1 Preliminaries

1.1 Optimization under constraints

The general problem we study in this course takes the form

$$\begin{array}{ll} \text{maximize} & f(x) \\ \text{subject to} & x \in X \\ & g(x) = b \end{array}$$

where

$$\begin{array}{rcll} x & \in & \mathbb{R}^n & (n \text{ decision variables}) \\ f : \mathbb{R}^n & \to & \mathbb{R} & (\text{objective function}) \\ X & \subseteq & \mathbb{R}^n & (\text{regional constraints}) \\ g : \mathbb{R}^n & \to & \mathbb{R}^m & (m \text{ functional equations}) \\ b & \subseteq & \mathbb{R}^m & \end{array}$$

Note that minimizing $f(x)$ is the same as maximizing $-f(x)$. In lectures we will discuss various examples of constrained optimization problems. We will also talk briefly about the way in which our methods are applied to real-world problems.

1.2 Representation of constraints

Slack variables
Sometimes we shall let the constraint be $g(x) \le b$. This form of constraint can be turned into an equality constraint by the addition of a **slack variable** z. We write

$$g(x) + z = b, \quad z \ge 0.$$

It will frequently be mathematically useful to turn all our constraints into equalities.

Regional and functional constraints
There is freedom in choosing how to divide the constraints between **regional constraints** of the form

$$x \in X$$

and **functional constraints** of the form

$$g(x) = b.$$

Together, these define the **feasible set** for x. Typically, 'obvious' constraints like $x \ge 0$ are catered for by defining X in an appropriate way and more complicated constraints, that may change from instance to instance of the problem, are expressed by functional constraints $g(x) = b$ or $g(x) \le b$.

Sometimes the choice is made for mathematical convenience. Methods of solution typically treat region and functional constraints differently.

1.3 Geometry of linear programming

Consider the two problems P and D.

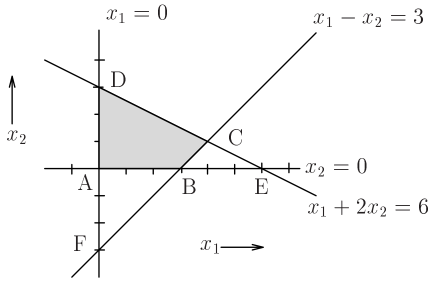
$$\begin{array}{llrcl} \text{P:} & \text{maximize} & x_1 + x_2 & & \\ & \text{subject to} & x_1 + 2x_2 & \le & 6 \\ & & x_1 - x_2 & \le & 3 \\ & & x_1, x_2 & \ge & 0 \\ \text{D:} & \text{minimize} & 6\lambda_1 + 3\lambda_2 & & \\ & \text{subject to} & \lambda_1 + \lambda_2 & \ge & 1 \\ & & 2\lambda_1 - \lambda_2 & \ge & 1 \\ & & \lambda_1, \lambda_2 & \ge & 0 \end{array}$$

These are completely linear problems – the objective function and all constraints are linear. The two problems are related, as we shall see. Note that in matrix/vector notation we can write

$$\text{P:} \;\; \text{maximize } c^T x \quad \text{s.t. } Ax \le b, \;\; x \ge 0,$$

$$\text{D:} \;\; \text{minimize } \lambda^T b \quad \text{s.t. } A^T\lambda \ge c, \;\; \lambda \ge 0.$$

Concentrate on problem P for the moment. The shaded region is the feasible set defined by the constraints for P.

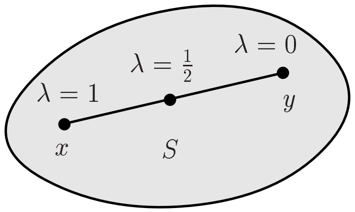


We now state some definitions and theorems that are at the heart of the theory of **linear programming** (LP) problems. The theorems are more or less obvious for our example. They will be proved in future lectures.

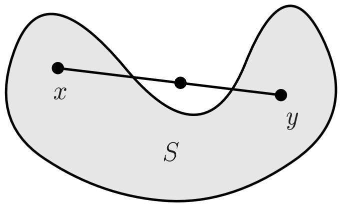
Definition 1.1 *A set $S \subseteq \mathbb{R}^n$ is a* **convex set** *if $x, y \in S \implies \lambda x + (1-\lambda) y \in S$ for all $x, y \in S$ and $0 \le \lambda \le 1$.*

In other words, the line segment joining x and y lies in S.

Theorem 1.1 *The feasible set of a LP problem is convex.*



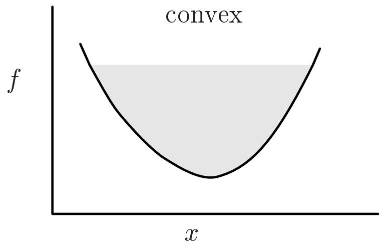
convex

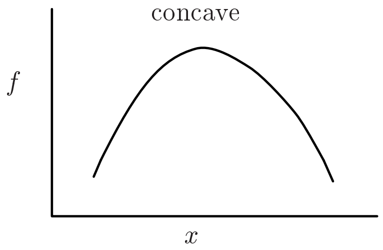


not convex

For functions defined on convex sets we make the following definition.

Definition 1.2 *A function $f : S \to \mathbb{R}$ is a* **convex function** *if the set above its graph is convex. Equivalently, $\lambda f(x) + (1-\lambda)f(y) \geq f(\lambda x + (1-\lambda)y)$, for all $0 \leq \lambda \leq 1$.*





For general S and f there may be local minima (maxima) of f over S which are not global minima (maxima). However, if S is a convex set and f is a convex (concave) function then you can be sure that any local minimum (maximum) of f is also a global minimum (maximum). Note that a linear function is both convex and concave, and so all local optima of a linear objective function are also global optima.

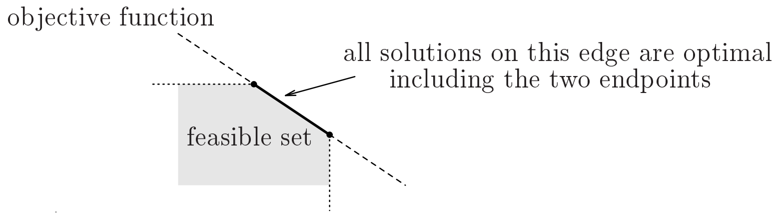
1.4 Extreme points and optimality

Notice that for problem P the optimum occurs (regardless of what the linear objective function is) at a 'corner' of the feasible set. (In our case at C.)

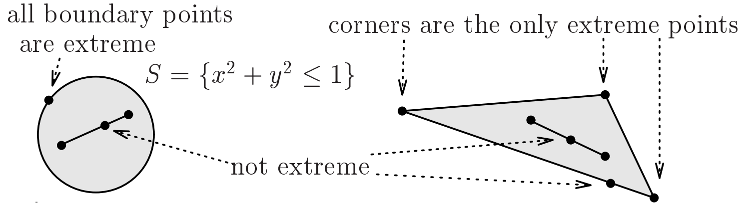
There may be optima elsewhere on the edge if the objective function is parallel to an edge, but there is always still an optimum at a 'corner'. This motivates the following definition.

Definition 1.3 *We say x is an* **extreme point** *of a convex set S if whenever $x = \lambda y + (1-\lambda)z$, for $y, z \in S$, $0 < \lambda < 1$, then $x = y = z$.*

That is, x is not in the interior of any line segment.



Examples of extreme points of two convex sets



Theorem 1.2 *If an LP has a finite optimum it has an optimum at an extreme point of the feasible set.*

For LP problems the feasible set will always have a finite number of extreme points (vertices). The feasible set is 'polyhedral', though it may be bounded or unbounded. This suggests the following algorithm for solving LPs.

Algorithm:
1. Find all the vertices of the feasible set.
2. Pick the best one.

This will work, but there may be very many vertices. In fact, for $Ax \leq b$, $x \geq 0$, there can be $\binom{n+m}{m}$ vertices. So if $m = n$, say, then the number of vertices is of order $(2n)^n$, which increases exponentially in n. This is **not** a good algorithm!

1.5 *Efficiency of algorithms*

There is an important distinction between those algorithms whose running times (in the worst cases) are exponential functions of 'problem size', e.g., $(2n)^n$, and those algorithms whose running times are polynomial functions of problem size, e.g., n^k. For example, the problem of finding the smallest number in a list of n numbers is solvable in polynomial-time n by simply scanning the numbers. There is a beautiful theory concerning the **computational complexity** of algorithms and one of its main messages is that problems solvable in polynomial-time are the 'easy' ones.

In fact, there exists a polynomial-time algorithm for general LP problems, but this was not known until 1981. In contrast, no polynomial-time algorithm is known for general **integer LP**, in which x is restricted to be integer-valued. ILP includes important problems such as bin packing, job-shop scheduling, traveling salesman and many other essentially equivalent problems of a combinatorial nature.