MA305 Non-linear Optimisation[1]

3. Numerical approximation

I look at numerical methods for dealing with problems of the form

find x such that $f(x) = c$ (one-dimensional),
find x maximizing/minimizing $f(x)$ (one-dimensional),
find $\mathbf{x}$ maximizing/minimizing $f(\mathbf{x})$ (many-dimensional)

where f is a real-valued function of one or more real variables. In each case it is supposed that we have effective methods of calculating $f(x)$ or $f(\mathbf{x})$, for given $x/\mathbf{x}$, to any required level of accuracy; but that the calculation may be 'expensive' (ordinarily, that just means 'slow'), and is presented as the output of an oracle with no hints as to what is really happening, so that we are searching blindly. All the methods to be discussed will assume that f is, at the very least, continuous; some will require us to be able to calculate derivatives of f directly. Note that in practice we shall wish to apply the methods to functions which are not defined everywhere on $\mathbb{R}$ or $\mathbb{R}^n$. The recipes below will mostly ignore the problems which arise if a method demands the calculation of f at a point outside its domain, but you must remember that any computer program you write for general use must include escape options to deal with this possibility.

Since the problem

$$\text{find } x \text{ such that } f(x) = c$$

can be changed at once to

$$\text{find } x \text{ such that } f(x) - c = 0,$$

and calculating $f(x) - c$ will be virtually as fast as calculating $f(x)$, the discussion below will consider only the case $c = 0$. In the same way, since the problems

$$\text{find } x \text{ maximizing } f(x), \quad \text{find } \mathbf{x} \text{ maximizing } f(\mathbf{x})$$

are identical to the problems

$$\text{find } x \text{ minimizing } -f(x), \quad \text{find } \mathbf{x} \text{ minimizing } -f(\mathbf{x}),$$

I will look only at minimization problems. I should also note here that the methods I discuss will, at best, find *local* minima; finding *global* minima is very much harder, and in my view no blind-search method can be reliable unless it is supported by a real understanding of the nature of the function f.

3A. Binary search

To solve the equation $f(x) = 0$ when we have found initial points a, b such that $a < b$ and $f(a)$, $f(b)$ have opposite signs.

(a) *The algorithm*: Set $x_0 = a$, $y_0 = b$. Given that $x_n < y_n$ and $f(x_n)$ and $f(y_n)$ have opposite signs (that is, $f(x_n)f(y_n) < 0$), set $x = \frac{1}{2}(x_n + y_n)$ and calculate $f(x)$. If $f(x)$ has the same sign as $f(x_n)$, set $x_{n+1} = x$ and $y_{n+1} = y_n$; otherwise set $x_{n+1} = x_n$ and $y_{n+1} = x$. Continue.

What if $f(x) = 0$? There is a theoretical possibility that we may come to a point $x = \frac{1}{2}(x_n + y_n)$ such that $f(x)$ is either exactly zero, or cannot be distinguished from zero by the method we have for calculating f. In this case, the algorithm should certainly be halted; if possible, you should investigate whether there is good reason to believe that $f(x)$ is indeed exactly zero.

When should you stop? This will depend on the context of the problem. In some cases, you will be satisfied when you have found an x such that $|f(x)| \le \epsilon$ for some pre-determined $\epsilon > 0$. In other cases, you will be satisfied when you have a pair x_n, y_n such that $0 < y_n - x_n \le \delta$ for some pre-determined $\delta > 0$.

When does the method work? Provided f is continuous and defined on the whole initial interval $[a, b]$, the algorithm will halt in finitely many steps for either stopping criterion.

[1] This course was taught without proofs. I devised the proofs in Chapter 4 here just to check that I was getting the theorems right. If you wish to use them for a class at this level you may need to refine the details. In addition, I had second thoughts when writing parts of Chapter 3, and they don't perfectly match the problem sets.

Typeset by $\mathcal{AMS}$-TEX

(b) *Advantages of the method* (i) It works for all continuous functions f, provided we can find initial values a and b for which $f(a)f(b) < 0$ and f is defined on the whole interval $[a, b]$.

(ii) The values $f(x)$ need be calculated only to sufficient accuracy to determine whether they are positive or negative. (Remember that in many cases we shall be calculating $f(x)$ by a numerical method of the kind described in this chapter. So it may be very much cheaper to calculate $f(x)$ to two decimal places rather than four.)

(iii) When the algorithm halts (other than in the exceptional case in which we have found an x for which we cannot distinguish $f(x)$ from zero), we shall have an explicit interval-of-uncertainty $[x_n, y_n]$ with a guarantee that there is at least one solution in that interval.

(iv) If our stopping criterion is that the interval-of-uncertainty $[x_n, y_n]$ should be of length at most δ, then we can say at the start how many times we shall have to calculate f, since $y_n - x_n = 2^{-n}(b-a)$ for every n, so that we shall need exactly $\lceil \log_2 \frac{b-a}{\delta} \rceil$ further calculations after finding $f(a)$ and $f(b)$.

Disadvantage of the method The 'geometric' reduction, by which the interval-of-uncertainty $y_n - x_n$ halves each time we calculate a new value of f, is slow by the standards of numerical methods in general.

3B False position (one-dimensional case)

To solve the equation $f(x) = 0$ starting from distinct initial points a, b.

(a) *The algorithm*: Set $x_0 = a$, $x_1 = b$. Given $x_n \neq x_{n+1}$, set

$$x_{n+2} = \frac{x_n f(x_{n+1}) - x_{n+1} f(x_n)}{f(x_{n+1}) - f(x_n)};$$

continue.

The picture: imagine a straight line drawn through the points $(x_n, f(x_n))$ and $(x_{n+1}, f(x_{n+1}))$ on the graph of f; this crosses the horizontal axis at x_{n+2}.

What if $f(x_{n+1}) = f(x_n)$? There is a theoretical possibility that we may come to points x_n, x_{n+1} such that $f(x_n)$ and $f(x_{n+1})$ are either equal, or cannot be distinguished from each other by the method we have for calculating f. In this case, the algorithm must be abandoned.

When should you stop? Normally you use this method only when you expect to be satisfied when you have found an x such that $|f(x)| \leq \epsilon$ for some pre-determined $\epsilon > 0$.

There is however a serious danger that the method will break down altogether, with the x_n moving apart rather than together as the algorithm proceeds. As a general rule, therefore, you should be willing to abandon the method whenever $|f(x_{n+2})| \geq |f(x_{n+1})|$.

(b) *Advantages of the method* (i) When f is differentiable, and there is a solution x with $f(x) = 0 \neq f'(x)$, we can hope that the values $f(x_n)$ will converge rapidly to zero; typically, when the method is successful, we find that, for large n, $|f(x_n)|$ is of the order of $\alpha \exp(-\beta n^{(1+\sqrt{5})/2})$ for some α, $\beta > 0$. In contrast, for similar functions, binary search will make $|f(x_n)|$ about $\alpha \exp(-n \ln 2)$.

(ii) Although the method cannot be expected to work for non-differentiable functions f, it does not rely on our being able to calculate the derivative of f.

Disadvantages of the method (i) It is rarely effective unless there is a solution x for which f is differentiable at x.

(ii) Even if f is very well behaved, if we start from the 'wrong' points x_0 and x_1 the method can fail.

3C Newton's method (one-dimensional case)

To solve the equation $f(x) = 0$ starting from an initial point a.

(a) *The algorithm*: Set $x_0 = a$. Given x_n, set

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)};$$

continue.

The picture: imagine a tangent to the graph of f drawn through the point $(x_n, f(x_n))$; this crosses the horizontal axis at x_{n+1}.

What if $f'(x_n) = 0$? There is a theoretical possibility that we may come to a point x_n such that $f'(x_n)$ is either zero, or cannot be distinguished from zero by the method we have for calculating f'. In this case, the algorithm must be abandoned.

When should you stop? Normally you use this method only when you expect to be satisfied when you have found an x such that $|f(x)| \leq \epsilon$ for some pre-determined $\epsilon > 0$.

As with the method of false position, the method can break down altogether, with the x_n moving apart rather than together as the algorithm proceeds. You should therefore be willing to abandon the method if you ever find that $|f(x_{n+1})| \geq |f(x_n)|$.

(b) *Advantage of the method* When f is differentiable, and there is a solution x with $f(x) = 0 \neq f'(x)$, we can hope that the values $f(x_n)$ will converge rapidly to zero.

Disadvantages of the method (i) It requires calculation of $f'(x)$ as well as $f(x)$.

(ii) Even if f is very well behaved, if we start from the 'wrong' point a the method can fail.

(iii) Each step of the algorithm requires two calculations, since we need to know both $f(x_n)$ and $f'(x_n)$. Accordingly, even though the number of *iterations* is generally fewer (for a given $\epsilon > 0$) than it is for the method of false position, the number of *calculations* is generally higher. For this reason, the method can be recommended only when it is actually quicker to calculate f' than to calculate f. (This may be the case if $f(x)$ is defined as an integral $\int_0^x g(t)dt$, so that $f'(x) = g(x)$ but we need a numerical integration to find $f(x)$.)

3D Minimization in one dimension If we have a differentiable real function f, with an effective method of calculating its derivative, then the best route to a (local) minimum of f is likely to be to solve the equation $f'(x) = 0$. If you seek to do this with binary search, remember that you want f' to be negative on the left and positive on the right, so you must make sure that you start with $f'(a) < 0$ and $f'(b) > 0$; this will necessarily (provided f' is defined and continuous in $[a, b]$) lead to a local minimum of f. If you are using the method of false position or (when f is twice differentiable) Newton's method, again remember to check that $f'(x_{n+1}) - f'(x_n)$ always has the same sign as $x_{n+1} - x_n$, since otherwise you may be chasing a maximum rather than a minimum.

I give two further methods which do not depend on calculation of f'.

3E Three-point method

To find a local minimum of f, starting from points a, b, c such that $a < b < c$ and $f(b) < \min(f(a), f(c))$.

(a) *The algorithm*: Set $x_0 = a$, $y_0 = b$, $z_0 = c$. Given that $x_n < y_n < z_n$ and $f(y_n) < \min(f(x_n), f(z_n))$, take the longer of the two intervals $[x_n, y_n]$ and $[y_n, z_n]$, and let x be its midpoint. (If, as frequently happens, the two intervals are of the same length, take x to be the midpoint of $[x_n, y_n]$ if $f(x_n) < f(z_n)$, and the midpoint of $[y_n, z_n]$ if $f(z_n) < f(x_n)$; if you cannot distinguish $f(x_n)$ from $f(z_n)$, take either midpoint.) Now,

- if $x = \frac{1}{2}(x_n + y_n)$ then
  - if $f(x) < f(y_n)$ set $x_{n+1} = x_n$, $y_{n+1} = x$, $z_{n+1} = y_n$;
  - if $f(x) > f(y_n)$ set $x_{n+1} = x$, $y_{n+1} = y_n$, $z_{n+1} = z_n$;
- if $x = \frac{1}{2}(y_n + z_n)$ then
  - if $f(x) < f(y_n)$ set $x_{n+1} = y_n$, $y_{n+1} = x$, $z_{n+1} = z_n$;
  - if $f(x) > f(y_n)$ set $x_{n+1} = x_n$, $y_{n+1} = y_n$, $z_{n+1} = x$.

What if $f(x) = f(y_n)$? There is a theoretical possibility that we may come to points x, y_n such that $f(x)$ and $f(y_n)$ are either equal, or cannot be distinguished from each other by the method we have for calculating f. In this case, it may be worth looking at the three points x, $\frac{1}{2}(x + y_n)$ and y_n; but if you cannot quickly find a convincing triplet in which the middle term gives the least value of f, the algorithm becomes unsafe.

When should you stop? Normally the stopping criterion will be when you have a triplet (x_n, y_n, z_n) with $z_n - x_n \leq \delta$ for some pre-determined $\delta > 0$.

When does the method work? Provided f is continuous and defined on the whole initial interval $[a, c]$, the algorithm will halt in finitely many steps, and there will then be a local minimum in the interval $[x_n, z_n]$.

(b) *Advantages of the method* (i) It works for all continuous functions f, provided we can find initial values a, b and c for which $f(b) < \min(f(a), f(c))$ and f is defined on the whole interval $[a, c]$.

(ii) The values $f(x)$ need be calculated only to sufficient accuracy to determine which of $f(x)$, $f(y_n)$ is least. If we set our f-calculating routine to produce only approximate values, then we may need to re-calculate some of the $f(y_n)$ to greater accuracy when we find that the new $f(x)$ is at first sight indistinguishable.

(iii) When the algorithm halts (other than in the exceptional case in which we have found an x for which we cannot distinguish $f(x)$ from $f(y_n)$), we shall have an explicit interval-of-uncertainty $[x_n, z_n]$ with a guarantee that there is at least one solution in that interval.

(iv) If our stopping criterion is that the interval-of-uncertainty $[x_n, z_n]$ should be of length at most δ, then we can estimate at the start how many times we shall have to calculate f, since every step except perhaps one must reduce the length of the interval of uncertainty by at least 25%.

Disadvantages of the method (i) The geometric reduction in the interval-of-uncertainty is slow.

(ii) Unless we have further information about our function f (e.g., that its derivative is bounded, or that it is convex) we can never be sure that our 'best' value $f(y_n)$ is really close to the minimum value of f.

3F Quadratic fit

To find a local minimum of f, starting from distinct points a, b and c.

(a) *The algorithm*: Set $x_0 = a$, $x_1 = b$ and $x_2 = c$. Given that x_n, x_{n+1} and x_{n+2} are distinct, set

$$x_{n+3} = \frac{(x_{n+1}^2 - x_{n+2}^2)f(x_n) + (x_{n+2}^2 - x_n^2)f(x_{n+1}) + (x_n^2 - x_{n+1}^2)f(x_{n+2})}{2((x_{n+1} - x_{n+2})f(x_n) + (x_{n+2} - x_n)f(x_{n+1}) + (x_n - x_{n+1})f(x_{n+2}))};$$

continue.

The picture: imagine a parabola drawn through the points $(x_n, f(x_n))$, $(x_{n+1}, f(x_{n+1}))$ and $(x_{n+2}, f(x_{n+2}))$ on the graph of f; this will be horizontal at x_{n+3}. (Its equation is

$$g(x) = \frac{(x-x_{n+1})(x-x_{n+2})}{(x_n-x_{n+1})(x_n-x_{n+2})}f(x_n) + \frac{(x-x_n)(x-x_{n+2})}{(x_{n+1}-x_n)(x_{n+1}-x_{n+2})}f(x_{n+1}) + \frac{(x-x_{n+1})(x-x_{n+2})}{(x_{n+2}-x_n)(x_{n+2}-x_{n+1})}f(x_{n+2}).)$$

What if $(x_{n+1} - x_{n+2})f(x_n) + (x_{n+2} - x_n)f(x_{n+1}) + (x_n - x_{n+1})f(x_{n+2}) = 0$? There is a theoretical possibility that we may come to points x_n, x_{n+1} and x_{n+2} such that the three points $(x_n, f(x_n))$, $(x_{n+1}, f(x_{n+1}))$ and $(x_{n+2}, f(x_{n+2}))$ are either not all distinct or are collinear, so that we cannot draw a unique parabola through them. Equally catastrophic, but not so obvious and not requiring such a coincidence, is the possibility that the parabola is the wrong way up, so that we are looking at a *maximum* of the approximating function g rather than a minimum. Any computer program for this algorithm must therefore check that the coefficient of x^2 in g is positive, that is, that

$$\frac{f(x_n)}{(x_n-x_{n+1})(x_n-x_{n+2})} + \frac{f(x_{n+1})}{(x_{n+1}-x_n)(x_{n+1}-x_{n+2})} + \frac{f(x_{n+2})}{(x_{n+2}-x_{n+1})(x_{n+2}-x_n)} > 0.$$

If this is not the case the algorithm must be abandoned at once. It would also be sensible to abort the process if you ever find that $f(x_{n+3})$ is not less than $f(x_{n+2})$.

When should you stop? There is no built-in criterion for stopping. If, however, you are interested in finding an interval of length at most δ with a local minimum in it, you can continue the method until $|x_{n+3} - x_n| \leq \frac{1}{2}\delta$, and then look at the values $f(x_{n+3})$, $f(x_n)$ and $f(z)$ where $z = x_{n+3} \pm \frac{\delta}{2}$ is on the other side of x_{n+3} from x_n; if $f(x_{n+3})$ is the least of these, then you will know (provided, as usual, that f is continuous and defined on the whole interval between x_n and z) that you have a local minimum in this interval.

(b) *Advantages of the method* (i) When f is twice differentiable, and there is a solution x with $f'(x) = 0 \neq f''(x)$, we can hope that the x_n will converge rapidly to a stationary point.

(ii) Although the method cannot be expected to work for non-differentiable functions f, it does not rely on our being able to calculate the derivative of f.

Disadvantages of the method (i) It is rarely effective unless there is a solution x for which f is twice differentiable at x.

(ii) Even if f is very well behaved, if we start from the 'wrong' points x_0, x_1 and x_2 the method can fail.

3G Remarks Observe here that 'binary search' and the 'three-point method' go together as slow-but-sure, while 'false position', 'Newton's method' and 'quadratic fit' are fast-but-unsafe. If, therefore, you are interested in writing programs which will be effective with totally unknown functions, you should be willing to alternate between methods; e.g., using a 'slow' method until you seem to be within sight of an answer, then a 'fast' method until it shows signs of breaking down, and back to the 'slow' method for a couple of steps if necessary, or to confirm that the 'fast' method has indeed located a solution, as suggested in 3F.

3H Many dimensions When we come to optimization in more than one dimension, we have difficulties of new kinds, and must expect to move very much more slowly. I describe two methods, neither of them either fast or safe.

Line search: steepest descent

> To minimize a differentiable real function f of many real variables, starting from an initial point $\mathbf{a}$.

The algorithm: Set $\mathbf{x}^{(0)} = \mathbf{a}$. Given $\mathbf{x}^{(n)}$, set $\mathbf{u}^{(n)} = \nabla f(\mathbf{x}^{(n)}$, $g_n(t) = f(\mathbf{x}^{(n)} - t\mathbf{u}^{(n)})$ for $t \in \mathbb{R}$; seek a minimum t_n for the real function g_n by any of the one-dimensional methods 3D-3F, or otherwise; set $\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - t_n\mathbf{u}^{(n)}$.

The idea: At least for $\mathbf{x} \simeq \mathbf{0}$, we expect $f(\mathbf{x}^{(n)} + \mathbf{x})$ to be approximately $f(\mathbf{x}^{(n)}) + \mathbf{u}^{(n)T}\mathbf{x}$. So we can reduce f fastest by moving in the direction $-\mathbf{u}^{(n)}$. Since one-dimensional searching is enormously faster than anything we can do in more than one dimension, it is worth investing a bit of time in estimating how far to go. But note that it is not worth while to locate t_n particularly accurately, at least to begin with, since we expect to have to repeat the process.

When should you stop? The natural criterion to impose is that you terminate when $\|\nabla f(\mathbf{x}^{(n)})\|$ seems to be small enough. But unless you know a lot about the function f there is no way to be sure that you are close even to a local minimum.

Remarks (i) Note that since we can calculate ∇f, we shall also be able to calculate

$$g_n'(t) = -\nabla f(\mathbf{x}^{(n)} - t\mathbf{u}^{(n)})^T\mathbf{u}^{(n)}$$

for any t, and when minimising g_n we have the option of seeking a solution of $g_n' = 0$, as in 3D.

(ii) It is helpful also to note that as $g_n'(0) = -\|\mathbf{u}^{(n)}\|^2 < 0$ (except in the case $\mathbf{u}^{(n)} = \mathbf{0}$, in which case we expect to stop), we know that we are looking for a minimum $t_n > 0$; so if we want to use binary search, we will try to start with a pair $(0, \hat{t})$ where $\hat{t} > 0$ and $g_n'(\hat{t}) > 0$. But in many cases finding such a $\hat{t}$ is already a non-trivial problem.

3I Newton's method If we can calculate the second derivative $\mathbf{H}f$, we have a further method available.

> To minimize a twice differentiable real function f of many real variables, starting from an initial point $\mathbf{a}$.

The algorithm: Set $\mathbf{x}^{(0)} = \mathbf{a}$. Given $\mathbf{x}^{(n)}$, set

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \mathbf{H}f(\mathbf{x}^{(n)})^{-1}\nabla f(\mathbf{x}^{(n)});$$

continue.

The idea: At least for $\mathbf{x} \simeq \mathbf{x}^{(n)}$, we expect $\nabla f(\mathbf{x})$ to be approximately $\nabla f(\mathbf{x}^{(n)}) + \mathbf{H}f(\mathbf{x}^{(n)})(\mathbf{x} - \mathbf{x}^{(n)})$. So we can look for a solution of $\nabla f(\mathbf{x}) = \mathbf{0}$ by solving $\nabla f(\mathbf{x}^{(n)}) + \mathbf{H}f(\mathbf{x}^{(n)})(\mathbf{x} - \mathbf{x}^{(n)}) = \mathbf{0}$, that is, $\mathbf{x} = \mathbf{x}^{(n)} - \mathbf{H}f(\mathbf{x}^{(n)})^{-1}\nabla f(\mathbf{x}^{(n)})$.

When should you stop? As before, the natural criterion to impose is that you terminate when $\|\nabla f(\mathbf{x}^{(n)})\|$ seems to be small enough. Here we are supposing that we can calculate $\mathbf{H}f$, so it would be sensible to check that $\mathbf{H}f(\mathbf{x}^{(n)})$ is positive definite, giving us a fair chance that we are close to a minimum.

Remark As usual, Newton's method is unsafe in that it can easily go astray if we start in the wrong place; steepest descent will at least give $f(\mathbf{x}^{(n+1)}) \le f(\mathbf{x}^{(n)})$. If our function is indeed twice differentiable, and the second derivative $\mathbf{H}f$ is continuous, and there is a solution $\mathbf{x}^*$ at which $\nabla f(\mathbf{x}^*) = 0$ and $\mathbf{H}(\mathbf{x}^*)$ is strictly positive, and if we start close to $\mathbf{x}^*$, then Newton's method can be expected to converge rapidly to $\mathbf{x}^*$.

4. Convex Programs

4A Definitions (a) A **convex program** is a problem of the form
(P) minimise $f(\mathbf{x})$
subject to $\mathbf{x} \in C$
and $g_1(\mathbf{x}), \dots, g_k(\mathbf{x}) \leq 0$
where $C \subseteq \mathbb{R}^n$ is a non-empty convex set and f, $g_1, \dots, g_k$ are continuous convex functions defined (at least) on C.

(b) The **dual** of the convex program (P) is now
(D) maximise $h(\boldsymbol{\lambda}) = \inf_{\mathbf{x}\in C} L(\mathbf{x}, \boldsymbol{\lambda})$
subject to $\lambda_1, \dots, \lambda_k \leq 0$,
where

$$L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \lambda_1 g_1(\mathbf{x}) - \dots - \lambda_k g_k(\mathbf{x}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \boldsymbol{g}(\mathbf{x})$$

for $\mathbf{x} \in C$, $\lambda \in \mathbb{R}^k$. (Here $\mathbf{g}(\mathbf{x})$ is the vector $\begin{pmatrix} g_1(\mathbf{x}) \\ \dots \\ g_k(\mathbf{x}) \end{pmatrix}$.) In this expression, I allow myself to write $h(\boldsymbol{\lambda}) = -\infty$ if $\{L(\mathbf{x}, \boldsymbol{\lambda}) : \boldsymbol{x} \in C\}$ is unbounded below.

(c) Suppose that a convex program (P) is set out as in (a) above.

(i) The **feasible set** of (P) is $F = \{\mathbf{x} : \mathbf{x} \in C, \ \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\}$

(ii) (P) is **consistent** if F is not empty, that is, there is an $\mathbf{x} \in C$ such that $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$.

(iii) (P) is **superconsistent** if there is an $\mathbf{x} \in C$ such that $g_j(\mathbf{x}) < 0$ for every $j \leq k$.

4B Dual optimal pairs (a) Suppose that a convex program (P) and its dual (D) is set out as in 4A. If $\mathbf{x} \in F$ and $\boldsymbol{\lambda} \leq \mathbf{0}$, then $\lambda_j \leq 0$ and $g_j(\mathbf{x}) \leq 0$ for every j, so $\lambda_j g_j(\mathbf{x}) \geq 0$ for every j and $\boldsymbol{\lambda}^T \boldsymbol{g}(\mathbf{x}) \geq 0$. Consequently

$$h(\boldsymbol{\lambda}) \leq L(\mathbf{x}, \boldsymbol{\lambda})$$

(because $\mathbf{x} \in C$)

$$= f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) \leq f(\mathbf{x}).$$

It follows that

$$\sup_{\boldsymbol{\lambda}\leq\mathbf{0}} h(\lambda) \leq \inf_{\mathbf{x}\in F} f(\mathbf{x}),$$

where I allow myself to write $\inf_{\mathbf{x}\in F} f(\mathbf{x}) = \infty$ if F is empty, and $\inf_{\mathbf{x}\in F} f(\mathbf{x}) = -\infty$ if f is not bounded below on F.

(b) The **duality gap** of the pair (P), (D) is the interval

$$\left]\sup_{\boldsymbol{\lambda}\leq\mathbf{0}} h(\boldsymbol{\lambda}), \inf_{\mathbf{x}\in F} f(\mathbf{x})\right[.$$

(c) A **dual optimal pair** for (P) and (D) is a pair $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ such that $\mathbf{x}^* \in F$, $\boldsymbol{\lambda}^* \leq \mathbf{0}$ and $h(\boldsymbol{\lambda}^* = f(\mathbf{x}^*)$. Note that in this case we must have $(\boldsymbol{\lambda}^*)^T \mathbf{g}(\mathbf{x}^*) = 0$, so that $\lambda_j^* g_j(\mathbf{x}^*) = 0$ for every j. At the same time,

$$h(\boldsymbol{\lambda}) \leq f(\mathbf{x}^*) = h(\boldsymbol{\lambda}^*) \leq f(\mathbf{x})$$

whenever $\mathbf{x} \in F$ and $\boldsymbol{\lambda} \leq \mathbf{0}$, so that $\mathbf{x}^*$ is an optimum for (P) and $\boldsymbol{\lambda}^*$ is an optimum for (D). Observe that if there is a dual optimal pair then the optimal values of (D) and (P) are the same, so that the duality gap is empty.

4C Why we take the trouble In the applications of this theory, we expect to understand the convex set C; for instance, it might be $\{\mathbf{x} : \mathbf{x} \geq \mathbf{0}\}$, or $\{\mathbf{x} : \|\mathbf{x}\| < 1\}$. We are often very unclear about the set F; we may not even know whether it is empty or not. The point of the dual problem (D) is that it replaces the single optimisation in (P) by two optimisations: find $h(\boldsymbol{\lambda})$, for enough $\boldsymbol{\lambda}$, and then optimise. In each of these, although the *function* $L(\mathbf{x},\boldsymbol{\lambda})$ or $h(\boldsymbol{\lambda})$ which we are minimising or maximising is likely to be more complicated than the function f involved in (P), the *sets* over which we are optimising, C or $\{\boldsymbol{\lambda} : \boldsymbol{\lambda} \le \mathbf{0}\}$ are likely to be simpler than the set F. For this reason, (D) can give us two problems each a little easier than the original problem (P). Almost the most important idea in the whole of mathematics is that it is sometimes worth going the long way round.

Of course we don't like to set out on the longer path without some assurance that the answer is to be found at the end of it. For this reason we look for general theorems to assure us that the duality gap is empty in a particular case; if possible, to guarantee the existence of a dual optimal pair.

4D Lemma Let $C \subseteq \mathbb{R}^n$ be a convex set, and $f,\, g : C \to \mathbb{R}$ continuous convex functions. Suppose that $V_f = \{x : x \in C,\ f(x) < 0\}$ and $V_g = \{x : x \in C,\ g(x) < 0\}$ are disjoint. Then there is a $t \in [0,1]$ such that $tf + (1-t)g \ge 0$ everywhere on C.

proof For $t \in [0,1]$, set $W_t = \{x : tf(x) + (1-t)g(x) < 0\}$. Then W_t is convex and $W_t \subseteq V_f \cup V_g$; we seek some t such that $W_t = \emptyset$. Now $G = \{t : W_t \cap V_f \ne \emptyset\}$ and $H = \{t : W_t \cap V_g \ne \emptyset\}$ are relatively open sets in $[0,1]$ (because f and g are continuous) and are disjoint. **P?** If $x \in W_t \cap V_f$ and $y \in W_t \cap V_g$, then every point on the line segment $[x,y]$ belongs to $V_f \cup V_g$; but these meet $[x,y]$ in disjoint non-empty relatively open sets. **XQ** So $G \cup H \ne [0,1]$, which is what we need to know.

4E Lemma Let $C \subseteq \mathbb{R}^n$ be a convex set, and $f_0, \dots, f_k : C \to \mathbb{R}$ continuous convex functions. Suppose that $\max_{i \le k} f_i(x) \ge 0$ for every $x \in C$. Then there is a convex combination g of $f_0, \dots, f_k$ such that $g(x) \ge 0$ for every $x \in C$.

proof Induce on k. For $k = 0$ the result is trivial and for $k = 1$ it is the last lemma. For the inductive step to $k+1 \ge 2$, set $C' = \{x : x \in C,\ f_i(x) < 0 \text{ for every } i < k\}$, so that C' is a convex set. By the last lemma, there is a convex combination g_0 of f_k and f_{k+1} such that $g_0(x) \ge 0$ for every $x \in C'$. But this means that $\max(\{g_0(x)\} \cup \{f_i(x) : i < k\}) \ge 0$ for every $x \in C$. By the inductive hypothesis, there is a convex combination g of $f_0, \dots, f_{k-1}, g_0$ which is non-negative everywhere on X, and g is a convex combination of $f_0, \dots, f_{k+1}$, so the induction proceeds.

4F Theorem Suppose we are given a superconsistent convex program (P) in the form set out in 4A. If $\mathbf{x}^*$ is a solution to (P), then there is a $\boldsymbol{\lambda}^* \le \mathbf{0}$ such that $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ is a dual optimal pair.

proof Set $\alpha = f(\mathbf{x}^*)$. Then there can be no $\mathbf{x} \in C$ such that $\max(f(\mathbf{x}) - \alpha, g_1(\mathbf{x}), \dots, g_k(\mathbf{x})) < 0$. So by 4E there is a convex combination g of $f - \alpha,\ g_1, \dots, g_k$ such that $g(\mathbf{x}) \ge 0$ for every $\mathbf{x} \in C$.

Because (P) is superconsistent, there is a $\mathbf{z} \in C$ such that $g_j(\mathbf{z}) < 0$ for every j. Then $g(\mathbf{z}) \ge 0$, so g is not a convex combination of $g_1, \dots, g_k$ and is expressible as $\beta(f - \alpha) + \sum_{i=1}^k \beta_i g_i$ where $\beta > 0$ and $\beta_i \ge 0$ for every i. Set $\lambda_i^* = -\beta_i/\beta$ for each i, so that $L(\mathbf{x}, \boldsymbol{\lambda}^*) = \alpha + \beta^{-1} g(\mathbf{x}) \ge \alpha$ for every $\mathbf{x} \in C$ and $h(\boldsymbol{\lambda}^*) \ge \alpha$. But also $h(\boldsymbol{\lambda}^*) \le f(\mathbf{x}^*) = \alpha$ so $h(\boldsymbol{\lambda}^*) = \alpha = f(\mathbf{x}^*)$ and we have a dual optimal pair.

4G Theorem Suppose we are given a consistent convex program (P) in the form set out in 4A in which C is closed and bounded. Then the duality gap is empty.

proof Because $F = \{\mathbf{x} : \mathbf{x} \in C,\ \mathbf{g}(\mathbf{x}) \le 0\}$, $\mathbf{g}$ is continuous and C is closed, F is closed. Because C is bounded, F is bounded; because (P) is consistent, F is not empty; because f is continuous, it attains its minimum on F at $\mathbf{x}^*$ say, which is now a solution of (P).

Set $\alpha = f(\mathbf{x}^*)$. Let $\epsilon > 0$. Then $C' = \{\mathbf{x} : \mathbf{x} \in C,\ f(\mathbf{x}) \le \alpha - \epsilon\}$ is a compact set not meeting the compact set $\{\mathbf{x} : \mathbf{x} \in C,\ \max_{j \le k} g_j(\mathbf{x}) \le 0\}$. So there is an $\eta > 0$ such that C' does not meet $\{\mathbf{x} : \mathbf{x} \in C,\ \max_{j \le k} g_j(\mathbf{x}) \le \eta\}$. Applying Lemma 4E to $f - \alpha + \epsilon,\ g_j - \eta$ we have $\beta,\ \beta_j \ge 0$, not all 0, such that $\beta(f(\mathbf{x}) - \alpha + \epsilon) + \sum_{j=1}^k \beta_j (g_j(\mathbf{x}) - \eta) \ge 0$ for every $\mathbf{x} \in C$. Since $g_j(\mathbf{x}^*) \le 0$ for every j, $\beta > 0$ and $f(\mathbf{x}) - \alpha + \epsilon + \sum_{j=1}^k \frac{\beta_j}{\beta}(g_j(\mathbf{x}) - \eta) \ge 0$ for every $\mathbf{x} \in C$.

Set $\lambda_j = -\frac{\beta_j}{\beta}$ for $j \le k$. Then

$$L(\mathbf{x},\boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{j=1}^{k} \frac{\beta_j}{\beta} g_j(\mathbf{x})$$
$$\geq f(\mathbf{x}) + \sum_{j=1}^{k} \frac{\beta_j}{\beta}(g_j(\mathbf{x}) - \eta) \geq \alpha - \epsilon$$

for every $\mathbf{x} \in C$. Taking the minimum over $\mathbf{x}$, $h(\boldsymbol{\lambda}) \geq \alpha - \epsilon$. As ϵ is arbitrary, $\sup_{\boldsymbol{\lambda}\leq\mathbf{0}} h(\boldsymbol{\lambda}) \geq \alpha$ and the duality gap is $]\alpha,\alpha[= \emptyset$.

4H Theorem Suppose we are given a convex program (P) and its dual (D) set out in the form in 4A. Suppose that moreover

f and every g_j is differentiable;

we have a differentiable function $\mathbf{x}^*(\boldsymbol{\lambda})$, defined at least for every $\boldsymbol{\lambda} \leq \mathbf{0}$, such that $\mathbf{x}^*(\boldsymbol{\lambda}) \in C$ and $h(\boldsymbol{\lambda}) = L(\mathbf{x}^*(\boldsymbol{\lambda}),\boldsymbol{\lambda})$ for each $\boldsymbol{\lambda} \leq \mathbf{0}$;

we have a $\boldsymbol{\lambda}^* \leq \mathbf{0}$ such that $h(\boldsymbol{\lambda}) \leq h(\boldsymbol{\lambda}^*)$ for every $\boldsymbol{\lambda} \leq \mathbf{0}$;

$x^*(\boldsymbol{\lambda}^*) \in \operatorname{int} C$.

Then $(x^*(\boldsymbol{\lambda}^*),\boldsymbol{\lambda}^*)$ is a dual optimal pair for (P) and (D).

Remark Note that in this theorem we do not have to examine the set F at any point, even to confirm that it is not empty. We need a comparatively strong hypothesis on the functions f and g_j (in the other theorems it is enough if they are continuous), and we need to be able to solve (D) in a very strong sense, because we have to solve the problems

$$\text{minimise } L(\mathbf{x},\boldsymbol{\lambda}) \text{ for } \mathbf{x} \in C$$

in a 'uniform' way, getting the solution as a (differentiable) function $\mathbf{x}^*(\boldsymbol{\lambda})$, and then we need to compute the optimum for h. But *if* we can do all this, and C is open, or for some other reason $\mathbf{x}^*(\boldsymbol{\lambda}^*)$ belongs to its interior, we shall have a solution for (P).

proof (a) The function $\mathbf{x}^*$ is continuous, so

$$G = \{\boldsymbol{\lambda} : x^*(\boldsymbol{\lambda}*) \in \operatorname{int} C\}$$

is a neighbourhood of $\boldsymbol{\lambda}^*$. Now for any $\boldsymbol{\lambda} \in G$, $i \leq n$ we have

$$\tfrac{\partial f}{\partial x_i}(\mathbf{x}^*(\boldsymbol{\lambda}) = \textstyle\sum_{l=1}^{k} \lambda_l \tfrac{\partial g_l}{\partial x_i}(\mathbf{x}^*(\boldsymbol{\lambda}).$$

P Because all our functions are differentiable, so is $\mathbf{x} \mapsto L(\mathbf{x},\boldsymbol{\lambda})$. This function takes an optimal value at $\mathbf{x}^*(\boldsymbol{\lambda})$; but as this belongs to the interior of C, this must be a stationary point, that is, $\nabla L(\mathbf{x},\boldsymbol{\lambda})$ must be zero at $\mathbf{x}^*(\boldsymbol{\lambda})$. But this means that

$$0 = \frac{\partial}{\partial x_i} L(\mathbf{x},\boldsymbol{\lambda})$$
$$= \frac{\partial}{\partial x_i}(f(\mathbf{x}) - \sum_{l=1}^{k} \lambda_l g_l(\mathbf{x})) = \frac{\partial f}{\partial x_i} - \sum_{l=1}^{k} \lambda_l \frac{\partial g_l}{\partial x_i}$$

at $\mathbf{x}^*(\boldsymbol{\lambda})$; rearranging, we have the given formula. **Q**

(b) Next, we have $h(\boldsymbol{\lambda}) = L(\mathbf{x}^*(\boldsymbol{\lambda}),\boldsymbol{\lambda})$ for $\boldsymbol{\lambda} \leq \mathbf{0}$, so this also is differentiable. It does not follow that ∇h is zero at $\boldsymbol{\lambda}^*$, because it may be that $\lambda_j^* = 0$ for one or more j. But since we are always allowed to shift λ_j^* *downwards* without leaving the feasible set $\{\boldsymbol{\lambda} : \boldsymbol{\lambda} \leq \mathbf{0}\}$, we must have $\frac{\partial h}{\partial \lambda_j} \geq 0$ at $\boldsymbol{\lambda}^*$, with equality if $\lambda_j^* < 0$. Now this means that, at $\boldsymbol{\lambda}^*$,

$$
\begin{aligned}
0 \leq \frac{\partial h}{\partial \lambda_j} &= \frac{\partial}{\partial \lambda_j} L(\mathbf{x}^*, \boldsymbol{\lambda}) \\
&= \frac{\partial}{\partial \lambda_j}(f(\mathbf{x}^*) - \sum_{l=1}^{k} \lambda_l g_l(\mathbf{x}^*)) \\
&= \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{\partial x_i^*}{\partial \lambda_j} - g_j(\mathbf{x}^*) - \sum_{l=1}^{k} \lambda_l \frac{\partial}{\partial \lambda_j} g_l(\mathbf{x}^*)) \\
&= \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{\partial x_i^*}{\partial \lambda_j} - g_j(\mathbf{x}^*) - \sum_{l=1}^{k} \sum_{i=1}^{n} \lambda_l \frac{\partial g_l}{\partial x_i} \frac{\partial x_i^*}{\partial \lambda_j} \\
&= \sum_{i=1}^{n} \frac{\partial x_i^*}{\partial \lambda_j} (\frac{\partial f}{\partial x_i} - \sum_{l=1}^{k} \lambda_l \frac{\partial g_l}{\partial x_i}) - g_j(\mathbf{x}^*) = -g_j(\mathbf{x}^*)
\end{aligned}
$$

with equality when $\lambda_j^* \neq 0$. Accordingly $g_j(\mathbf{x}^*) \leq 0$ for every j, and $\mathbf{x}^* \in F$; moreover, $\lambda_j^* g_j(\mathbf{x}^*) = 0$ for every j so

$$h(\boldsymbol{\lambda}^*) = L(\mathbf{x}^*, \boldsymbol{\lambda}^*) = f(\mathbf{x}^*),$$

and $\mathbf{x}^* = \mathbf{x}^*(\boldsymbol{\lambda}^*)$, $\boldsymbol{\lambda}^*$ are a dual optimal pair.