Chapter 2

Linear programming

2.1 The problem

We start with a minimal *diet problem.* We have three types of food available to us: bread, milk, and eggs. We have three nutrients we need above certain minimum levels: carbohydrates, proteins, and vitamins. We want to minimise the cost of the food we need whilst ensuring we get enough nutrients.

2.1.1 The problem in words

The example numbers we will use are as follows. The unit cost (cost in GB pounds per 100 gram serving) of each food type is

Food	Cost
Bread	2.5
Milk	1.2
Eggs	0.8

The amount (in grams) of each nutrient per unit (100 gram serving) of food is

Food	Carbs	Proteins	Vitamins
Bread	300	5	0.07
Milk	30	50	0.02
Eggs	20	90	0.12

To stay healty, the daily diet should contain at least the minimum quantity of each nutrient, which is

Nutrient	Minimum quantity
Carbs	300
Proteins	600
Vitamins	1

2.1.2 The problem as equations

We will now abstract away the details to construct a problem to solve.

The variables we care about are the number of servings of each food stuff. We denote these x_B, x_M, x_E for the number of servings of bread, milk, and eggs respectively. These are real numbers (we can eat a fraction of a serving), but cannot be negative (we cannot eat a negative amount of food). Hence

$$x_B, x_M, x_E \geq 0. \tag{2.1}$$

The total cost of the diet is the sum of the unit costs multiplied by the number of servings in the diet. This is the *objective function* that we want to minimise. Hence

$$\min 2.5x_B + 1.2x_M + 0.8x_E \tag{2.2}$$

is the total cost to be minimised.

Finally, we have the constraints, which are the nutritional needs for the diet. Each separate nutrient gets its own equation. The total amount of each nutrient in the diet is the amount of nutrients per unit serving multiplied by the number of servings. Therefore the constraints are

$$\begin{array}{rcrcrcll} 300x_B & + & 30x_M & + & 20x_E & \geq & 300 & : \text{minimum carbs,} \\ 5x_B & + & 50x_M & + & 90x_E & \geq & 600 & : \text{minimum proteins,} \\ 0.07x_B & + & 0.02x_M & + & 0.12x_E & \geq & 1 & : \text{minimum vitamins.} \end{array} \tag{2.3}$$

Typically the mathematical problem is summarized as

$$\begin{array}{lrcrcrcl} \min & 2.5x_B & + & 1.2x_M & + & 0.8x_E & & \\ \text{subject to} & 300x_B & + & 30x_M & + & 20x_E & \geq & 300 \\ & 5x_B & + & 50x_M & + & 90x_E & \geq & 600 \\ & 0.07x_B & + & 0.02x_M & + & 0.12x_E & \geq & 1 \\ & x_B, & & x_M, & & x_E & \geq & 0. \end{array} \tag{2.4}$$

2.1.3 A more general form

The specific coefficients are needed to solve the problem, but obscure some of the mathematical structure. We introduce more general notation to make the structure clearer. First, the number of constraints will be denoted m (this is the number of nutrients in this example). Next, the number of foods will be denoted n (this is the number of variables in the objective function). Then the problem can be written as

$$\begin{array}{llcll} \min & \sum_{j=1}^n c_j x_j & & & \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j & \geq & b_j & \forall i = 1, \dots, m \\ & x_j & \geq & 0 & \forall j = 1, \dots, n. \end{array} \tag{2.5}$$

The coefficients are written a_{ij} (the amount of the i^{th} nutrient per unit of the j^{th} food), b_j (the daily required amount of the j^{th} nutrient), and c_j (the cost of the j^{th} food).

2.2 The graphical solution

We will simplify the problem even further to illustrate the key points of the solution method. Restrict to two variables (for example, consider only bread and eggs, as no milk is available). We will look at the simple problem

$$\begin{array}{lrcrcl} \max & 0.02x_A & + & 0.03x_B & & \\ \text{subject to} & x_A & + & x_B & \leq & 10 \\ & x_A & & & \leq & 7 \\ & & & x_B & \leq & 5 \\ & x_A, & & x_B & \geq & 0. \end{array} \tag{2.6}$$

This is a problem with three constraints ($m = 3$) and two variables ($n = 2$), and the coefficients can be read off from the equations (for example, $c = (0.02, 0.03)$). Note that key points of the problem are reversed compared to the diet problem: we are maximising, not minimising, and the constraints are less-than-or-equals rather than greater-than-or-equals.

Each of the constraints describes a line in $\mathbb{R}^2$ which splits the plane into two regions. In one region the constraint is satisfied; in the other it is not. We want to *draw* the region where all the constraints are satisfied. This is the *feasible region*. An example is in figure 2.1.

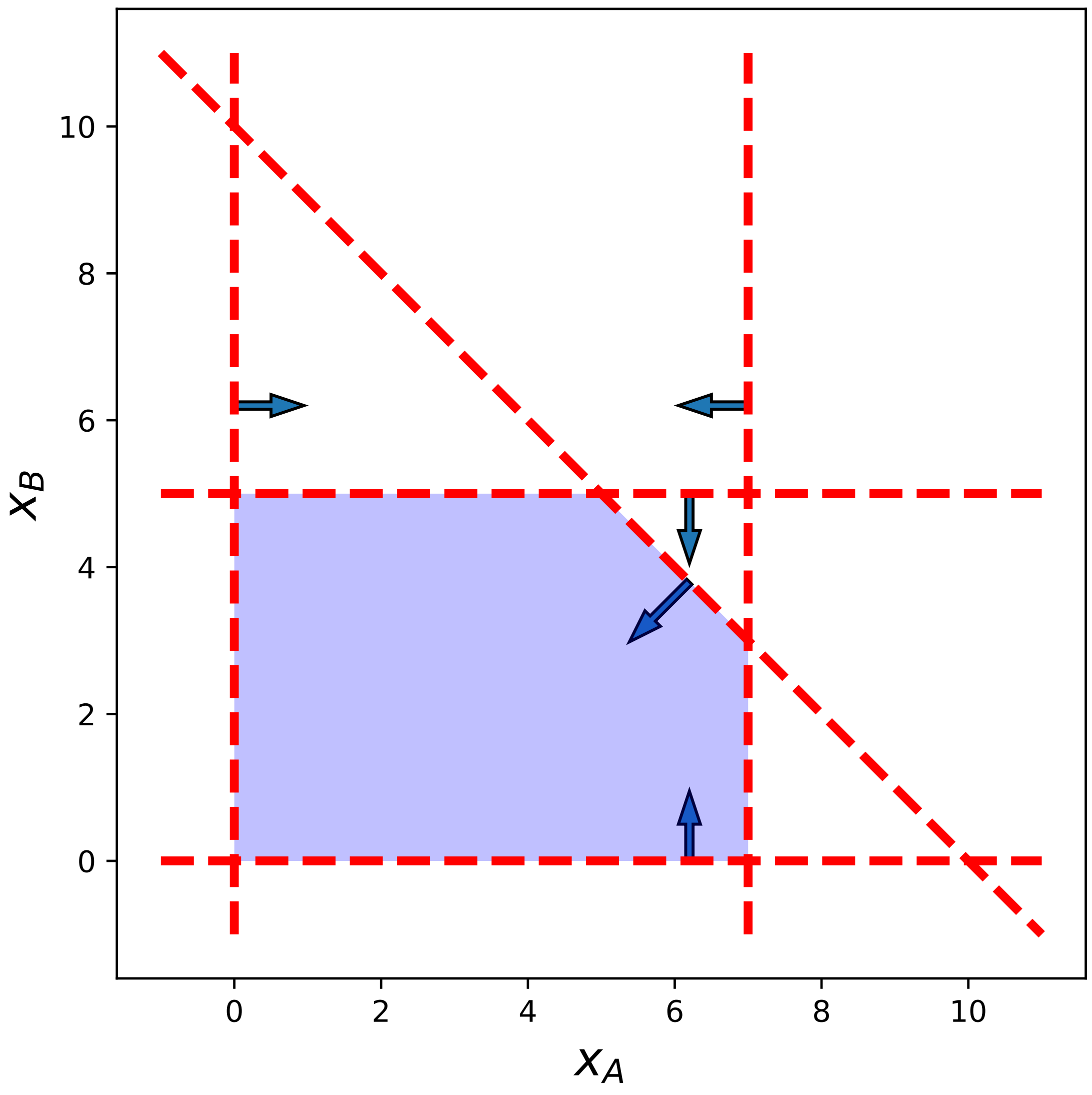

Figure 2.1: The feasible region (shaded blue) constructed from the constraints (red dashed lines, with arrows pointing to which side is feasible) for a problem with two variables x_A, x_B.

We want to find the point in the feasible region where the objective function is maximised (in this case: in the diet problem, we want to minimise). The *level curves* of the objective function are lines where $f(x) = 0.02x_A + 0.03x_B = z$ are constant. These are straight lines which we can also plot. We also need the line that is orthogonal to the level curves: this is the gradient of the objective function ∇f which, in this case, is $(0.02, 0.03)$. For linear problems we generally have $\nabla f = c$.

Our solution method is then as follows. Start from some point within the feasible region (for example, the origin). Move in the direction of the gradient ∇f that increases the objective function z. Draw level curves through this point. Each point that intersects the feasible region is a valid solution to the problem with the value z of the objective function. The last level curve (as we move along the gradient vector) that we can draw will give us the optimal value with its optimal solution x^*. This is illustrated in figure 2.2.

Note that the solution we find has to be at a *corner* or *vertex* of the feasible region. Whilst the graphical solution only works for problems with two variables, the intuition (using the gradient and looking at vertices of the feasible region) extends to the general case.

2.3 Properties

Before we construct the general solution method it is useful to look at the form and structure of the problem. This is because there are multiple ways of looking at the problem, each of which is useful in different cases. There are also rules to transform between the cases.

2.3.1 The different forms

Canonical form

This is the form used so far:

$$\begin{array}{lrcll} \max & \sum_{j=1}^{n} c_j x_j & & & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \leq & b_j & \forall i = 1, \dots, m \\ & x_j & \geq & 0 & \forall j = 1, \dots, n. \end{array} \tag{2.7}$$

This is a maximisation problem, with the inequalities being less-than's.

Standard form

This is a new form:

$$\begin{array}{lrcll} \min & \sum_{j=1}^{n} c_j x_j & & & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & = & b_j & \forall i = 1, \dots, m \\ & x_j & \geq & 0 & \forall j = 1, \dots, n. \end{array} \tag{2.8}$$

This is a minimisation problem. We no longer have inequalities, but only equalities.

Simplified canonical form

This is the simplification of the canonical form we have used so far:

$$\begin{array}{lrcll} \max & \sum_{j=1}^{n} c_j x_j & & & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \leq & b_j & \forall i = 1, \dots, m. \end{array} \tag{2.9}$$

The key difference with canonical form is that we do not impose the variables to be non-negative.

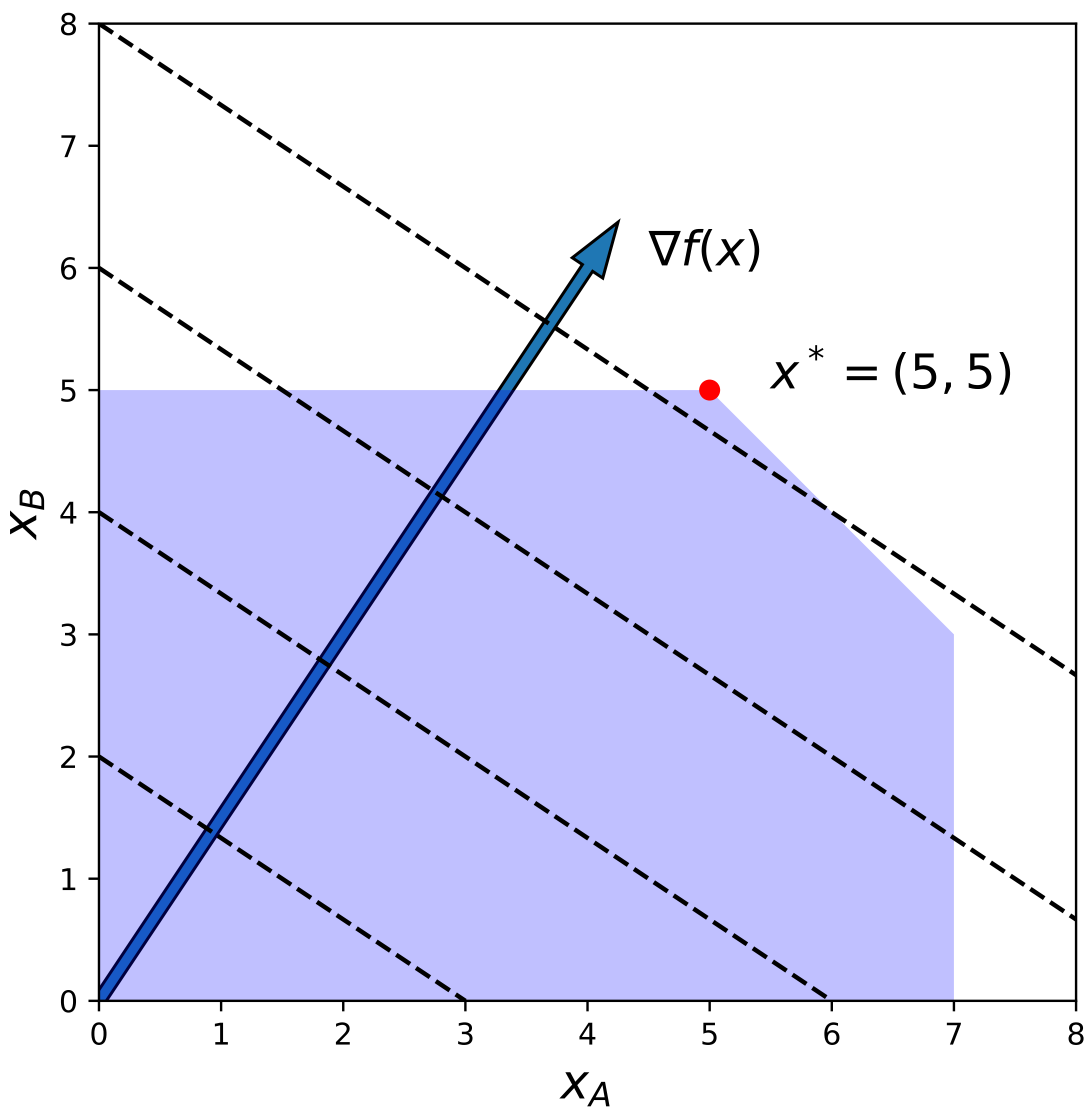

Figure 2.2: The optimal solution (red dot) is the last point in the feasible region (shaded blue) crossed by a level curve (black dashed lines) when moving in the direction of increasing objective function (the arrow pointing in the direction $\nabla f = c$).

Matrix form

This is the canonical form written in the more compact matrix notation:

$$\begin{array}{lrcl} \max & cx & & \\ \text{subject to} & Ax & \leq & b \\ & x & \geq & 0. \end{array} \tag{2.10}$$

Here $A \in \mathbb{R}^{m \times n}$ is a matrix with entries a_{ij}, and $c, x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ are vectors with entries c_j, x_j and b_j respectively. Note that in the objective function we have written

$$cx = c \cdot x = \sum_j c_j x_j \tag{2.11}$$

for the inner product between the vector c and the vector x.

This matrix form is useful for its compactness and the links to Linear Algebra concepts it allows us to make.

2.3.2 Transformation rules

There are a range of rules that allow us to transform between, for example, canonical and standard form, or to change a problem in neither standard form into one of the standard forms.

1. Convert between maximisation and minimisation:

$$\max cx \implies -\min(-cx), \tag{2.12}$$
$$\min cx \implies -\max(-cx). \tag{2.13}$$

2. Change direction of an inequality:

$$ax \leq b \implies -ax \geq -b, \tag{2.14}$$
$$ax \geq b \implies -ax \leq -b. \tag{2.15}$$

3. Turn an equation into two inequalities:

$$ax = b \implies ax \geq b \text{ and } ax \leq b. \tag{2.16}$$

4. Turn an inequality into an equation:

   (a) For $\leq$ inequalities, introduce $s \geq 0$, a *slack* variable. Then

$$ax \leq b \implies ax + s = b, \text{ with } s \geq 0. \tag{2.17}$$

   (b) For $\geq$ inequalities, introduce $s \geq 0$, a *surplus* variable. Then

$$ax \geq b \implies ax - s = b, \text{ with } s \geq 0. \tag{2.18}$$

5. Turn any variable x_j without a sign restriction into two variables $x_j^{\pm} \geq 0$, where $x_j = x_j^+ - x_j^-$.

It is important to note that a number of these steps introduce additional variables and constraints. The exact number of variables and constraints are therefore linked to the form in which we write the problem, and are not properties of the problem itself.

2.4 Geometry of linear programming

2.4.1 Setup

For this section we will work with the simplified canonical form in matrix notation,

$$\begin{array}{lccl} \max & cx & & \\ \text{subject to} & Ax & \leq & b \\ & x & & \text{free.} \end{array} \tag{2.19}$$

We can write any non-negativity constraint as $e_{(j)}x \leq 0$, where $e_{(j)}$ is a vector that is zero in all entries except the j^{th}, where it is 1. This additional constraint can be embedded in the matrix form as an additional row in $Ax \leq b$.

2.4.2 Purpose

Start by remembering the graphical solution method. We argued that the optimal solution had to lie on a vertex of the feasible region. We also argued that we could sort these points using the gradient of the objective function.

The purpose of this section is to prove (at some level of rigour) that this intuition extends away two variables, $n = 2$, where the feasible region is a subset of the plane $\mathbb{R}^2$, to the general case where the feasible region is a subset of $\mathbb{R}^n$. That is, we want to show that the optimal solution lies on a vertex of this feasible region.

2.4.3 Fundamental theorem of Linear Programming

Theorem 2.4.1 (Fundamental theorem)**.** *Let $P = \{x \in \mathbb{R}^n \colon Ax \leq b\}$ and consider the linear program* $\max\{cx \colon x \in P\}$*. If P is non-empty, the linear program either admits an optimal solution x^* corresponding to one of its vertices, or it is unbounded.*

Before we consider the proof let us look at what this says. The region P defined in the theorem is the feasible region. We have constructed the subset of $\mathbb{R}^n$ in which the constraints are satisfied. As noted in the theorem, it may be that there are no points where the constraints are satisfied ("if P is non-empty"): we ignore that case. It is also possible that the feasible region is unbounded and the objective function can increase without limit: this case (where the linear program is unbounded) also needs checking.

To prove this theorem we need to be able to characterise any point in P in terms of its vertices in some way. To do this, we need a number of definitions and preliminary results.

Definitions

Consider a single inequality $a_i x \leq b_i$. This splits $\mathbb{R}^n$ into two pieces. Define $\{x \in \mathbb{R}^n \colon a_i x \leq b_i\}$ to be the *halfspace* of points satisfying the inequality. Further define $\{x \in \mathbb{R}^n \colon a_i x = b_i\}$ to be the *hyperplane* of points satisfying the inequality as an equation. The hyperplane bounds the halfspace. The vector a_i is orthogonal to the hyperplane and points out of the halfspace. This is illustrated in figure 2.3.

The feasible region $P = \{x \in \mathbb{R}^n \colon Ax \leq b\}$ is obtained by intersecting the m halfspaces for each inequality. The intersection of a finite number of halfspaces is called a *polyhedron*. Thus the feasible region is a polyhedron.

Given k points $x_1, \dots, x_k \in \mathbb{R}^n$, their *convex combinations* are all the points $\{y\}$, with

$$y = \sum_{i=1}^{k} \lambda_i x_i, \tag{2.20}$$

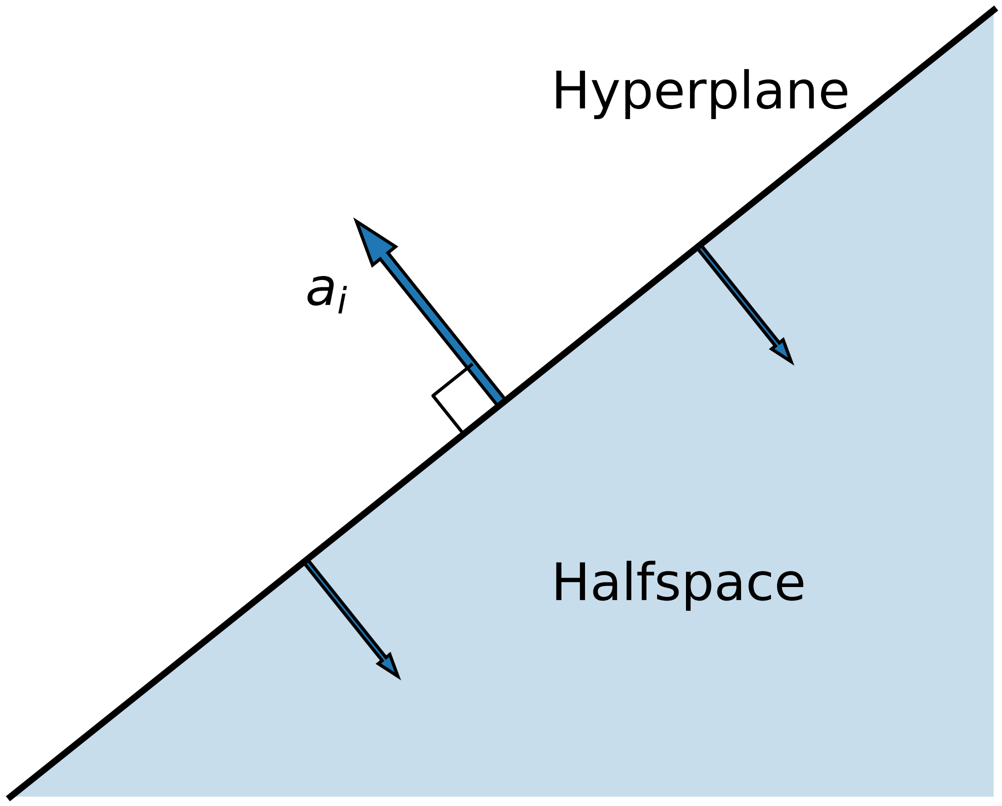

Figure 2.3: A hyperplane splitting $\mathbb{R}^n$ into two pieces. The halfspace (shaded blue) is where the inequality is satisfied. a_i is orthogonal to the hyperplane.

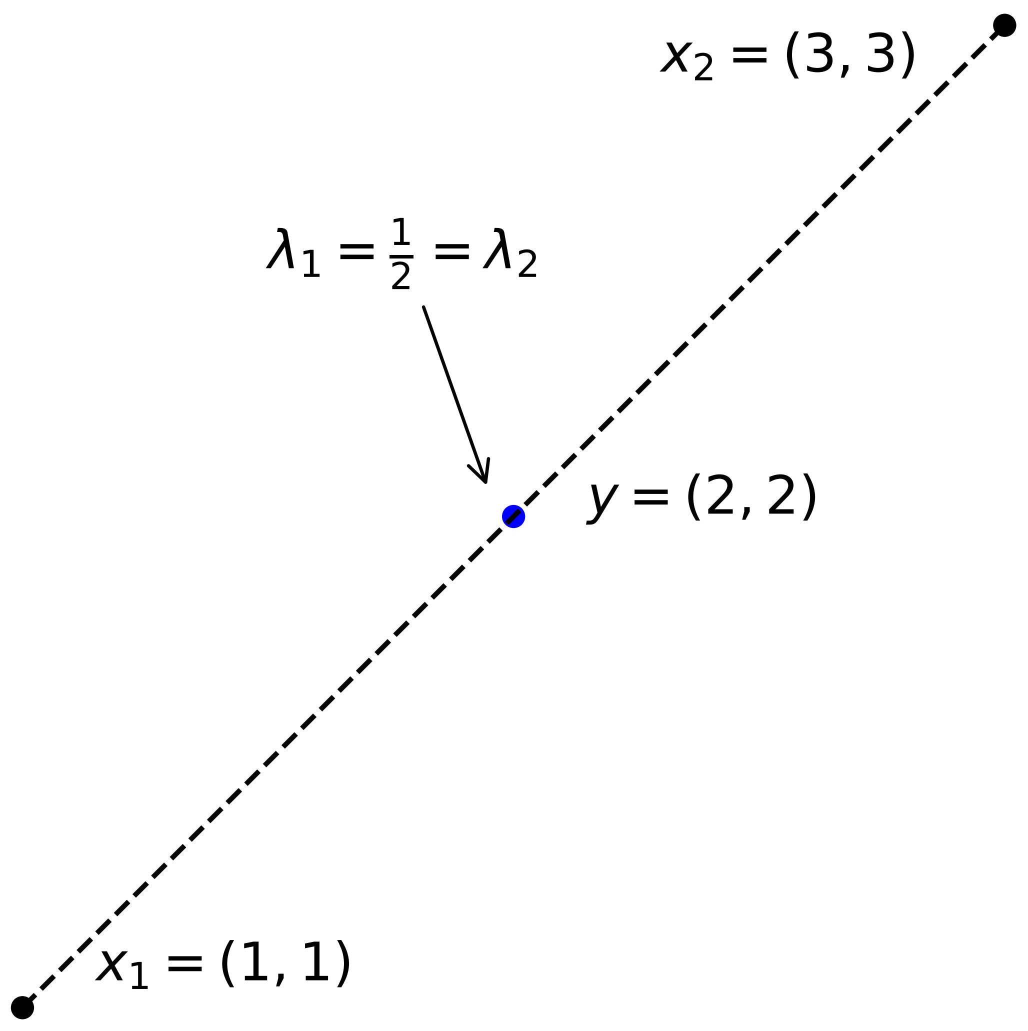

Figure 2.4: All convex combinations of two points lie on the line segment between those two points.

with the restrictions

$$\sum_{i=1}^{k} \lambda_i = 1, \tag{2.21}$$

$$\lambda_1, \ldots, \lambda_k \geq 0. \tag{2.22}$$

Intuitively, if the x_i points are the vertices of a face of the polyhedron, the convex combinations describe all points on the face. Illustrations of convex combinations are in figure 2.4 for two points and in figure 2.5 for three points.

The *conic combinations* are all the points without the restriction that the multipliers λ_i sum to one. Intuitively they generate the hyperplane in full. For two points this is illustrated in figure 2.6.

A set $S \subseteq \mathbb{R}^n$ is *convex* if it contains the convex combinations of all of its elements, that is

$$\lambda x_1 + (1-\lambda)x_2 \in S \quad \forall x_1, x_2 \in S \text{ and } \forall \lambda \in [0,1]. \tag{2.23}$$

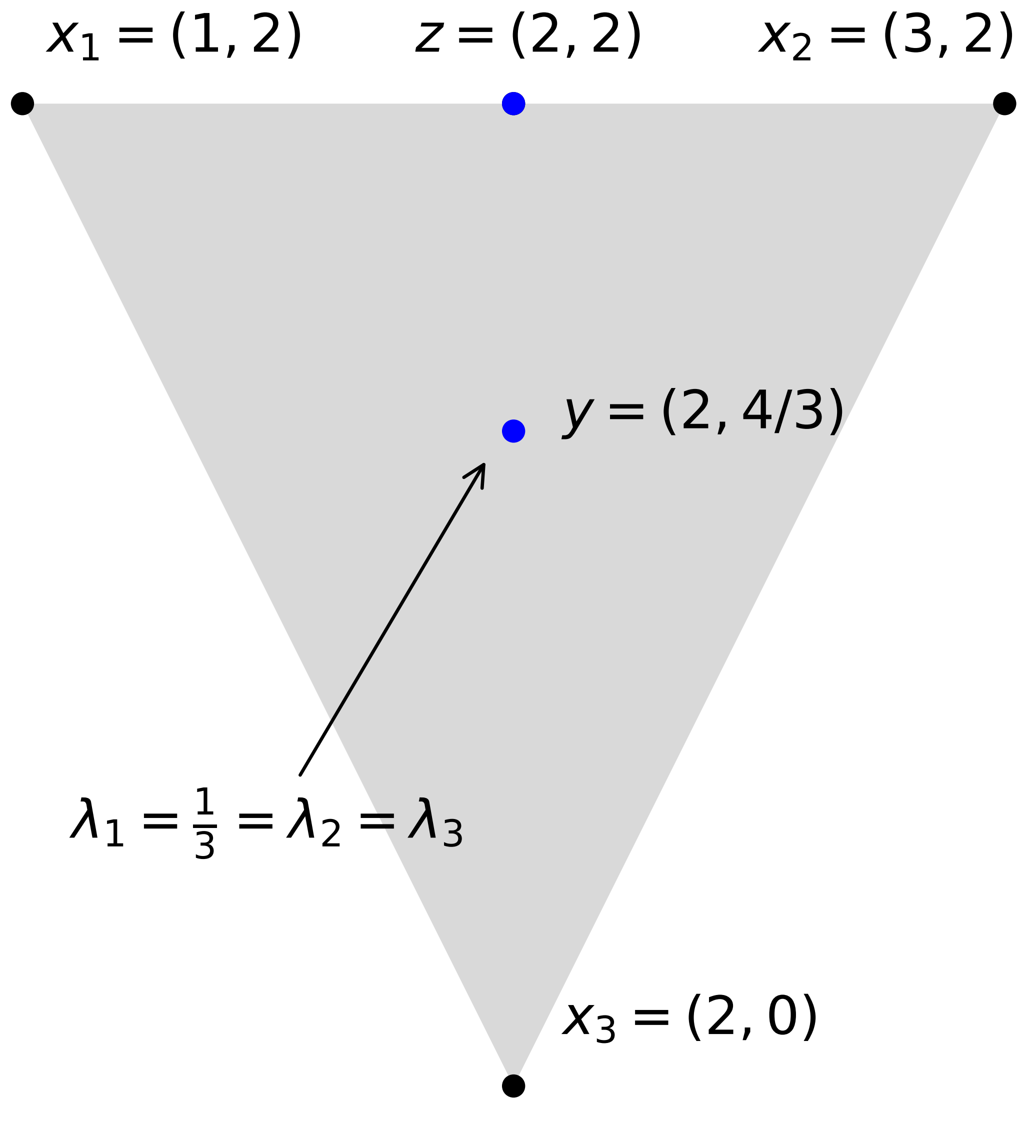

Figure 2.5: All convex combinations of three points form a triangle in a plane within $\mathbb{R}^n$.

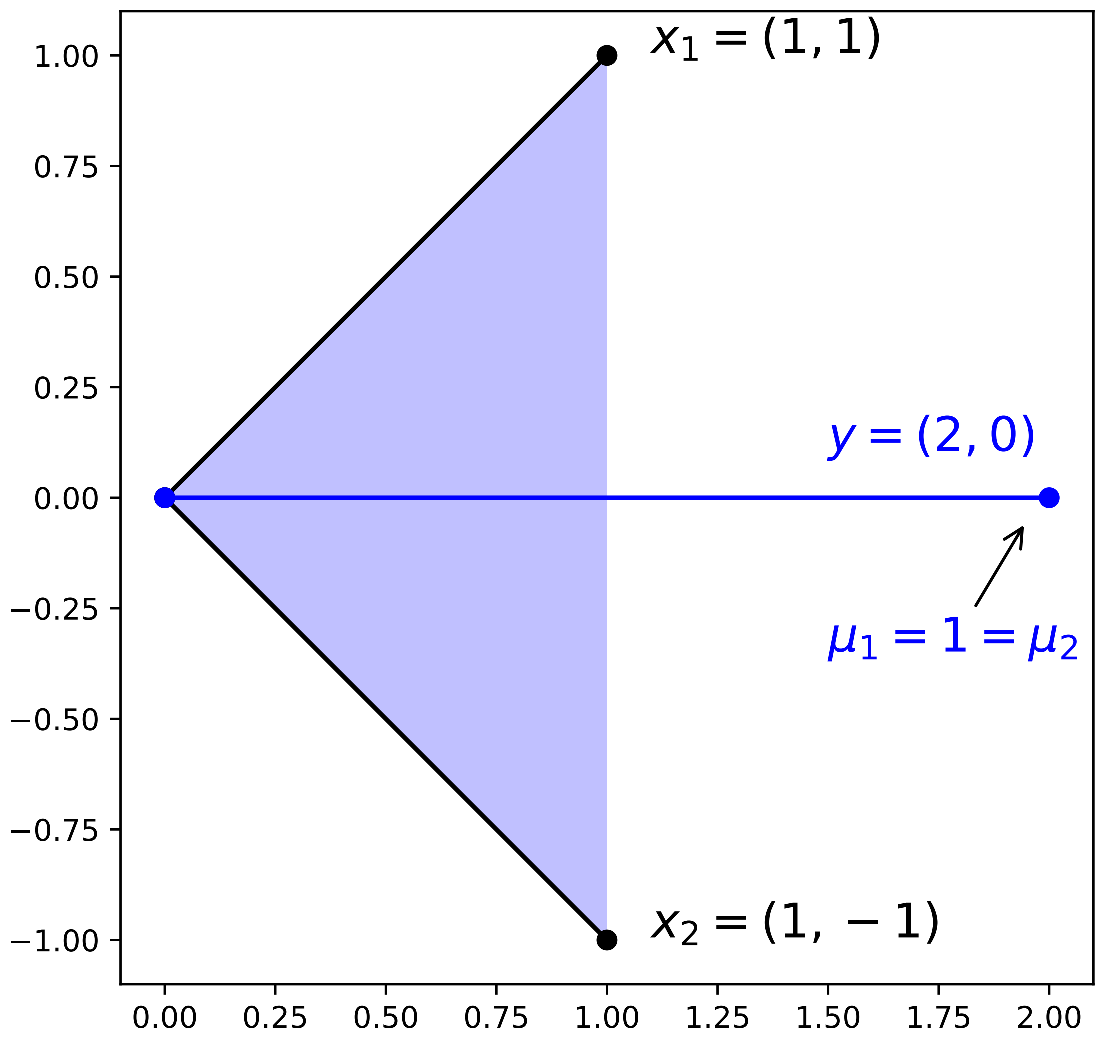

Figure 2.6: The conic combinations of two points generation a quadrant of $\mathbb{R}^2$.

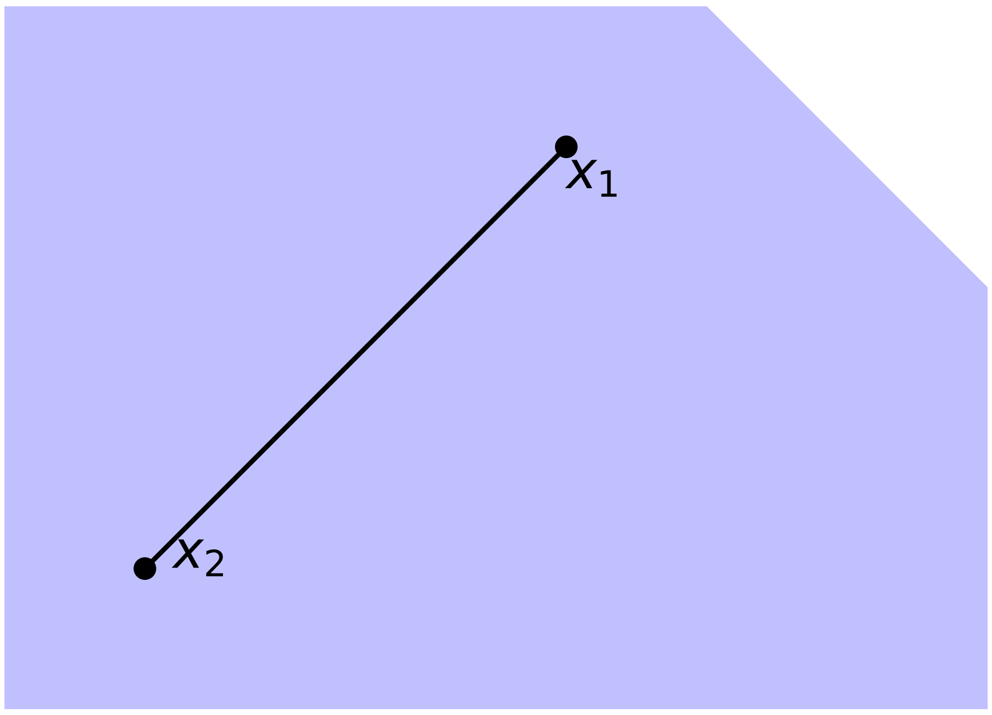

Figure 2.7: A convex set has all points linked by line segments that lie within the set.

We think of this as meaning that the straight line segment linking $x_1, x_2 \in S$ is contained within S. Within the plane $\mathbb{R}^2$ a circle is convex and a “C” shape is not. These are illustrated in figures 2.7 and 2.8 respectively.

For a given polyhedron P, a *vertex* v is a point that cannot be written as a convex combination of two other points x, y in P (with v, x, y all distinct). A *face* is any planar surface belonging to the boundary of P. A *facet* is any $n-1$-dimenional face. This is illustrated in figure 2.9.

For a set $S \subseteq \mathbb{R}^n$, the vector $r \in \mathbb{R}^n$ is a *ray* if

$$x_0 + \mu r \in S \quad \forall x_0 \in S \text{ and } \forall \mu \geq 0. \tag{2.24}$$

This requires that the *semiline* (starting from x_0 and moving in the direction of r) is completely contained in S (for all points $x_0 \in S$).

An *extreme ray* is a ray that cannot be expressed as a conic combination of two other distinct rays p, q, where r, p, q are all distinct. This is illustrated in figure 2.10.

A set is *bounded* if the norm of all vectors in the set is bounded. That is, $\exists \delta \geq 0$ such that $\|x\| \leq \delta \; \forall x \in S$.

We call a bounded polyhedron a *polytope*. This is illustrated in figure 2.11.

2.4.4 Key results

After that blizzard of definitions we can prove some key results.

Theorem 2.4.2. *Polytopes have no rays.*

Proof. If a polytope had a ray it would contain a semiline $x_0 + \mu r$ for all values of μ. By increasing μ we can increase $\|x\|$ without bound. This contradicts that a polytope must be bounded. $\square$

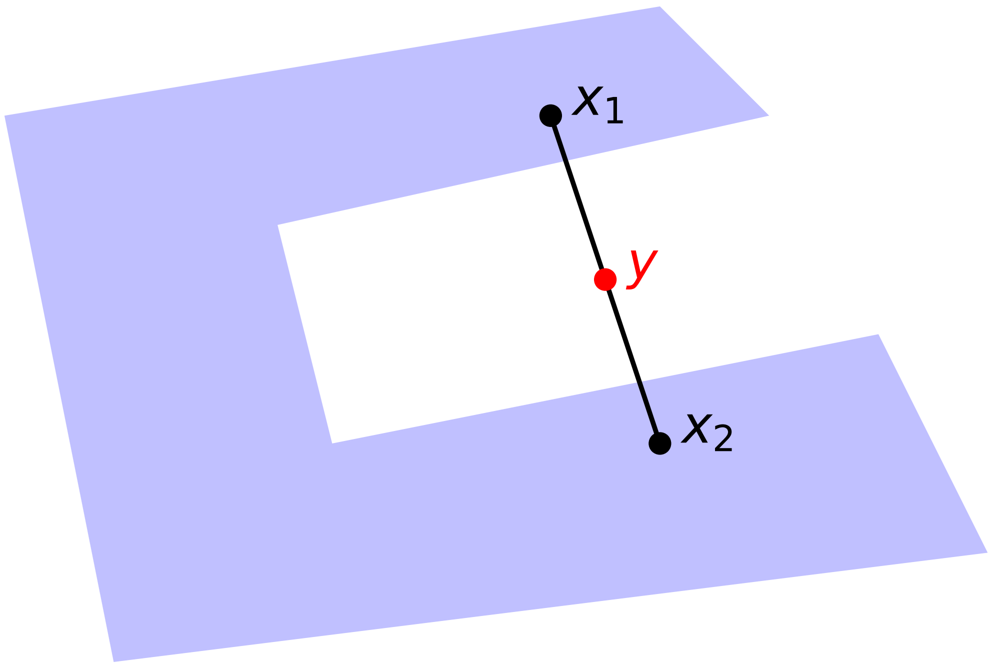

Figure 2.8: A non-convex set has some pair of points that are not linked by a line segment that lies within the set.

Theorem 2.4.3. *The feasible region is convex.*

Proof. Work by contradiction. Assume that P, the polyhedron which is the feasible region, is not convex. Then there exists $x, y \in P$ such that the line segment joining x, y has points not in P. Start at $x \in P$ and move along the line segment until we reach the boundary of P. By construction this point must be in a hyperplane. By definition all points on one side of the hyperplane are in P and all points on the other side are not. As the line segment has not yet left P we have not yet reached y, so y is on the other side of the hyperplane to x. Therefore $y \notin P$. This contradicts our assumption. $\square$

Convexity makes constructing an algorithm to solve the linear program much simpler. Non-convex problems are much harder to solve. However, our main concern is linking the vertices to points in the interior in order to prove the fundamental theorem.

Theorem 2.4.4 (Weyl-Minkowski theorem)**.** *For a subset $P \subseteq \mathbb{R}^n$, the following statements are equivalent:*

- *P is a non-empty polyhedron;*
- *There are two sets, one of $k \geq 1$ vertices $v_1, \dots, v_k$, and another of $h \geq 0$ extreme rays $r_1, \dots, r_h$, such that any point $x \in P$ can be written*

$$x = \sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{h} \mu_j r_j. \tag{2.25}$$

The proof of this theorem is complex and not our key purpose. The point, for our purpose, is that it allows us to express any point within P in terms of its vertices. This is illustrated in figure 2.12. This is what we needed to prove the fundamental theorem.

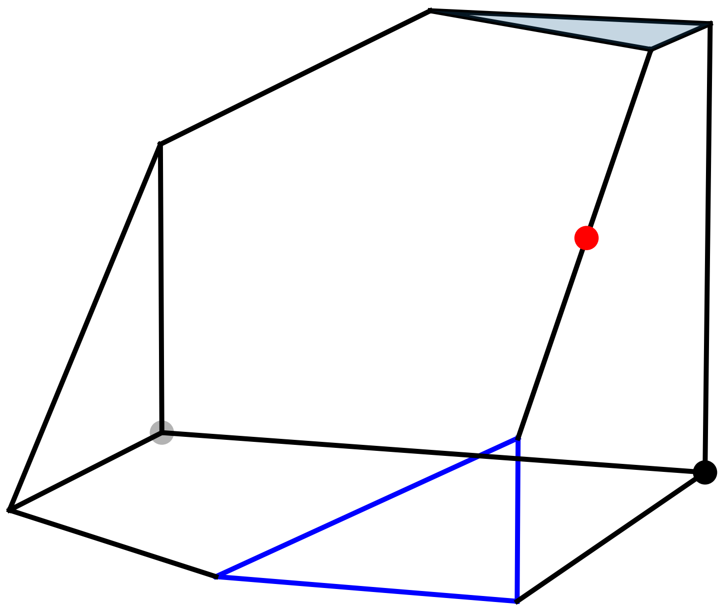

Figure 2.9: A vertex is a corner. The black circles are vertices; the red circle is not. A face is any planar part of the boundary. The blue lines are one dimenional faces; the blue shaded region is a two dimensional face. A facet is an $(n-1)$-dimesional face. The shaded region is a facet as well as a face, whilst the edges are not facets.

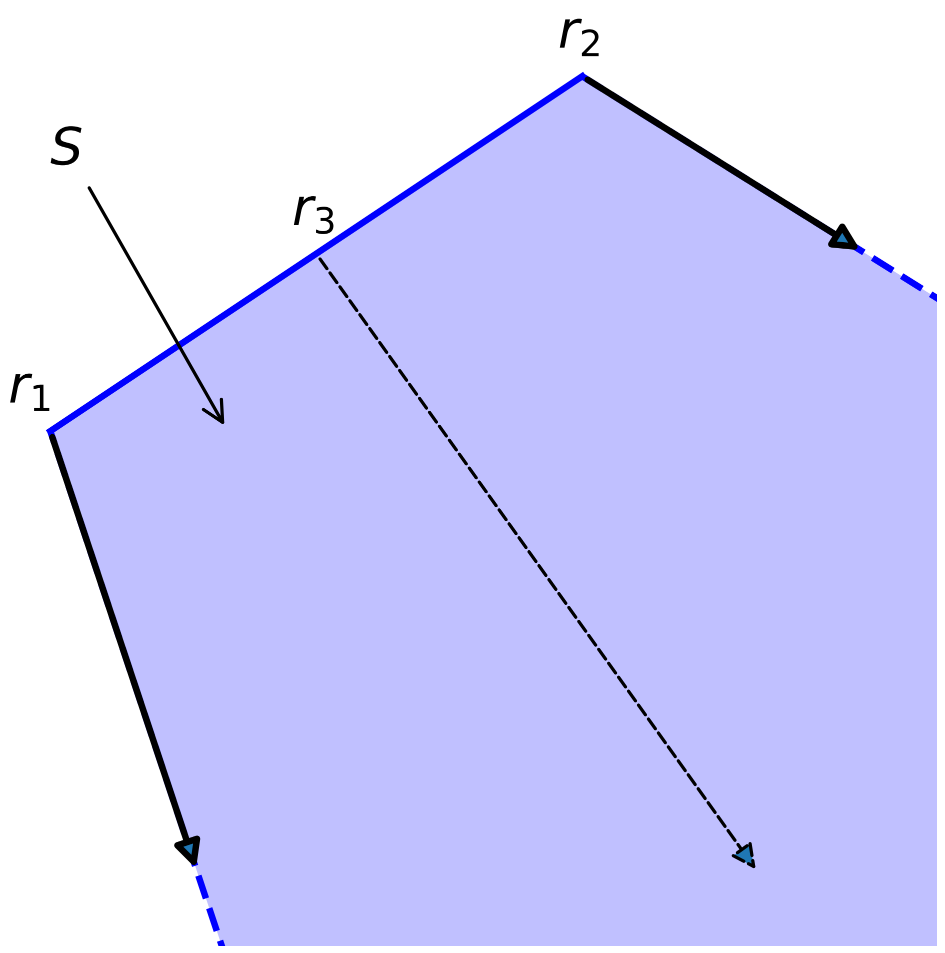

Figure 2.10: The given set S is the blue shaded region, assumed to extend indefinitely down and to the right. The semilines r_1 and r_2 are extreme rays. The semiline r_3 is a ray but not an extreme one.

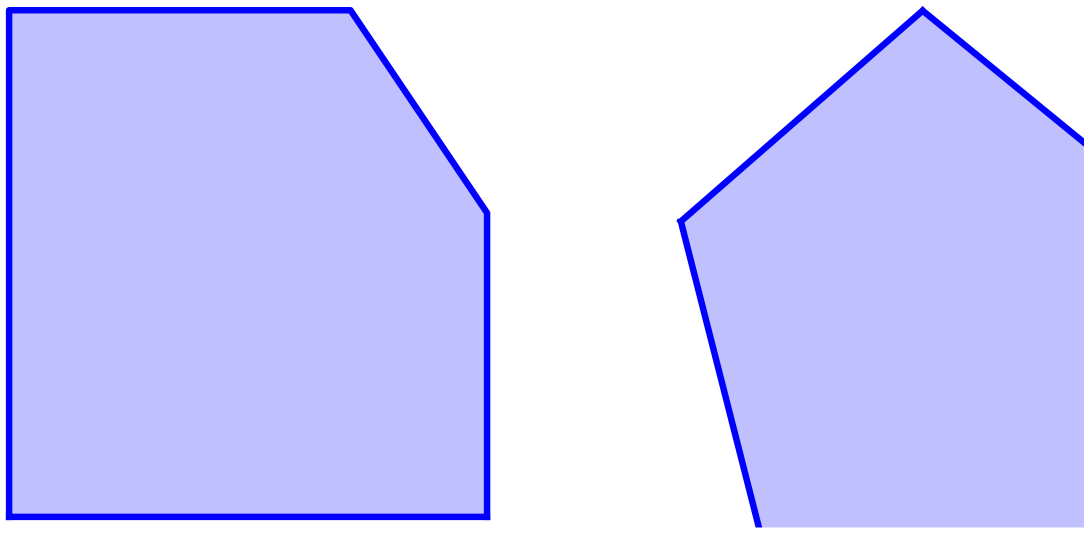

Figure 2.11: The shaded region on the left is bounded by the solid blue lines, and is therefore a polytope. The shaded region on the right is not bounded at the bottom and right and assumed to extend indefinitely. It is therefore not a polytope.

2.4.5 Proof of the Fundamental Theorem

Theorem 2.4.5 (Fundamental theorem)**.** *Let $P = \{x \in \mathbb{R}^n \colon Ax \leq b\}$ and consider the linear program $\max\{cx \colon x \in P\}$. If P is non-empty, the linear program either admits an optimal solution x^* corresponding to one of its vertices, or it is unbounded.*

Proof. Let $x \in P$ be a point in the feasible region. Write

$$x = \sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{h} \mu_j r_j \tag{2.26}$$

using the Weyl-Minkowski theorem. The values of λ, μ are constrained to be non-negative, and additionally the values of λ must sum to one.

We now want to *change variables* from x to λ, μ. That is, we want to re-write our linear program in terms of the vector (λ, μ). First re-write the objective function as

$$cx = \sum_{i=1}^{k} \lambda_i (cv_i) + \sum_{j=1}^{h} \mu_j (cr_j). \tag{2.27}$$

This sums over every entry of the vector (λ, μ); the coefficients are the inner product of c with either a vertex v_i or a ray r_j.

Next, write the constraints applied to (λ, μ). These are

$$\sum_{i=1}^{k} \lambda_i = 1, \tag{2.28}$$

$$\lambda_1, \ldots, \lambda_k \geq 0, \tag{2.29}$$

$$\mu_1, \ldots, \mu_h \geq 0. \tag{2.30}$$

We are therefore solving the linear program given by maximsing over all points in the feasible region within (λ, μ)-space.

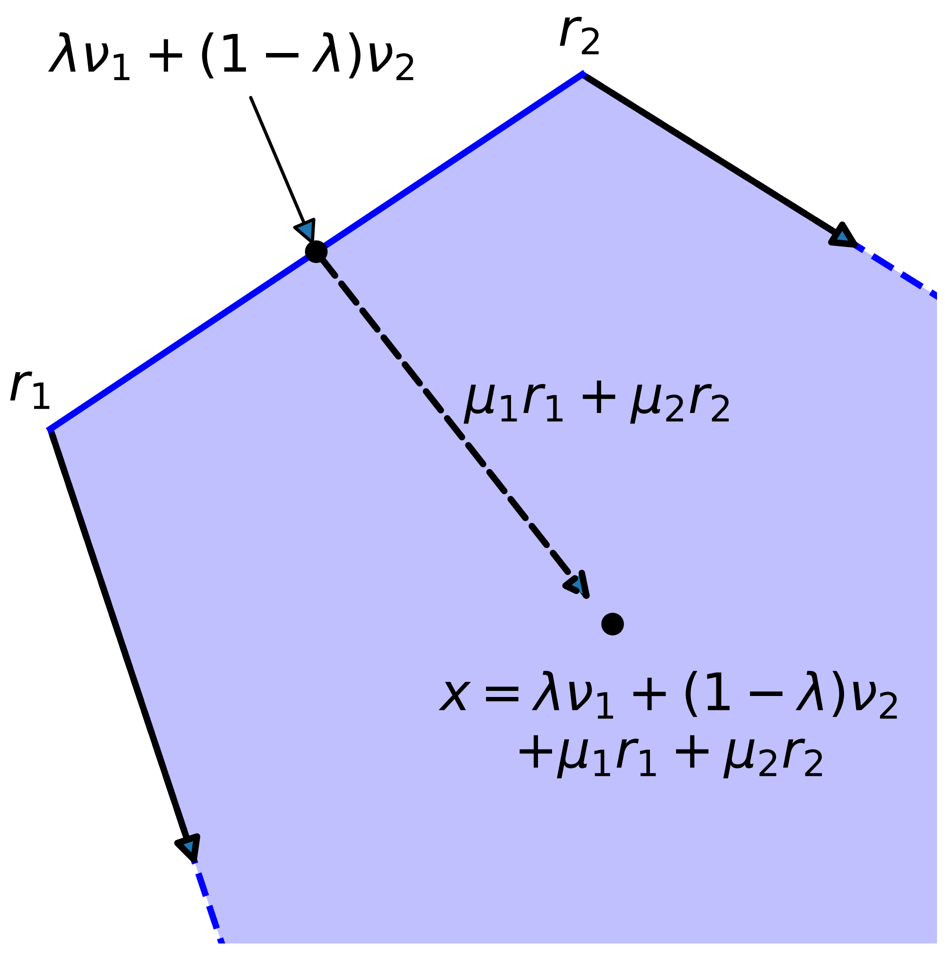
$\lambda\nu_1 + (1-\lambda)\nu_2$
r_2
r_1
$\mu_1 r_1 + \mu_2 r_2$
$x = \lambda\nu_1 + (1-\lambda)\nu_2$
$+\mu_1 r_1 + \mu_2 r_2$

Figure 2.12: The Weyl-Minkowski theorem says that we can express every point within a polyhedron in terms of some set of vertices (here ν_1, ν_2) and extreme rays (here r_1, r_2).

Now look at the objective function. If $cr_j > 0$ for some ray r_j we can increase μ_j, and hence the objective function, arbitrarily. This means the linear program is unbounded.

The alternative is that $cr_j \leq 0$ for all rays. In this case the optimal solution requires $\mu_j = 0$ for all j, as any non-zero coefficient μ_j would only reduce the objective function. Therefore

$$cx^* = \sum_{i=1}^{k} \lambda_i (cv_i) \tag{2.31}$$

and the optimal value has to be a linear combination of the values taken by the objective function at vertices. This gives us a finite number of values to check, and we conclude that

$$cx^* = \max_{i=1,\dots,k} \{cv_i\}. \tag{2.32}$$

There is, then, an optimal solution corresponding to a vertex v_i. □

2.4.6 Sanity checks

Interior points cannot be optimal

An alternative negative proof can show that interior points cannot be optimal. This does not show the link between vertices and optimal points, but gives more confidence. An illustration is in figure 2.13.

Theorem 2.4.6. *Any point x in the interior of P cannot be optimal.*

Proof. Work by contradiction. Assume there is a point x in the interior of P that is optimal. Construct a ball B of radius $\delta > 0$ around x, $B = \{y \colon \|y - x\| \leq \delta\}$. We need $B \subseteq P$ so that all points are within the feasible region.

Now consider a point $w = x + \epsilon c$ where ϵ is small enough that $w \in B \subseteq P$. As c is the gradient of the objective function it must be the case that (for maximisation problems) the objective function at w is greater than at x. Therefore x is not optimal and we have a contradiction. □

Non-vertices may be optimal

Note that it is possible for *multiple* vertices to have the same, optimal, value of the objective function. In this case points within the facet (in its *relative interior*) linking these vertices will have the same optimal value. This may occur when the facet is orthogonal to the gradient of the objective function, c.

However, this does not matter. The theorem guarantees that *an* optimal solution can always be found at *some* vertex. In this case there are infinite optimal solutions obtained by convex combinations of the vertices of the facet, and all of the vertices are optimal. This is illustrated in figure 2.14.

2.4.7 Brute force solutions

The fundamental theorem therefore gives us a method to find the optimal solution of a linear program. That is, we first identify all vertices of the feasible region. We then evaluate the objective region at each vertex. By enumeration we then find which is optimal.

For example, consider the problem from the graphical solution section,

$$\begin{array}{lrcrcl} \text{max} & 0.02x_A & + & 0.03x_B & & \\ \text{subject to} & x_A & + & x_B & \leq & 10 \\ & x_A & & & \leq & 7 \\ & & & x_B & \leq & 5 \\ & x_A, & & x_B & \geq & 0. \end{array} \tag{2.33}$$

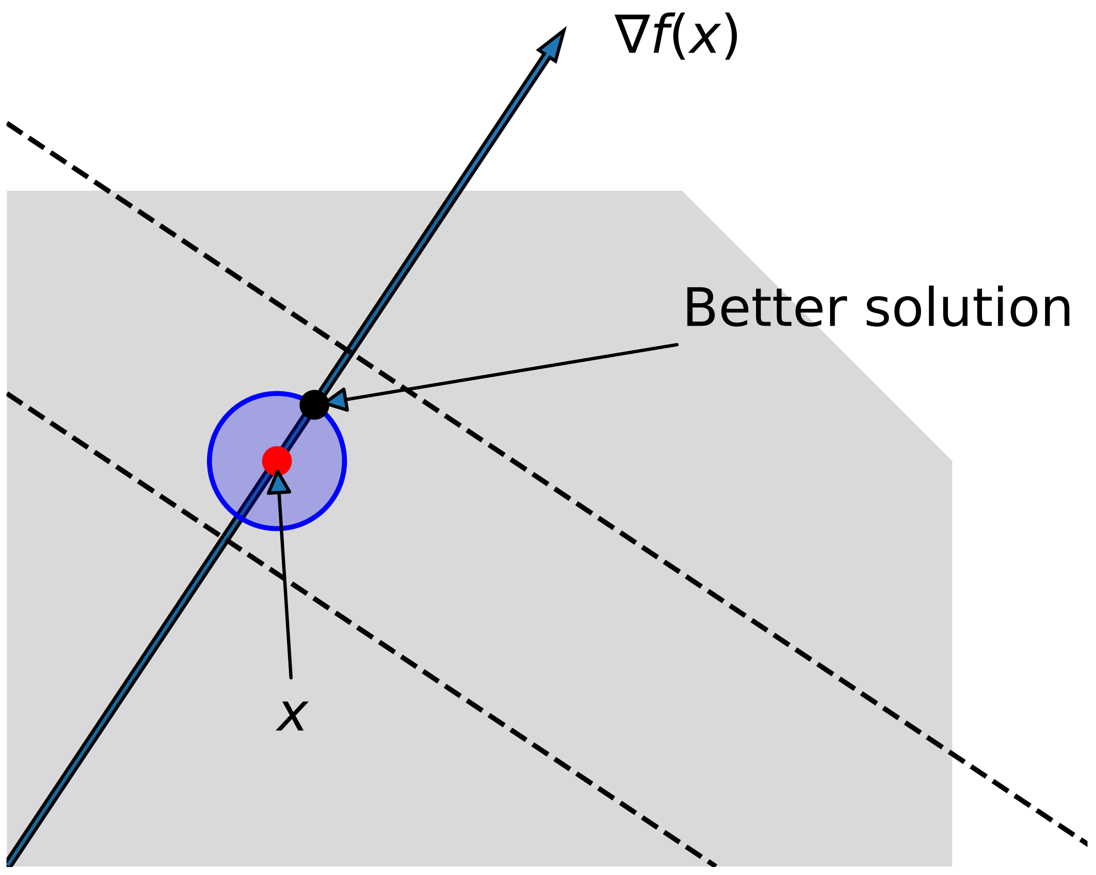

Figure 2.13: An interior point (red dot) of the feasible region (grey shaded region) cannot be optimal, as we can always construct a ball (blue shaded region) around it and improve the value of the objective function (whose level curves are the dashed lines) by moving along ∇f (the arrow).

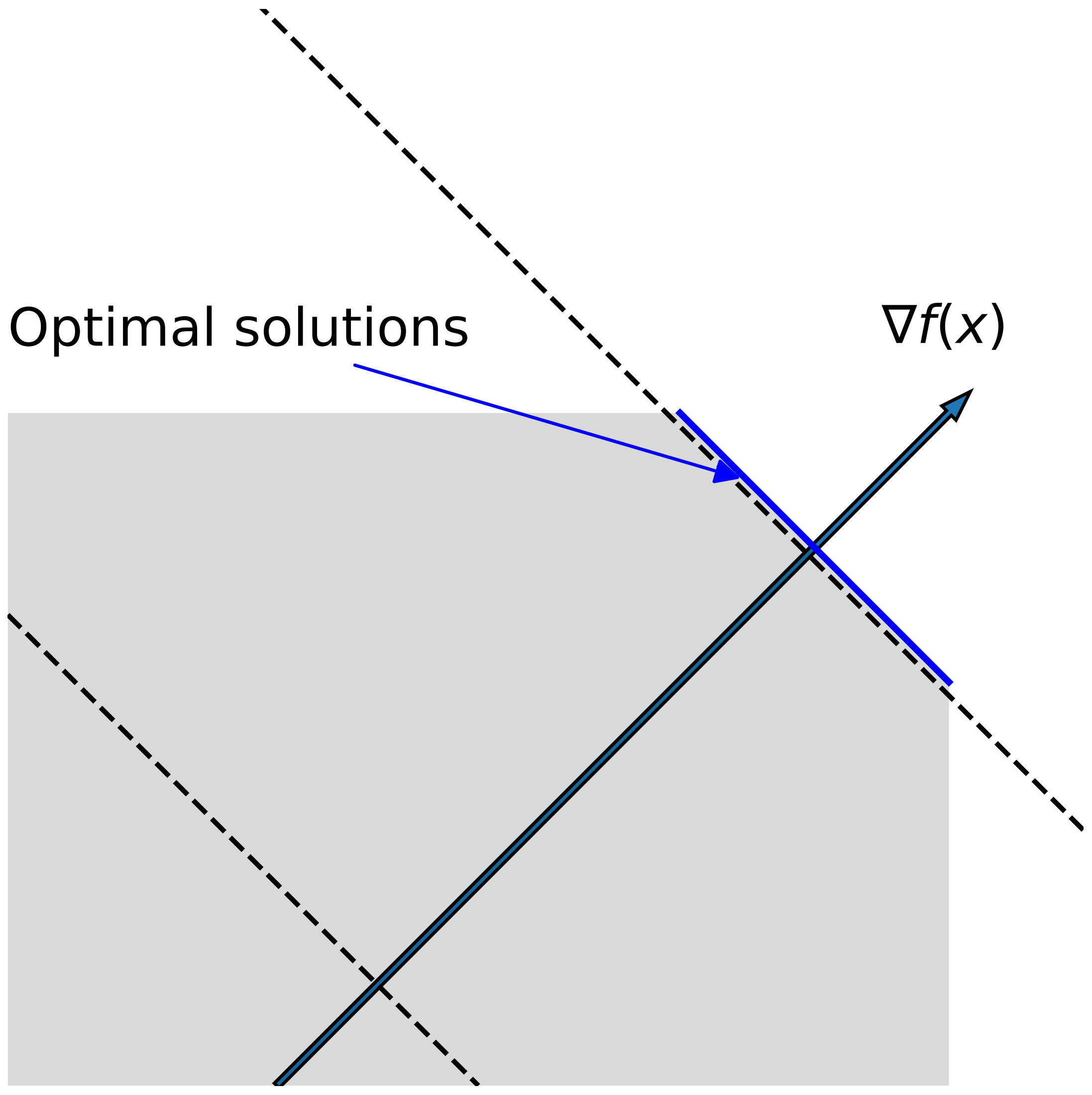

Figure 2.14: Points in the relative interior of a facet (the blue line) can be optimal if the level curve (dashed line) is constant within the facet so that ∇f is orthogonal to it. The value of the objective function is then the same at all points in the facet. If it is optimal, the vertices of the facet are then optimal (as is every point in the facet).

This has five vertices from the intersections of the four constraints.

In general we expect a feasible region with n variables and m constraints to be represented by a polyhedron with approximately

$$\binom{m}{n} = \frac{m!}{n!(m-n)!} \simeq \frac{m^m}{n^n (m-n)^{m-n}} \tag{2.34}$$

vertices. The geometric growth of the number of vertices make enumeration impractical, even with modern computers.