MAT5OPT Subject Notes

Contents

1. Constrained and unconstrained optimisation 1

1.1 Introduction to optimisation . . . 2

1.2 Minimisers . . . 6

1.3 First-order conditions for local minimisers . . . 12

1.4 Second-order conditions for local minimisers . . . 18

2. The method of least squares 23

2.1 Linear regression . . . 24

2.2 Systems of linear equations . . . 28

2.3 Nonlinear regression: polynomials and linear combinations . . . 35

2.4 Linearisation . . . 39

3. Iterative search methods 43

3.1 Limitations of algebra and calculus . . . 44

3.2 Iterative algorithms . . . 47

3.3 The golden section and Fibonacci methods . . . 49

3.4 Root-finding methods . . . 53

3.5 Line search methods and gradient methods . . . 57

3.6 The downhill simplex method . . . 60

3.7 Regression revisited . . . 64

4. Linear programming 71

4.1 Introduction to linear programming . . . 72

4.2 The corner point theorem . . . 75

4.3 Basic solutions . . . 78

4.4 The simplex algorithm . . . 84

4.5 Non-uniqueness and degeneracy . . . 89

4.6 The 2-phase method . . . 92

5. Nonlinear optimisation 97

5.1 An introductory example . . . 98

5.2 Tangent and normal spaces . . . 101

5.3 The method of Lagrange multipliers 108
5.4 Nonlinear optimisation with inequality constraints 115
5.5 Support vector machines 123

Bibliography 127

Index 128

Chapter 1

Constrained and unconstrained optimisation

1.1 Introduction to optimisation

Let's start with an example. Suppose we have an amount of money, M, which we can spend on two materials. We spend x_1 on material 1 and x_2 on material 2, and aim to maximise the total weight of material. If w_i denotes the unit weight of material i, then:

- we wish to maximise $w_1x_1 + w_2x_2$ (this is the total weight of material),
- we require $x_1 + x_2 \leqslant M$ (because we cannot spend more money than we have), and
- we require $x_1, x_2 \geqslant 0$ (because we will be buying a positive amount of material).

This is more conveniently described using vectors.

Background: vectors

Let $n \in \mathbb{N}$ be a natural number (i.e., a positive whole number). A **vector** (in $\mathbb{R}^n$) is an $n \times 1$ column matrix. If

$$\mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix},$$

is a vector in $\mathbb{R}^n$, then the real numbers $v_1, v_2, \ldots, v_n$ are the **coordinates** of $\mathbf{v}$. Note that vectors are denoted in writing by using boldface letters. Non-bold characters usually represent **scalars**, which are elements of $\mathbb{R}$.

It is not always convenient to write column matrices, so we often write vectors in terms of the transpose operation applied to row matrices, so that the following are two equivalent ways of writing the same thing:

$$\mathbf{x} = (x_1, x_2, x_3)^T \iff \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

This convention arises largely because it takes less space to write "Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$".

Mathematically, we write the example above as follows:

$$\begin{aligned} \text{maximise} \quad & f(x_1, x_2) = w_1x_1 + w_2x_2 \\ \text{subject to} \quad & x_1 + x_2 - M \leqslant 0 \\ & \qquad x_1, x_2 \geqslant 0. \end{aligned}$$

Alternatively, we can express this in terms of functions taking vectors as inputs:

$$\begin{aligned} \text{maximise} \quad & f(\mathbf{x}) = w_1x_1 + w_2x_2 \\ \text{subject to} \quad & g(\mathbf{x}) \leqslant 0 \\ & x_1, x_2 \geqslant 0, \end{aligned}$$

where

- $f\colon \mathbb{R}^2 \to \mathbb{R}$ is given by $f(\mathbf{x}) = w_1x_1 + w_2x_2$, and
- $g\colon \mathbb{R}^2 \to \mathbb{R}$ is given by $g(\mathbf{x}) = x_1 + x_2 - M$.

Note that f and g are real-valued functions (i.e., they output a single real number).

Background: functions and operations involving vectors

For each n, the space $\mathbb{R}^n$ is a **vector space**, which means we have vector addition and scalar multiplication. **Vector addition** is defined coordinatewise; if $\mathbf{u} = (u_1, u_2, \dots, u_n)^T$ and $\mathbf{v} = (v_1, v_2, \dots, v_n)^T$, then

$$\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2, \dots, u_n + v_n)^T.$$

The operation is commutative and associative, there is a zero vector $\mathbf{0} = (0, 0, \dots, 0)^T$, and there is an additive inverse, $-\mathbf{u} = (-u_1, -u_2, \dots, -u_n)^T$.

For a scalar $a \in \mathbb{R}$, we define **scalar multiplication** coordinatewise as well:

$$a\mathbf{u} = (au_1, au_2, \dots, au_n)^T.$$

This operation is distributive and associative. Note that $(-1)\mathbf{u} = -\mathbf{u}$, $0\mathbf{u} = a\mathbf{0} = \mathbf{0}$, and $1\mathbf{u} = \mathbf{u}$.

The Euclidean **norm**, or **length**, of a vector $\mathbf{u}$ is

$$\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2 + \ldots + u_n^2}.$$

The **distance** between $\mathbf{u}$ and $\mathbf{v}$ is given by $\|\mathbf{u} - \mathbf{v}\|$.

A **vector-valued function** (or **vector function** for short) is a function that gives outputs in $\mathbb{R}^n$, for some value of n. Typically, the function is expressed using boldface letters. For example, the function $\mathbf{h}\colon \mathbb{R}^2 \to \mathbb{R}^3$ given by

$$\mathbf{h}(\mathbf{x}) = \begin{pmatrix} x_1 + x_2 \\ x_1^2 + x_2^2 \\ x_2 - x_1^2 \end{pmatrix}$$

is a vector-valued function, but can also be viewed as three separate scalar functions:

$$h_1(\mathbf{x}) = x_1 + x_2, \quad h_2(\mathbf{x}) = x_1^2 + x_2^2, \quad h_3(\mathbf{x}) = x_2 - x_1^2.$$

Below, we will see that it is often convenient to express multiple scalar functions as a single vector function instead.

The basic **set-constrained optimisation** problem we consider in MAT5OPT is to:

$$\begin{aligned} \text{minimise} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \Omega. \end{aligned} \tag{1.1}$$

We take f to be a real-valued function $f\colon \mathbb{R}^n \to \mathbb{R}$, called the **objective function**. The set $\Omega \subset \mathbb{R}^n$ is called the **feasible set** or **feasible region**. The variables $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ are the **decision variables**. It is sufficiently general to consider minimisation problems: if you have a function g which needs to maximised, you can minimise $-g$ instead.

Typically, the feasible set is defined in terms of one or more **constraints**. A rather general form of (1.1) takes the following form:

$$\begin{aligned} \text{minimise} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}) \leqslant \mathbf{0}. \end{aligned} \tag{1.2}$$

The vector-valued constraint $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ comprises multiple *equality* constraints, and the vector-valued constraint $\mathbf{g}(\mathbf{x}) \leqslant \mathbf{0}$ comprises multiple *inequality* constraints.

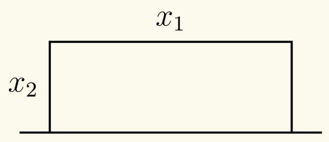
Let's now make this concrete by exploring some specific examples.

Example: a classic fencing problem

You want to enclose a rectangular field using 800 metres of fencing. A pre-existing fence on an adjacent field means that one side of the field will not need new fencing. Determine the size of the field that will enclose the largest area.

Since we want the *largest* possible area, this is a maximisation problem.

The variables are the side lengths of the rectangular region. Let x_1 be the length of the fence that is parallel to the existing fence, and let x_2 be the length of the other two sides, as shown:



Now, the objective function. The quantity we wish to maximise is the area of the rectangular region. The area of this region is x_1x_2, so the objective function is $f(\mathbf{x}) = x_1x_2$.

Finally, the constraints. The total fencing used is $x_1 + 2x_2$, and we can use up to 800 metres. This gives the constraint $x_1 + 2x_2 \leqslant 800$. Moreover, we cannot use a negative amount of fencing, so we will have $x_1 \geqslant 0$ and $x_2 \geqslant 0$.

So we can express the problem as follows:

$$\begin{aligned} \text{maximise} \quad & f(x_1, x_2) = x_1x_2 \\ \text{subject to} \quad & x_1 + 2x_2 \leqslant 800 \\ & \quad x_1, x_2 \geqslant 0. \end{aligned}$$

Note that the constraints could be equivalently expressed using a single vector-valued function. Let $\mathbf{g}\colon \mathbb{R}^2 \to \mathbb{R}^3$ be given by

$$\mathbf{g}(x_1, x_2) = \begin{pmatrix} x_1 + 2x_2 - 800 \\ -x_1 \\ -x_2 \end{pmatrix}.$$

Then the problem can be expressed as

$$\begin{aligned} \text{maximise} \quad & f(\mathbf{x}) = x_1x_2 \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leqslant \mathbf{0}, \end{aligned}$$

which matches the general form shown in (1.2).

Alternatively, if the farmer was to use *exactly* 800 metres of fencing, the first constraint would be best expressed as $x_1+2x_2 = 800$. This, too, would match the general form, by defining $h(\mathbf{x}) = x_1+2x_2-800$ and $\mathbf{g}(\mathbf{x}) = (-x_1, -x_2)^T$.

As for solving this example, it can be addressed using high school calculus techniques (giving $\mathbf{x} = (400, 200)$), although this is not a luxury afforded to all problems of the form (1.2).

Example: machine manufacturing time

This example is based on an example in Calvert and Voxman's *Linear Programming* [2].

A machine shop produces two products, each requiring manufacturing time on 3 machines. Each product has a fixed profit associated with it. The total available time on each of the three machines is limited. The situation is represented in a table:

hours per machine per item			
machine type	product I	product II	total hours available per machine per week
A	2	1	70
B	1	1	40
C	1	3	90
profit per item	\$40	\$60	

The decision variables are the number of units of each product produced. Let x_1 and x_2 denote the number of units of product I and product II respectively produced each week.

The objective function is the profit, which we wish to maximise:

$$f(\mathbf{x}) = 40x_1 + 60x_2$$

The constraints are linear inequalities:

$$\begin{aligned} 2x_1 + x_2 &\leqslant 70 && \text{available time on machine A} \\ x_1 + x_2 &\leqslant 40 && \text{available time on machine B} \\ x_1 + 3x_2 &\leqslant 90 && \text{available time on machine C} \\ x_1 &\geqslant 0 && \text{cannot produce a negative number of items} \\ x_2 &\geqslant 0 && \text{cannot produce a negative number of items} \end{aligned}$$

Collating this information together into the optimisation problem gives:

$$\begin{aligned} \text{maximise} \quad & f(\mathbf{x}) = 40x_1 + 60x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \leqslant 70 \\ & x_1 + x_2 \leqslant 40 \\ & x_1 + 3x_2 \leqslant 90 \\ & x_1, x_2 \geqslant 0. \end{aligned}$$

This is an example of a *linear programming problem*, because the objective function and constraints are all expressed in terms of linear functions of the decision variables. We will learn about linear programming in more depth in Chapter 4.

1.2 Minimisers

Let's start by taking a look at **unconstrained optimisation**, which takes the form

$$\text{minimise } f(\mathbf{x}), \tag{1.3}$$

i.e., there are no constraints.

You may already be familiar with applying calculus techniques to optimisation in one variable. Consider a differentiable function $f\colon \mathbb{R} \to \mathbb{R}$. A point x_0 is a **stationary point** if $f'(x_0) = 0$. The **second derivative test** tells us that for such a point:

- if $f''(x_0) > 0$, then x_0 gives a local minimum, and
- if $f''(x_0) < 0$, then x_0 gives a local maximum, and
- if $f''(x_0) = 0$, then the test is inconclusive (the point x_0 may give a local minimum, local maximum, or point of inflection).

Example: calculus refresher

Consider the function $f\colon \mathbb{R} \to \mathbb{R}$ given by $f(x) = x^3 - 12x^2 + 9x + 3$. Then $f'(x) = 3x^2 - 24x + 9$ and $f''(x) = 6x - 24$. Solving $f'(x) = 0$ gives $x = 4 \pm \sqrt{13}$. These are the candidate points, and we now apply the second derivative test.

We have $f''\left(4 \pm \sqrt{13}\right) = \pm 6\sqrt{13}$, so,

- when $x = 4 + \sqrt{13}$, there is a local minimum (as $f''(4 + \sqrt{13}) > 0$), and
- when $x = 4 - \sqrt{13}$, there is a local maximum (as $f''(4 - \sqrt{13}) < 0$).

In the language we are about to meet, we would describe this by saying $4 + \sqrt{13}$ is a *local minimiser* of f and $4 - \sqrt{13}$ is a *local maximiser* of f.

For functions of two variables $f(x, y)$, the situation is similar. For such a function, the stationary points are those points at which both partial derivatives equal zero. Then, the following table can be used to classify its stationary points.

$f_{xx}f_{yy} - (f_{xy})^2$ negative $\longrightarrow$	saddle point (see Figure 1.1)
$f_{xx}f_{yy} - (f_{xy})^2$ positive $\longrightarrow$	f_{xx} positive $\longrightarrow$ minimum
	f_{xx} negative $\longrightarrow$ maximum

Here, f_{xx} is shorthand for $\frac{\partial^2 f}{\partial x^2}$, f_{yy} for $\frac{\partial^2 f}{\partial y^2}$, and f_{xy} for $\frac{\partial^2 f}{\partial y \partial x}$.

To exhibit the general case, we will first formally define the main concepts. Recall the general form

$$\begin{array}{rl} \text{minimise} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in \Omega \end{array}$$

and recall that Ω is called the feasible set. A vector in the feasible set is called a **feasible vector**.

We will deal with minimisers, but similar notions can be defined for maximisers. Collectively, minimisers and maximisers are called **extremisers**.

A **global minimiser** of f over Ω is a feasible vector $\mathbf{x}^*$ for which the value of the function is the smallest possible, i.e.,

$$(\forall \mathbf{x} \in \Omega)\ f(\mathbf{x}) \geqslant f(\mathbf{x}^*).$$

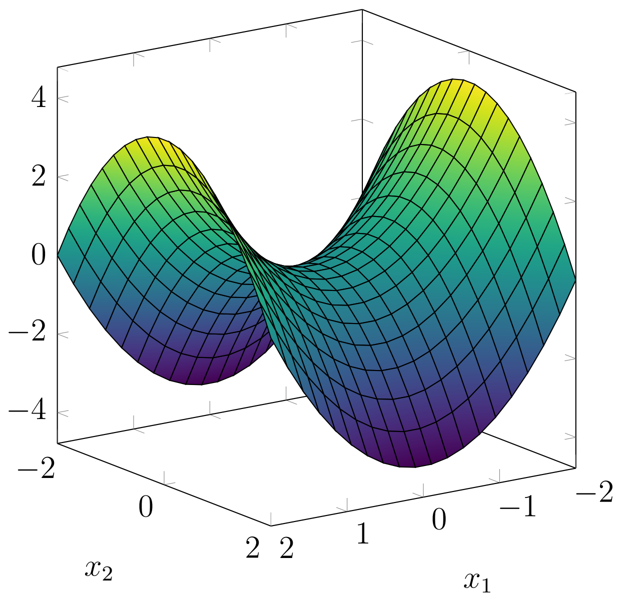


Figure 1.1: The function $x_1^2 - x_2^2$ has a saddle point at $(0, 0)$.

Note that there may exist several global minimisers. If $\mathbf{x}^*$ is a global minimiser of f over Ω, we write

$$f(\mathbf{x}^*) = \min_{\mathbf{x}\in\Omega} f(\mathbf{x}), \text{ and } \mathbf{x}^* \in \arg\min_{\mathbf{x}\in\Omega} f(\mathbf{x}).$$

Observe the differences in the two: min describes the smallest value of the function, whereas arg min describes the set of inputs which yield that minimum.

A **global maximiser** of f over Ω is defined analogously, as are the operators max and arg max.

Background: universal quantification

The symbol $\forall$ is called a universal quantifier, and represents a "for all" statement. When this symbol is used, we are saying that a particular property or statement holds true for every element in a given set. For example, the statement $(\forall x \in S)\ P(x)$ can be read as "for every element x in the set S, the property $P(x)$ holds true". In other words, it says that the statement "$P(x)$" is true for all elements in the set S.

So, for the statement $(\forall \mathbf{x} \in \Omega)\ f(\mathbf{x}) \geqslant f(\mathbf{x}^*)$ above, we can read this as, "for every element $\mathbf{x}$ in the feasible set, $f(\mathbf{x})$ is greater than or equal to $f(\mathbf{x}^*)$", which describes the property of $f(\mathbf{x}^*)$ being minimal.

If we have uniqueness, i.e., if there exists $\mathbf{x}^*$ such that

$$(\forall \mathbf{x} \in \Omega)\ \mathbf{x} \neq \mathbf{x}^* \implies f(\mathbf{x}) > f(\mathbf{x}^*),$$

then we write

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Omega} f(\mathbf{x})$$

and say that $\mathbf{x}^*$ is a **strict global minimiser**.

To properly define a local minimiser, we need to work with neighbourhoods. Recall that $\|\mathbf{u} - \mathbf{x}\|$ is the distance between the vectors $\mathbf{u}$ and $\mathbf{x}$. The set

$$\mathcal{B}_\varepsilon(\mathbf{x}) = \{\mathbf{u} : \|\mathbf{u} - \mathbf{x}\| < \varepsilon\},$$

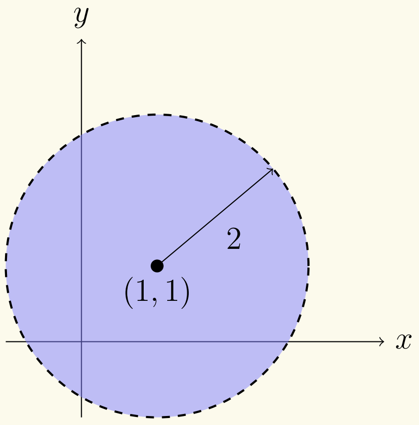
with $\varepsilon > 0$, is called the **ball** with radius ε and centre $\mathbf{x}$, and it forms a **neighbourhood** of $\mathbf{x}$.

Example: balls in 2D space

In two-dimensional space, a ball of radius ε corresponds exactly to the notion of an open disc. For example, consider $\mathcal{B}_2\left((1,1)\right)$. By definition,

$$\mathcal{B}_2\left((1,1)\right) = \{\mathbf{u} : \|\mathbf{u} - (1,1)\| < 2\}$$

In words, this is the set of points which have distance *smaller than 2* from the point (1,1):



Likewise, in $\mathbb{R}^3$, a ball represents an open sphere, and in $\mathbb{R}$, a ball is an open interval.

A **local minimiser** is a feasible vector $\mathbf{x}^*$ for which the value of the function is the smallest in some neighbourhood, i.e.,

$$(\exists \varepsilon \in \mathbb{R})(\forall \mathbf{x} \in \mathcal{B}_\varepsilon(\mathbf{x}^*) \cap \Omega)\ f(\mathbf{x}) \geqslant f(\mathbf{x}^*)$$

Moreover, if

$$(\exists \varepsilon \in \mathbb{R})(\forall \mathbf{x} \in \mathcal{B}_\varepsilon(\mathbf{x}^*) \cap \Omega)\ \mathbf{x} \neq \mathbf{x}^* \implies f(\mathbf{x}) > f(\mathbf{x}^*),$$

then $\mathbf{x}^*$ is called a **strict local minimiser**.

Background: existential quantification

The symbol $\exists$ is called an existential quantifier, and represents a "there exists" statement. When this symbol is used, we are saying that there is at least one element in a given set that has a particular property or satisfies a certain condition. For example, the statement $(\exists x \in S)\ P(x)$ can be read as "there exists at least one element x in the set S for which the property $P(x)$ holds true". In other words, it says that the statement "$P(x)$" is true for at least one element in the set S.

For the statement above,

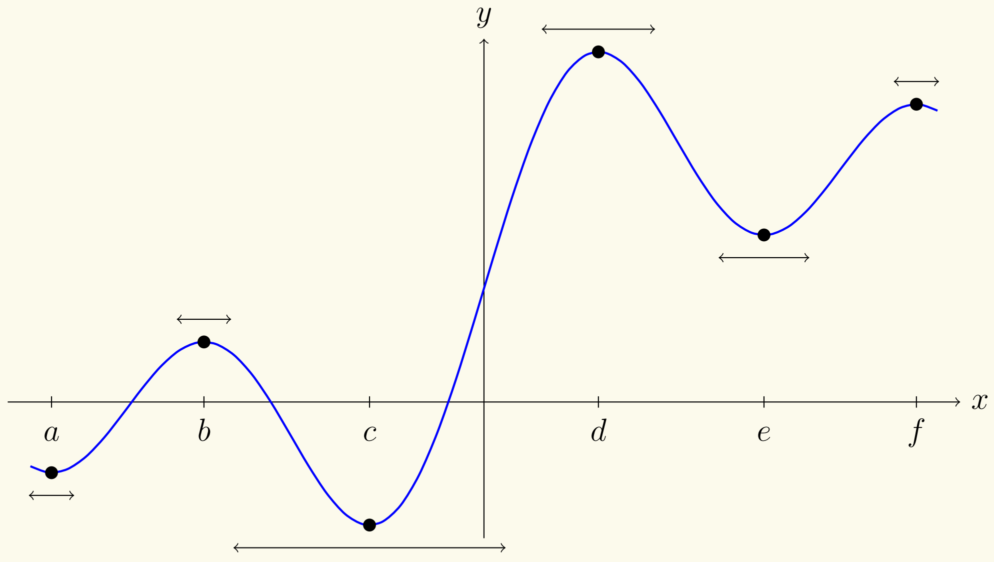
$$(\exists \varepsilon \in \mathbb{R})(\forall \mathbf{x} \in \mathcal{B}_\varepsilon(\mathbf{x}^*) \cap \Omega)\ f(\mathbf{x}) \geqslant f(\mathbf{x}^*),$$

the existential quantifier $(\exists \varepsilon \in \mathbb{R})$ describes the size of the ball $\mathcal{B}_\varepsilon(\mathbf{x}^*)$. Combined with the universal quantifier $(\forall \mathbf{x} \in \mathcal{B}_\varepsilon(\mathbf{x}^*))$, it can be read as: *there exists* a ball of radius ε centred at $\mathbf{x}^*$ such that, *for every* element $\mathbf{x}$ in that ball (and also in the feasible region), the value of $f(\mathbf{x})$ is greater than or equal to $f(\mathbf{x}^*)$.

More succinctly, it can be interpreted as: there exists a ball of *some radius* centered at $\mathbf{x}^*$ for which $f(\mathbf{x}^*)$ is minimal when applying f to the intersection of that ball and the feasible region.

Example: local extremisers for functions of one variable

Consider the function $f\colon \mathbb{R} \to \mathbb{R}$ depicted below.



Since a ball in $\mathbb{R}$ is an open interval, both strict and non-strict local minimisers and maximisers are defined in terms of open intervals around the point of interest.

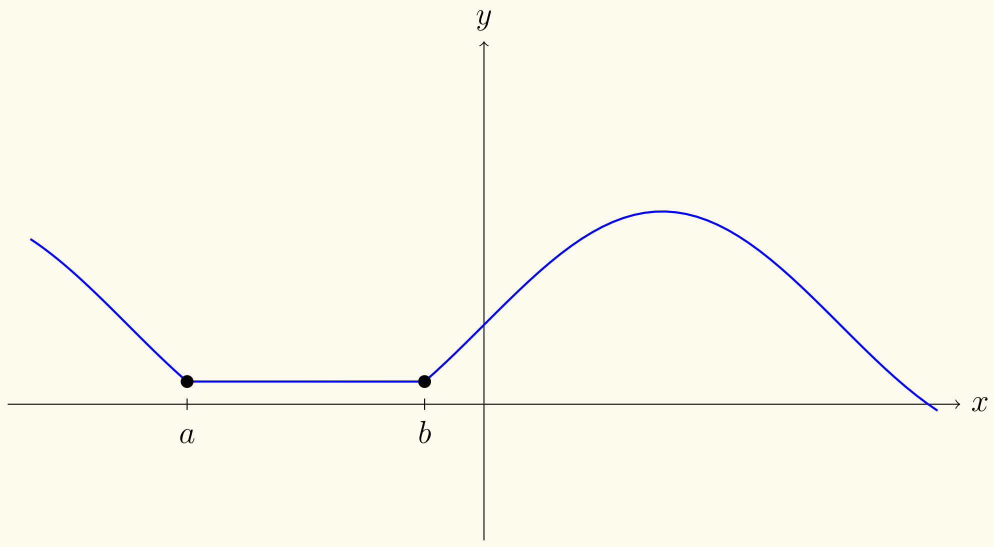
In the diagram above, suitable open intervals are indicated for each point, but note that the size of interval is not unique; in each case, some larger and any smaller intervals could have been chosen.

The points a, c and e are local minimisers, while the points b, d and f are local maximisers.

For example, consider the point e and the depicted interval. When $f(x)$ is evaluated for any x in that open interval, its value is smallest when $x = e$, so e is a local minimiser.

Moreover, all of the points a, b, c, d, e and f are strict. This is because there is an open interval for which they are the only points attaining that extremum.

An example of a function with local minimisers that are not strict is depicted below:



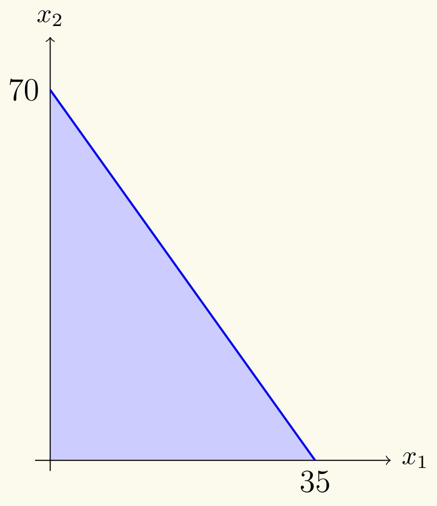
In this case, all points in the interval $[a, b]$ are local minimisers, but they are not strict local minimisers.

Example: a visual method

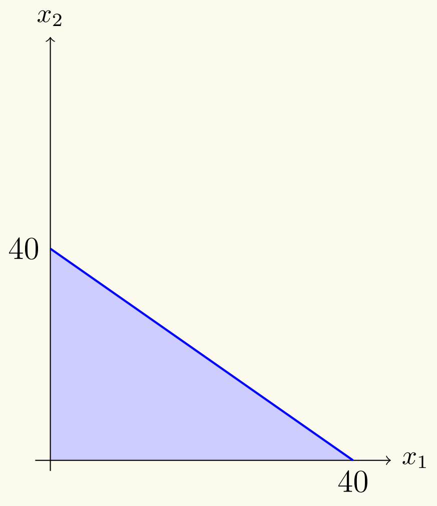
Consider again the example adapted from Calvert and Voxman's *Linear Programming* [2], where we expressed the problem of maximising profit based on limited time available on three different machines:

$$\begin{aligned} \text{maximise} \quad & f(\mathbf{x}) = 40x_1 + 60x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \leqslant 70 \\ & x_1 + x_2 \leqslant 40 \\ & x_1 + 3x_2 \leqslant 90 \\ & x_1, x_2 \geqslant 0. \end{aligned}$$

Each of these constraints defines a boundary of the feasible region. For example,

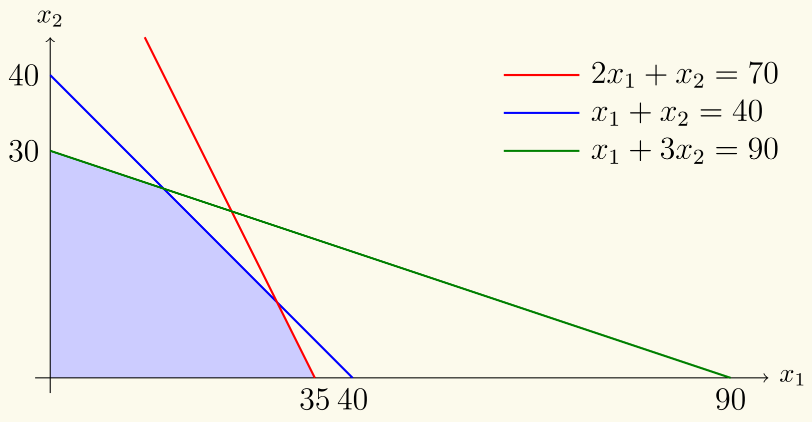


The constraint $2x_1 + x_2 \leqslant 70$ is equivalent to $x_2 \leqslant 70 - 2x_1$, which is plotted above.



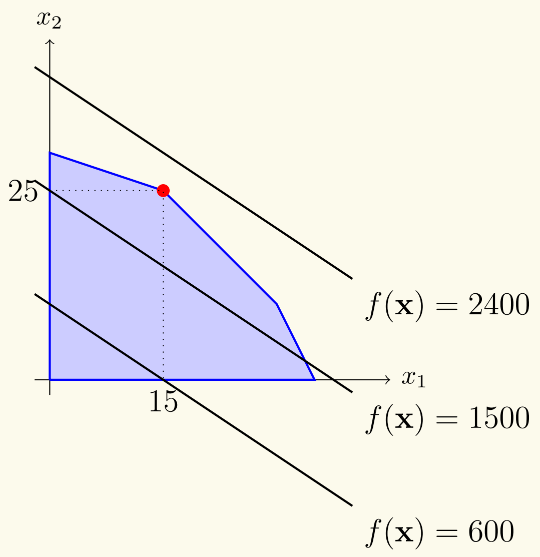
The constraint $x_1 + x_2 \leqslant 40$ is equivalent to $x_2 \leqslant 40 - x_1$, which is plotted above.

Taking the intersection of all the regions determined by each constraint yields the feasible region:



By plotting some level sets of the objective function, we can locate point(s) in the feasible region where the objective function is largest. Consider the following:

- $f(\mathbf{x}) = 600 \Leftrightarrow 40x_1 + 60x_2 = 600 \Leftrightarrow x_2 = 10 - \frac{2}{3}x_1$
- $f(\mathbf{x}) = 1500 \Leftrightarrow 40x_1 + 60x_2 = 1500 \Leftrightarrow x_2 = 25 - \frac{2}{3}x_1$
- $f(\mathbf{x}) = 2400 \Leftrightarrow 40x_1 + 60x_2 = 2400 \Leftrightarrow x_2 = 40 - \frac{2}{3}x_1$



We can observe that as the level sets of the objective function sweep across the feasible region, they leave the feasible region at one of the corner points. In this case, we can observe that the maximum is obtained at the point $P = (15, 25)^T$ and hence the maximum profit is attained by producing 15 units of product I and 25 units of product II, attaining a profit of $f(15, 25) = 40 \times 15 + 60 \times 25 = \$2100$.

For an interactive version of these diagrams, follow this URL:

`https://www.desmos.com/calculator/fua5mta6y2`

Bibliography

[1] Robert G. Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107, 1977.

[2] J.E. Calvert and W.L. Voxman. *Linear Programming*. Harcourt Brace Jovanovich, 1989.

[3] Edwin K. P. Chong and Stanislaw H. Żak. *An Introduction to Optimization*. Wiley, fourth edition, 2013.

[4] Vašek Chvátal. *Linear Programming*. W. H. Freeman and Company, New York, 1983.

[5] Alan J. Hoffman. Cycling in the simplex algorithm. *National Bureau of Standards Report No. 2974*, 1953.

Index

n-simplex, 60
2-phase method, 94

absolute tolerance, 47
active, 115
active tangent space, 119
affine, 12
artificial variable, 92
augmented matrix, 31

ball, 7
basic solution, 79
basic variables, 78
basis, 101
bisection method, 53
Bland's rule, 91
boundary point, 14
bounded, 16

canonical form, 78
centre of mass, 60
closed, 14
closed half-spaces, 75
coefficient matrix, 31
compact, 16
constraints, 3
continuously differentiable, 53
contraction step, 61
convex, 75
coordinates, 2
corner point, 75
corner point theorem, 75
cost coefficients, 74
cycling, 91

decision variables, 3
degeneracy, 91
departing basic variable, 84
derivative, 12
descent property, 59
determinant, 28
dimension, 101
directional derivative, 13
distance, 3
downhill simplex method, 62

entering basic variable, 84
expansion, 61
extreme point, 75
extremisers, 6

feasible direction, 14
feasible region, 3
feasible set, 3
feasible vector, 6
Fibonacci method, 51
Fibonacci number, 52
FONC, 14
FOSC, 16
fundamental theorem of linear algebra, 101
fundamental theorem of linear programming, 75

Gaussian elimination, 33
global maximiser, 7
global minimiser, 6
globally convergent, 59
golden ratio, 51
golden section method, 50
gradient, 12
gradient method, 58

Hessian, 18
hyperplane, 75

identity matrix, 28
image, 101
inactive, 115
inside contraction, 62
interior, 14
interior point, 14
inverse, 29
invertible, 29

Jacobian, 12

kernel, 101
KKT-multipliers, 116

Lagrangian, 110, 117
leading principal minors, 19
leading variable, 32
least squares method, 24
length, 3
line search method, 57
line segment, 75
linear, 12
linear combination, 101
linear program, 74
linear programming, 72
linearly dependent, 101
linearly independent, 101
local minimiser, 8
locally convergent, 59
locally convex, 17

margin, 123
method of steepest descent, 58
minimum ratio test, 84

negative definite, 19
negative half-space, 75
negative semi-definite, 19
neighbourhood, 7
Newton's method, 53, 58
nonsingular, 29
norm, 3
normal, 75
normal equations, 26, 35
normal space, 105

objective function, 3
open, 14
open half-space, 75
optimal separating hyperplane, 123
orthogonal complement, 101
outside contraction, 62

pivoting, 84
positive definite, 19
positive half-space, 75
positive semi-definite, 19
profit, 74

quadratic form, 19
quadratic programming, 115

rank-nullity theorem, 101
rate of change, 13
rate of convergence, 59
reduced row-echelon, 31
reflection, 61
regular point, 107, 115
residual, 24
resource values, 74
root, 53
root-finding method, 53
row leader, 31

scalar multiplication, 3
search direction, 57
secant method, 55
second derivative test, 6
separable, 123
separating hyperplane, 123
set-constrained optimisation, 3
shrink, 62
simplex algorithm, 84, 89
singular, 29
slack variable, 80
SONC, 19
SOSC, 20
span, 101
stalling, 91
standard form, 80
stationary point, 6
step size, 57
strict global minimiser, 7
strict local minimiser, 8
support vectors, 123
supporting hyperplane, 75
Sylvester's criterion, 19
symmetric, 19
system of linear equations, 30

tangent space, 105

unconstrained optimisation, 6
unimodal, 49

vector, 2
vector addition, 3
vector function, 3
vector space, 3
vector subspace, 101
vector-valued function, 3
vertex, 60