarkably similar to a linear program. However, the standard LP constraint that x must lie in the nonnegative orthant is replaced by the constraint that the variable X must lie in the cone of positive semidefinite matrices. Just as "$x \geq 0$" states that each of the n components of x must be nonnegative, it may be helpful to think of "$X \succeq 0$" as stating that each of the n *eigenvalues* of X must be nonnegative. It is easy to see that a linear program LP is a special instance of an SDP. To see one way of doing this, suppose that $(c, a_1, \ldots, a_m, b_1, \ldots, b_m)$ comprise the data for LP. Then define:

$$A_i = \begin{pmatrix} a_{i1} & 0 & \ldots & 0 \\ 0 & a_{i2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{in} \end{pmatrix}, \quad i = 1, \ldots, m, \text{ and } C = \begin{pmatrix} c_1 & 0 & \ldots & 0 \\ 0 & c_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & c_n \end{pmatrix}.$$

Then LP can be written as:

$$\begin{array}{lll} SDP: & \text{minimize} & C \bullet X \\ & \text{s.t.} & A_i \bullet X = b_i, \ i = 1, \ldots, m, \\ & & X_{ij} = 0, \quad i = 1, \ldots, n, \quad j = i+1, \ldots, n, \\ & & X \succeq 0, \end{array}$$

with the association that

$$X = \begin{pmatrix} x_1 & 0 & \ldots & 0 \\ 0 & x_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & x_n \end{pmatrix}.$$

Of course, in practice one would never *want* to convert an instance of LP into an instance of SDP. The above construction merely shows that SDP includes linear programming as a special case.

16.5 Semidefinite programming duality

The dual problem of SDP is defined (or derived from first principles) to be:

$$\begin{array}{lll} SDD: & \text{maximize} & \sum_{i=1}^{m} y_i b_i \\ & \text{s.t.} & \sum_{i=1}^{m} y_i A_i + S = C \\ & & S \succeq 0. \end{array}$$

One convenient way of thinking about this problem is as follows. Given multipliers $y_1, \ldots, y_m$, the objective is to maximize the linear function $\sum_{i=1}^{m} y_i b_i$. The constraints of SDD state that the matrix S defined as

$$S = C - \sum_{i=1}^{m} y_i A_i$$

must be positive semidefinite. That is,

$$C - \sum_{i=1}^{m} y_i A_i \succeq 0.$$

We illustrate this construction with the example presented earlier. The dual problem is:

$$\begin{array}{lll} SDD: & \text{maximize} & 11y_1 + 19y_2 \\ & \text{s.t.} & y_1 \begin{pmatrix} 1 & 0 & 1 \\ 0 & 3 & 7 \\ 1 & 7 & 5 \end{pmatrix} + y_2 \begin{pmatrix} 0 & 2 & 8 \\ 2 & 6 & 0 \\ 8 & 0 & 4 \end{pmatrix} + S = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 9 & 0 \\ 3 & 0 & 7 \end{pmatrix} \\ & & S \succeq 0, \end{array}$$

which we can rewrite in the following form:

$$\begin{array}{lll} SDD: & \text{maximize} & 11y_1 + 19y_2 \\ & \text{s.t.} & \\ & & \begin{pmatrix} 1 - 1y_1 - 0y_2 & 2 - 0y_1 - 2y_2 & 3 - 1y_1 - 8y_2 \\ 2 - 0y_1 - 2y_2 & 9 - 3y_1 - 6y_2 & 0 - 7y_1 - 0y_2 \\ 3 - 1y_1 - 8y_2 & 0 - 7y_1 - 0y_2 & 7 - 5y_1 - 4y_2 \end{pmatrix} \succeq 0. \end{array}$$

It is often easier to "see" and to work with a semidefinite program when it is presented in the format of the dual SDD, since the variables are the m multipliers $y_1, \ldots, y_m$.

As in linear programming, we can switch from one format of SDP (primal or dual) to any other format with great ease, and there is no loss of generality in assuming a particular specific format for the primal or the dual.

The following proposition states that weak duality must hold for the primal and dual of SDP:

Proposition 91 *Given a feasible solution X of SDP and a feasible solution (y, S) of SDD, the duality gap is $C \bullet X - \sum_{i=1}^m y_i b_i = S \bullet X \geq 0$. If $C \bullet X - \sum_{i=1}^m y_i b_i = 0$, then X and (y, S) are each optimal solutions to SDP and SDD, respectively, and furthermore, $SX = 0$.*

In order to prove Proposition 91, it will be convenient to work with the *trace* of a matrix, defined below:

$$\text{trace}(M) = \sum_{j=1}^n M_{jj}.$$

Simple arithmetic can be used to establish the following two elementary identifies:

Property: $A \bullet B = \text{trace}(A^T B)$. To prove this, notice that $\text{trace}(A^T B) = \sum_{j=1}^n \left(A^T B\right)_{jj} = \sum_{j=1}^n \left(\sum_{i=1}^n A_{ij} B_{ij}\right) = A \bullet B$.

Property: $\text{trace}(MN) = \text{trace}(NM)$. To prove this, simply notice that $\text{trace}(MN) = M^T \bullet N = \sum_{i=1}^n \sum_{j=1}^n M_{ji} N_{ij} = \sum_{i=1}^n \sum_{j=1}^n N_{ij} M_{ji} = \sum_{i=1}^n \sum_{j=1}^n N_{ji} M_{ij} = N^T \bullet M = \text{trace}(NM)$.

Proof of Proposition 91. For the first part of the proposition, we must show that if $S \succeq 0$ and $X \succeq 0$, then $S \bullet X \geq 0$. Let $S = PDP^T$ and $X = QEQ^T$ where P, Q are orthonormal matrices and D, E are nonnegative diagonal matrices. We have:

$$S \bullet X = \text{trace}(S^T X) = \text{trace}(SX) = \text{trace}(PDP^T QEQ^T)$$

$$= \text{trace}(DP^T QEQ^T P) = \sum_{j=1}^n D_{jj} (P^T QEQ^T P)_{jj} \geq 0,$$

where the last inequality follows from the fact that all $D_{jj} \geq 0$ and the fact that the diagonal of the symmetric positive semidefinite matrix $P^T QEQ^T P$ must be nonnegative.

To prove the second part of the proposition, suppose that trace$(SX) = 0$. Then from the above equalities, we have

$$\sum_{j=1}^{n} D_{jj}(P^T QEQ^T P)_{jj} = 0.$$

However, this implies that for each $j = 1, \ldots, n$, either $D_{jj} = 0$ or the $(P^T QEQ^T P)_{jj} = 0$. Furthermore, the latter case implies that the j^{th} row of $P^T QEQ^T P$ is all zeros. Therefore $DP^T QEQ^T P = 0$, and so $SX = PDP^T QEQ^T = 0$. ∎

Unlike the case of linear programming, we cannot assert that either SDP or SDD will attain their respective optima, and/or that there will be no duality gap, unless certain regularity conditions hold. One such regularity condition which ensures that strong duality will prevail is a version of the "Slater condition," summarized in the following theorem which we will not prove:

Theorem 92 *Let z_P^* and z_D^* denote the optimal objective function values of SDP and SDD, respectively. Suppose that there exists a feasible solution $\hat{X}$ of SDP such that $\hat{X} \succ 0$, and that there exists a feasible solution $(\hat{y}, \hat{S})$ of SDD such that $\hat{S} \succ 0$. Then both SDP and SDD attain their optimal values, and $z_P^* = z_D^*$.*

16.6 Key properties of linear programming that do not extend to SDP

The following summarizes some of the more important properties of linear programming that do not extend to SDP:

- There may be a finite or infinite duality gap. The primal and/or dual may or may not attain their optima. However, as noted above in Theorem 92, both programs will attain their common optimal value if both programs have feasible solutions that are SPD.
- There is no finite algorithm for solving SDP. There is a simplex algorithm, but it is not a finite algorithm. There is no direct analog of a "basic feasible solution" for SDP.

16.7 SDP in combinatorial optimization

SDP has wide applicability in combinatorial optimization. A number of $NP-$hard combinatorial optimization problems have convex relaxations that are semidefinite programs. In many instances, the SDP relaxation is very tight in practice, and in certain instances in particular, the optimal solution to the SDP relaxation can be converted to a feasible solution for the original problem with provably good objective value. An example of the use of SDP in combinatorial optimization is given below.

16.7.1 An SDP relaxation of the MAX CUT problem

Let G be an undirected graph with nodes $N = \{1, \ldots, n\}$, and edge set E. Let $w_{ij} = w_{ji}$ be the weight on edge (i, j), for $(i, j) \in E$. We assume that $w_{ij} \geq 0$ for all $(i, j) \in E$. The MAX CUT

problem is to determine a subset S of the nodes N for which the sum of the weights of the edges that cross from S to its complement $\bar{S}$ is maximized (where $\bar{S} := N \setminus S$).

We can formulate MAX CUT as an integer program as follows. Let $x_j = 1$ for $j \in S$ and $x_j = -1$ for $j \in \bar{S}$. Then our formulation is:

$$\begin{array}{rlll} MAXCUT: & \text{maximize}_x & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(1 - x_i x_j) \\ & \text{s.t.} & x_j \in \{-1, 1\}, & j = 1, \ldots, n. \end{array}$$

Now let

$$Y = xx^T,$$

whereby

$$Y_{ij} = x_i x_j, \; i = 1, \ldots, n, \quad j = 1, \ldots, n.$$

Also let W be the matrix whose $(i,j)^{\text{th}}$ element is w_{ij} for $i = 1, \ldots, n$ and $j = 1, \ldots, n$. Then MAX CUT can be equivalently formulated as:

$$\begin{array}{rlll} MAXCUT: & \text{maximize}_{Y,x} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} - \frac{1}{4} W \bullet Y \\ & \text{s.t.} & x_j \in \{-1, 1\}, & j = 1, \ldots, n \\ & & Y = xx^T. \end{array}$$

Notice in this problem that the first set of constraints are equivalent to $Y_{jj} = 1, \; j = 1, \ldots, n$. We therefore obtain:

$$\begin{array}{rlll} MAXCUT: & \text{maximize}_{Y,x} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} - \frac{1}{4} W \bullet Y \\ & \text{s.t.} & Y_{jj} = 1, & j = 1, \ldots, n \\ & & Y = xx^T. \end{array}$$

Last of all, notice that the matrix $Y = xx^T$ is a symmetric rank-1 positive semidefinite matrix. If we *relax* this condition by removing the rank-1 restriction, we obtain the following relaxtion of MAX CUT, which is a semidefinite program:

$$\begin{array}{rlll} RELAX: & \text{maximize}_{Y} & \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} - \frac{1}{4} W \bullet Y \\ & \text{s.t.} & Y_{jj} = 1, & j = 1, \ldots, n \\ & & Y \succeq 0. \end{array}$$

It is therefore easy to see that RELAX provides an upper bound on MAXCUT, i.e.,

$$MAXCUT \leq RELAX.$$

As it turns out, one can also prove without too much effort that:

$$0.87856 \; RELAX \leq MAXCUT \leq RELAX.$$

This is an impressive result, in that it states that the value of the semidefinite relaxation is guaranteed to be no more than 12.2% higher than the value of NP-hard problem MAX CUT.

16.8 SDP in convex optimization

As stated above, SDP has very wide applications in convex optimization. The types of constraints that can be modelled in the SDP framework include: linear inequalities, convex quadratic inequalities, lower bounds on matrix norms, lower bounds on determinants of SPSD matrices, lower bounds on the geometric mean of a nonnegative vector, plus many others. Using these and other constructions, the following problems (among many others) can be cast in the form of a semidefinite program: linear programming, optimizing a convex quadratic form subject to convex quadratic inequality constraints, minimizing the volume of an ellipsoid that covers a given set of points and ellipsoids, maximizing the volume of an ellipsoid that is contained in a given polytope, plus a variety of maximum eigenvalue and minimum eigenvalue problems. In the subsections below we demonstrate how some important problems in convex optimization can be re-formulated as instances of SDP.

16.8.1 SDP for convex quadratically constrained quadratic programming

A convex quadratically constrained quadratic program is a problem of the form:

$$\begin{array}{lll} QCQP: & \underset{x}{\text{minimize}} & x^T Q_0 x + q_0^T x + c_0 \\ & \text{s.t.} & x^T Q_i x + q_i^T x + c_i \leq 0,\ i = 1, \ldots, m, \end{array}$$

where the $Q_0 \succeq 0$ and $Q_i \succeq 0, \quad i = 1, \ldots, m$. This problem is the same as:

$$\begin{array}{lll} QCQP: & \underset{x,\theta}{\text{minimize}} & \theta \\ & \text{s.t.} & x^T Q_0 x + q_0^T x + c_0 - \theta \leq 0 \\ & & x^T Q_i x + q_i^T x + c_i \leq 0,\ i = 1, \ldots, m. \end{array}$$

We can factor each Q_i into

$$Q_i = M_i^T M_i$$

for some matrix M_i. Then note the equivalence:

$$\begin{pmatrix} I & M_i x \\ x^T M_i^T & -c_i - q_i^T x \end{pmatrix} \succeq 0 \quad \Longleftrightarrow \quad x^T Q_i x + q_i^T x + c_i \leq 0.$$

In this way we can write $QCQP$ as:

$$\begin{array}{lll} QCQP: & \underset{x,\theta}{\text{minimize}} & \theta \\ & \text{s.t.} & \\ & & \begin{pmatrix} I & M_0 x \\ x^T M_0^T & -c_0 - q_0^T x + \theta \end{pmatrix} \succeq 0 \\ & & \begin{pmatrix} I & M_i x \\ x^T M_i^T & -c_i - q_i^T x \end{pmatrix} \succeq 0,\ i = 1, \ldots, m. \end{array}$$

Notice in the above formulation that the variables are θ and x and that all matrix coefficients are linear functions of θ and x.

16.8.2 SDP for second-order cone optimization

A second-order cone optimization problem (SOCP) is an optimization problem of the form:

$$\begin{array}{lll} \text{SOCP:} & \min_x & c^T x \\ & \text{s.t.} & Ax = b \\ & & \|Q_i x + d_i\| \leq \left(g_i^T x + h_i\right) \ , \quad i = 1, \ldots, k. \end{array}$$

In this problem, the norm $\|v\|$ is the standard Euclidean norm:

$$\|v\| := \sqrt{v^T v}.$$

The norm constraints in SOCP are called "second-order cone" constraints. Note that these are convex constraints.

Here we show that any second-order cone constraint can be written as an SDP constraint. Indeed we have:

Property:

$$\|Qx + d\| \leq \left(g^T x + h\right) \iff \begin{pmatrix} (g^T x + h)I & (Qx + d) \\ (Qx + d)^T & g^T x + h \end{pmatrix} \succeq 0.$$

Note in the above that the matrix involved here is a linear function of the variable x, and so is in the general form of an SDP constraint. This property is a direct consequence of the fact (stated earlier) that

$$M = \begin{pmatrix} P & v \\ v^T & d \end{pmatrix} \succeq 0 \iff d - v^T P^{-1} v \geq 0.$$

Therefore we can write the second-order cone optimization problem as:

$$\begin{array}{lll} \text{SDPSOCP:} & \min_x & c^T x \\ & \text{s.t.} & Ax = b \\ & & \begin{pmatrix} (g_i^T x + h_i)I & (Q_i x + d_i) \\ (Q_i x + d_i)^T & g_i^T x + h_i \end{pmatrix} \succeq 0 \ , \quad i = 1, \ldots, k. \end{array}$$

16.8.3 SDP for eigenvalue optimization

There are many types of eigenvalue optimization problems that can be formulated as SDPs. In a typical eigenvalue optimization problem, we are given symmetric matrices B and $A_i,\ i = 1, \ldots, k$, and we choose weights $w_1, \ldots, w_k$ to create a new matrix S:

$$S := B - \sum_{i=1}^{k} w_i A_i.$$

In some applications there might be restrictions on the weights w, such as $w \geq 0$ or more generally linear inequalities of the form $Gw \leq d$. The typical goal is then to choose w in such a way that the eigenvalues of S are "well-aligned," for example:

- $\lambda_{\min}(S)$ is maximized
- $\lambda_{\max}(S)$ is minimized

- $\lambda_{\max}(S) - \lambda_{\min}(S)$ is minimized
- $\sum_{j=1}^{n} \lambda_j(S)$ is minimized or maximized

Let us see how to work with these problems using SDP. First, we have:

Property: $M \succeq tI$ if and only if $\lambda_{\min}(M) \geq t$.

To see why this is true, let us consider the eigenvalue decomposition of $M = QDQ^T$, and consider the matrix R defined as:

$$R = M - tI = QDQ^T - tI = Q(D - tI)Q^T.$$

Then

$$M \succeq tI \iff R \succeq 0 \iff D - tI \succeq 0 \iff \lambda_{\min}(M) \geq t.$$

Property: $M \preceq tI$ if and only if $\lambda_{\max}(M) \leq t$.

To see why this is true, let us consider the eigenvalue decomposition of $M = QDQ^T$, and consider the matrix R defined as:

$$R = M - tI = QDQ^T - tI = Q(D - tI)Q^T.$$

Then

$$M \preceq tI \iff R \preceq 0 \iff D - tI \preceq 0 \iff \lambda_{\max}(M) \leq t.$$

Now suppose that we wish to find weights w to minimize the difference between the largest and the smallest eigenvalues of S. This problem can be written down as:

$$\begin{array}{rll} EOP: & \underset{w,S}{\text{minimize}} & \lambda_{\max}(S) - \lambda_{\min}(S) \\ & \text{s.t.} & S = B - \sum_{i=1}^{k} w_i A_i \\ & & Gw \leq d. \end{array}$$

Then EOP can be written as:

$$\begin{array}{rll} EOP: & \underset{w,S,\mu,\lambda}{\text{minimize}} & \mu - \lambda \\ & \text{s.t.} & S = B - \sum_{i=1}^{k} w_i A_i \\ & & Gw \leq d \\ & & \lambda I \preceq S \preceq \mu I. \end{array}$$

This last problem is a semidefinite program.

Using constructs such as those shown above, very many other types of eigenvalue optimization problems can be formulated as SDPs. For example, suppose that we would like to work with $\sum_{j=1}^{n} \lambda_j(S)$. Then one can use elementary properties of the determinant function to prove:

Property: If M is symmetric, then $\sum_{j=1}^{n} \lambda_j(S) = \sum_{j=1}^{n} M_{jj}$.

Then we can work with $\sum_{j=1}^{n} \lambda_j(S)$ by using instead $I \bullet S$. Therefore enforcing a constraint that the sum of the eigenvalues must lie between l and u can be written as:

$$\begin{array}{rlll} EOP2: & \underset{w,S,\mu,\lambda}{\text{minimize}} & \mu - \lambda & \\ & \text{s.t.} & S = B - \sum_{i=1}^{k} w_i A_i & \\ & & Gw \leq d & \\ & & \lambda I \preceq S \preceq \mu I & \\ & & l \leq I \bullet S \leq u. & \end{array}$$

This last problem is a semidefinite program.

16.8.4 The logarithmic barrier function

At the heart of an interior-point method is a barrier function that exerts a repelling force from the boundary of the feasible region. For SDP, we need a barrier function whose values approach $+\infty$ as points X approach the boundary of the semidefinite cone S_+^n.

Let $X \in S_+^n$. Then X will have n eigenvalues, say $\lambda_1(X), \ldots, \lambda_n(X)$ (possibly counting multiplicities). We can characterize the boundary of the semidefinite cone as follows:

$$\partial S_+^n = \{X \in S^n \mid \lambda_j(X) \geq 0, j = 1, \ldots, n, \text{ and } \lambda_j(X) = 0 \text{ for some } j \in \{1, \ldots, n\}\}.$$

A natural barrier function to use to repel X from the boundary of S_+^n then is

$$B(X) := -\sum_{j=1}^{n} \ln(\lambda_i(X)) = -\ln(\prod_{j=1}^{n} \lambda_i(X)) = -\ln(\det(X)).$$

This function is called the log-determinant function or the logarithmic barrier function for the semidefinite cone. It is not too difficult to derive the gradient and the Hessian of $B(X)$ and to construct the following quadratic Taylor expansion of $B(X)$:

$$B(\bar{X} + \alpha S) \approx B(\bar{X}) + \alpha \bar{X}^{-1} \bullet S + \frac{1}{2}\alpha^2 \left(\bar{X}^{-\frac{1}{2}} S \bar{X}^{-\frac{1}{2}}\right) \bullet \left(\bar{X}^{-\frac{1}{2}} S \bar{X}^{-\frac{1}{2}}\right).$$

The barrier function $B(X)$ has the same remarkable properties in the context of interior-point methods for SDP as the barrier function $-\sum_{j=1}^{n} \ln(x_j)$ does in the context of linear optimization.

16.8.5 The analytic center problem for SDP

Just as in linear optimization, we can consider the analytic center problem for SDP. Given a system of the form:

$$\sum_{i=1}^{m} y_i A_i \preceq C,$$

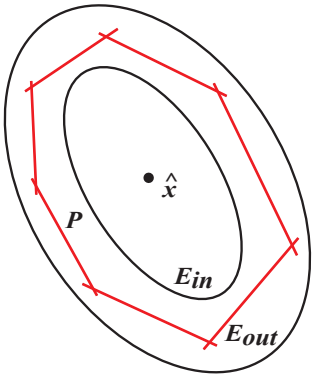


Figure 5: Illustration of the ellipsoid construction at the analytic center.

the *analytic center* is the solution $(\hat{y}, \hat{S})$ of the following optimization problem:

$$\begin{array}{lll} \text{(ACP:)} & \text{maximize}_{y,S} & \prod_{i=1}^{n} \lambda_i(S) \\ & \text{s.t.} & \sum_{i=1}^{m} y_i A_i + S = C \\ & & S \succeq 0. \end{array}$$

This is easily seen to be the same as:

$$\begin{array}{lll} \text{(ACP:)} & \text{minimize}_{y,S} & -\ln\det(S) \\ & \text{s.t.} & \sum_{i=1}^{m} y_i A_i + S = C \\ & & S \succ 0. \end{array}$$

Just as in linear inequality systems, the analytic center possesses a very nice "centrality" property in the feasible region P of the semi-definite inequality system. Suppose that $(\hat{y}, \hat{S})$ is the analytic center. Then there are easy-to-construct ellipsoids E_{IN} and E_{OUT}, both centered at $\hat{y}$ and where E_{OUT} is a scaled version of E_{IN} with scale factor n, with the property that:

$$E_{\text{IN}} \subset P \subset E_{\text{OUT}},$$

as illustrated in Figure 5.

16.8.6 SDP for the minimum volume circumscription problem

A given matrix $R \succ 0$ and a given point z can be used to define an ellipsoid in $\mathbb{R}^n$:

$$E_{R,z} := \{y \mid (y-z)^T R(y-z) \leq 1\}.$$

One can prove that the volume of $E_{R,z}$ is proportional to $\sqrt{\det(R^{-1})}$.

Suppose we are given a convex set $X \in \mathbb{R}^n$ described as the convex hull of k points $c_1, \ldots, c_k$. We would like to find an ellipsoid circumscribing these k points that has minimum volume, see Figure 6.

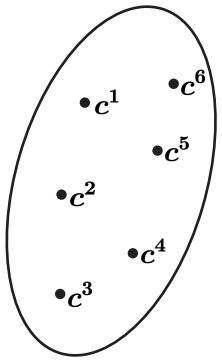


Figure 6: Illustration of the circumscribed ellipsoid problem.

Our problem can be written in the following form:

$$\begin{array}{lll} MCP: & \underset{R,z}{\text{minimize}} & \text{vol}\,(E_{R,z}) \\ & \text{s.t.} & c_i \in E_{R,z}, \quad i = 1, \ldots, k, \end{array}$$

which is equivalent to:

$$\begin{array}{lll} MCP: & \underset{R,z}{\text{minimize}} & -\ln(\det(R)) \\ & \text{s.t.} & (c_i - z)^T R (c_i - z) \leq 1, \quad i = 1, \ldots, k \\ & & R \succ 0, \end{array}$$

Now factor $R = M^2$ where $M \succ 0$ (that is, M is a square root of R), and now MCP becomes:

$$\begin{array}{lll} MCP: & \underset{M,z}{\text{minimize}} & -\ln(\det(M^2)) \\ & \text{s.t.} & (c_i - z)^T M^T M (c_i - z) \leq 1, \quad i = 1, \ldots, k, \\ & & M \succ 0. \end{array}$$

Next notice the equivalence:

$$\begin{pmatrix} I & Mc_i - Mz \\ (Mc_i - Mz)^T & 1 \end{pmatrix} \succeq 0 \quad \Longleftrightarrow \quad (c_i - z)^T M^T M (c_i - z) \leq 1$$

In this way we can write MCP as:

$$\begin{array}{lll} MCP: & \underset{M,z}{\text{minimize}} & -2\ln(\det(M)) \\ & \text{s.t.} & \begin{pmatrix} I & Mc_i - Mz \\ (Mc_i - Mz)^T & 1 \end{pmatrix} \succeq 0, \quad i = 1, \ldots, k, \\ & & M \succ 0. \end{array}$$

Last of all, make the substitution $y = Mz$ to obtain:

$$\begin{array}{lll} MCP: & \underset{M,y}{\text{minimize}} & -2\ln(\det(M)) \\ & \text{s.t.} & \begin{pmatrix} I & Mc_i - y \\ (Mc_i - y)^T & 1 \end{pmatrix} \succeq 0, \quad i = 1, \ldots, k, \\ & & M \succ 0. \end{array}$$

Notice that this last program involves semidefinite constraints where all of the matrix coefficients are linear functions of the variables M and y. The objective function is the logarithmic barrier function $-\ln(\det(M))$. As discussed earlier, this function has the same remarkable properties as the logarithmic barrier function $-\sum_{j=1}^{n}\ln(x_j)$ does for linear optimization, and optimization of this function using Newton's method is extremely easy.

Finally, note that after solving the formulation of MCP above, we can recover the matrix R and the center z of the optimal ellipsoid by computing

$$R = M^2 \text{ and } z = M^{-1}y.$$

16.9 SDP in control theory

A variety of control and system problems can be cast and solved as instances of SDP. However, this topic is beyond the scope of these notes.

16.10 Interior-point methods for SDP

The primal and dual SDP problems are:

$$\begin{array}{rlll} SDP: & \text{minimize} & C \bullet X & \\ & \text{s.t.} & A_i \bullet X = b_i, & i = 1, \ldots, m, \\ & & X \succeq 0, & \end{array}$$

and

$$\begin{array}{rlll} SDD: & \text{maximize} & \sum_{i=1}^{m} y_i b_i & \\ & \text{s.t.} & \sum_{i=1}^{m} y_i A_i + S = C & \\ & & S \succeq 0. & \end{array}$$

If X and (y, S) are feasible for the primal and the dual, the duality gap is:

$$C \bullet X - \sum_{i=1}^{m} y_i b_i = S \bullet X \geq 0.$$

Also,

$$S \bullet X = 0 \iff SX = 0.$$

Interior-point methods for semidefinite optimization are based on the logarithmic barrier function:

$$B(X) = -\sum_{j=1}^{n}\ln(\lambda_i(X)) = -\ln(\prod_{j=1}^{n}\lambda_i(X)) = -\ln(\det(X)).$$

Consider the logarithmic barrier problem $BSDP(\mu)$ parameterized by the positive barrier parameter μ:

$$\begin{array}{rlll} BSDP(\mu): & \text{minimize} & C \bullet X - \mu\ln(\det(X)) & \\ & \text{s.t.} & A_i \bullet X = b_i, & i = 1, \ldots, m, \\ & & X \succ 0. & \end{array}$$

Let $f_\mu(X)$ denote the objective function of $BSDP(\mu)$. Then it is not too difficult to derive:

$$-\nabla f_\mu(X) = C - \mu X^{-1},$$

and so the Karush-Kuhn-Tucker conditions for $BSDP(\mu)$ are:

$$\begin{cases} A_i \bullet X = b_i, \ i = 1, \ldots, m, \\ X \succ 0, \\ C - \mu X^{-1} = \sum\limits_{i=1}^{m} y_i A_i. \end{cases}$$

We can define

$$S = \mu X^{-1},$$

which implies

$$XS = \mu I,$$

and we can rewrite the Karush-Kuhn-Tucker conditions as:

$$\begin{cases} A_i \bullet X = b_i, \ i = 1, \ldots, m, \\ X \succ 0 \\ \sum\limits_{i=1}^{m} y_i A_i + S = C \\ XS = \mu I. \end{cases}$$

It follows that if (X, y, S) is a solution of this system, then X is feasible for SDP, (y, S) is feasible for SDD, and the resulting duality gap is

$$S \bullet X = \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij} X_{ij} = \sum_{j=1}^{n} (SX)_{jj} = \sum_{j=1}^{n} (\mu I)_{jj} = n\mu.$$

This suggests that we try solving $BSDP(\mu)$ for a variety of values of μ as $\mu \to 0$.

Interior-point methods for SDP are very similar to those for linear optimization, in that they use Newton's method to solve the KKT system as $\mu \to 0$.

16.11 Website for SDP

A good website for semidefinite programming is:

`http://www-user.tu-chemnitz.de/ helmberg/semidef.html`.