Optimization

Dr Brad Baxter
Department of Mathematics
Imperial College, London SW7 2BZ
b.baxter@ic.ac.uk
www.ma.ic.ac.uk/∼baxter

1. Introduction

Here are some examples of optimization problems.

(i) Changing the orbit of a spacecraft using as little fuel as possible.
(ii) Minimizing the energy wasted in electrical power distribution each day in Britain.
(iii) Maximizing the profit of a portfolio of stocks and bonds subject to the estimated risk of the portfolio not exceeding some chosen number.
(iv) Minimizing the estimated risk of a portfolio of stocks and bonds subject to the profit exceeding some chosen number.
(v) Minimizing the cost of a complex chemical reaction in plastics manufacture.
(vi) Calculating the most economical pipe diameter for a system based on flowrate, fluid density and fluid viscosity.

In all of these problems, we want to minimize, or maximize, a function $f(\mathbf{x})$. In fact, because maximizing a function $f(\mathbf{x})$ is equivalent to minimizing $-f(\mathbf{x})$, we shall only consider minimization. In many applications, we have constraints on the minimization, as in the examples above. This is called *constrained optimization.* If there are no constraints, then we have an *unconstrained optimization problem.* We shall mainly study the latter problem in this short introductory course. Some jargon: the function being optimized is often called the *objective function.*

Example 1.1 (i) An unconstrained optimization problem: Minimize $f(x, y) = x^2 + y^2$.
(ii) A constrained optimization problem: Minimize $f(x, y) = x^2 + y^2$ subject to $x + 3y = 1$ and $x, y \geq 0$. Here we have linear equality and inequality constraints.
(iii) Another constrained optimization problem: Minimize $f(x, y) = x^2 + y^2$ subject to $g(x, y) = (x - 2)^2 + 3(y - 4.2)^2 \leq 0.4$. Here the constraint is a nonlinear inequality constraint.

Excellent optimization software is available these days, and you will use some in your other courses. You might therefore wonder why we're teaching you the fundamental theory of optimization — after all, you only want to use the programs, not write your own. Here are some reasons.

(i) Even good software is far from perfect, and some understanding of the fundamental mathematics is necessary for those many occasions when programs do not behave as advertised.
(ii) Many optimization problems need some modification before they're suitable for standard software.
(iii) You may need to change optimization software before it's suitable for your application.

(iv) It's very easy to re-invent poor methods, such as the steepest descent method studied later.

Section 2 contains most of the course material, whilst Section 3 collects together some mathematical background material.

1.1. References

There is no ideal textbook for this course, although Chapter 10 of *Numerical Recipes*, by W. Press, B. Flannery, S. Teukolsky and W. Vetterling, is useful. Another good book is *Practical Optimization*, by S. Gill, W. Murray and M. Wright, although this is a little too advanced. Three good websites are `www-fp.mcs.anl.gov/opt`, `www.mathprog.org` and `www.netlib.org`.

2. Methods for Unconstrained Optimization

2.1. Taylor's theorem in one dimension

In one dimension, you already know that a function $f(x)$ can be expanded in a Taylor series

$$f(a+h) = f(a) + hf'(a) + \frac{1}{2}h^2 f''(a) + \cdots. \tag{2.1}$$

The omitted terms represented by "$\cdots$" contain h^3 and higher powers of h. Then the *quadratic approximation*

$$q(h) = f(a) + hf'(a) + \frac{1}{2}h^2 f''(a) \tag{2.2}$$

satisfies

$$f(a+h) = q(h) + E(h), \tag{2.3}$$

where the error term $E(h)$ is bounded by

$$|E(h)| \le Ch^3 \tag{2.4}$$

when $|h|$ is sufficiently small. We often summarize this by writing

$$f(a+h) = q(h) + \mathcal{O}(h^3), \tag{2.5}$$

and "$\mathcal{O}(h^3)$" is read as "big-oh of h^3". This is an extremely useful notation when discussing errors in approximation. As a further example, the *linear approximation*

$$\ell(h) = f(a) + hf'(a) \tag{2.6}$$

satisfies

$$f(a+h) = \ell(h) + \mathcal{O}(h^2). \tag{2.7}$$

Example 2.1 The function $\cos x = 1 + \mathcal{O}(x^2)$ for small $|x|$; the function $\sin x = x - x^3/6 + \mathcal{O}(x^5)$ for small $|x|$.

Definition 2.1 The function $f(x)$ has a *local minimum* at $x = a$ if

$$f(a+h) \ge f(a) \tag{2.8}$$

when $|h|$ is sufficiently small. It has a *local maximum* at $x = a$ if

$$f(a+h) \le f(a) \tag{2.9}$$

when $|h|$ is sufficiently small.

Example 2.2 Find the local maxima and local minima of $f(x) = \sin x$.

Theorem 2.1 If $f'(a) = 0$ and $f''(a) > 0$, then $f(x)$ has a local minimum at $x = a$.

Proof. We use the quadratic approximation (2.2). In this case, we have

$$q(h) = f(a) + \frac{1}{2}h^2 f''(a) > f(a),$$

for all $h \neq 0$. Since

$$f(a+h) = q(h) + \mathcal{O}(h^3)$$

for small $|h|$, we deduce that $f(a+h) \geq f(a)$ when $|h|$ is sufficiently small. □

Exercise 2.1 What's the corresponding theorem for local maxima?

Proposition 2.2 If $f'(a) \neq 0$, then f possesses neither a local maximum nor a local minimum at $x = a$.

Proof. In this case we use the linear approximation (2.6). The key observations are

$$\ell(h) > f(a)$$

when $hf'(a) > 0$, and

$$\ell(h) < f(a)$$

when $hf'(a) < 0$. Thus, because $f(a+h) = \ell(h) + \mathcal{O}(h^2)$ when $|h|$ is small, we cannot have a local maximum or local minimum of $f(x)$ at $x = a$. □

It's important to grasp that a local maximum is *not* a global maximum. For example, the function $f(x) = -(x^2 - 1)^2$ has a local minimum at $x = 0$, but this is not a global minimum. Further, the converse of Theorem 2.1 is *not* true.

Exercise 2.2 Find a function $f(x)$ with a global minimum at $x = 0$ for which $f''(a) = 0$.

2.2. Taylor's theorem in n dimensions

The simple results of the last section all have analogues when $f(\mathbf{x})$ is a function of n variables, but life is no longer quite so easy ...

Definition 2.2 Let $f(\mathbf{x})$ be a function of n variables. The *gradient* $\nabla f(\mathbf{x})$ of $f(\mathbf{x})$ is the vector function

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \tag{2.10}$$

of first partial derivatives of $f(\mathbf{x})$.

IMPORTANT NOTATION: Some students confuse the symbol ∂ for partial differentiation with the Greek letter δ. They are different! Confusing these symbols will be penalized in the coursework and at examination.

Example 2.3 Let $f(\mathbf{x}) = x_1^2 + x_2^2 + \cdots + x_n^2$. Then

$$\frac{\partial f}{\partial x_j} = 2x_j, \qquad \text{for } 1 \le j \le n,$$

or, more briefly,

$$\nabla f(\mathbf{x}) = 2\mathbf{x}. \tag{2.11}$$

Exercise 2.3 Find the gradient of $f(\mathbf{x}) = \sin x_1 \sin x_2 \cdots \sin x_n$.

The n-dimensional analogue of the linear approximation defined in (2.6) is the function

$$\ell(\mathbf{h}) = f(\mathbf{a}) + \sum_{k=1}^{n} h_j \frac{\partial f}{\partial x_j}(\mathbf{a}), \tag{2.12}$$

which we shall usually express by

$$\ell(\mathbf{h}) = f(\mathbf{a}) + \mathbf{h} \cdot \nabla f(\mathbf{a}). \tag{2.13}$$

Further, recalling that the *transpose* of the column vector

$$\mathbf{h} = \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{pmatrix}$$

is the row vector

$$\mathbf{h}^T = \begin{pmatrix} h_1 & h_2 & \cdots & h_n \end{pmatrix},$$

we write

$$\ell(\mathbf{h}) = f(\mathbf{a}) + \mathbf{h}^T \nabla f(\mathbf{a}), \tag{2.14}$$

which is the same as

$$\ell(\mathbf{h}) = f(\mathbf{a}) + \nabla f(\mathbf{a})^T \mathbf{h}. \tag{2.15}$$

The n-dimensional analogue of the quadratic approximation is more complex. You might expect this on realizing that $f(\mathbf{x})$ has n^2 second partial derivatives

$$\frac{\partial^2 f}{\partial x_j \partial x_k}, \qquad \text{for } j, k = 1, 2, \ldots, n.$$

Definition 2.3 The *second derivative matrix* $D^2 f(\mathbf{a})$, or *Hessian*, of the function $f(\mathbf{x})$ is defined by

$$\left(D^2 f(\mathbf{a})\right)_{jk} = \frac{\partial^2 f}{\partial x_j \partial x_k}, \qquad 1 \le j, k \le n. \tag{2.16}$$

We shall also assume that, for every function $f(\mathbf{x})$ studied in this course,

$$\frac{\partial^2 f}{\partial x_j \partial x_k} = \frac{\partial^2 f}{\partial x_k \partial x_j}. \tag{2.17}$$

Thus every second derivative matrix in this course will be a symmetric matrix.

Example 2.4 Let $f(\mathbf{x}) = x_1^2 x_2^2$, for $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2$. Then the second derivative matrix is

$$D^2 f(\mathbf{x}) = \begin{pmatrix} 2x_2^2 & 4x_1x_2 \\ 4x_1x_2 & 2x_1^2 \end{pmatrix}.$$

Exercise 2.4 Find the second derivative matrix when $f(\mathbf{x}) = x_1^2 + \cdots x_n^2$.

Definition 2.4 Let $f(\mathbf{x})$ be a function of n variables. The vector $\mathbf{d} \in \mathbb{R}^n$ is called a *descent direction* for $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{a}$ if

$$\mathbf{d}^T \nabla f(\mathbf{a}) < 0. \tag{2.18}$$

If $\mathbf{d}^T \nabla f(\mathbf{a}) > 0$, then we say that $\mathbf{d}$ is an *ascent direction*.

One way to understand descent directions is to let $z = f(\mathbf{x})$ be the height of a surface above sea level at the point $\mathbf{x} \in \mathbb{R}^2$. If you imagine walking in a particular direction $\mathbf{d}$ starting at the point $\mathbf{a}$, then $\mathbf{d}$ is a descent direction if and only if we're walking downhill. (In some books, descent directions are called downhill directions.)

Example 2.5 If $\nabla f(\mathbf{a}) \neq 0$, then $\mathbf{d} = -\nabla f(\mathbf{a})$ is a descent direction because

$$\mathbf{d}^T \nabla f(\mathbf{a}) = -\nabla f(\mathbf{a})^T \nabla f(\mathbf{a}) = \|\nabla f(\mathbf{a})\|^2.$$

Proposition 2.3 If $\nabla f(\mathbf{a}) \neq 0$, then $f(\mathbf{x})$ possesses neither a local minimum nor a local maximum at $\mathbf{x} = \mathbf{a}$.

Proof. We observe that $\mathbf{d} = -\nabla f(\mathbf{a})$ is a descent direction and $\mathbf{d} = +\nabla f(\mathbf{a})$ is an ascent direction. □

Exercise 2.5 Let $f(x, y) = x^2 + y^2$ and let $\mathbf{a} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$. Calculate $\nabla f(\mathbf{a})$ and sketch a diagram showing all descent directions and all ascent directions.

Descent directions lead to one of the most important ideas in optimization: *descent direction methods*. Here we attempt to find a local minimum of a function of n variables by starting at a point $\mathbf{x}_1 \in \mathbb{R}^n$ and choosing a descent direction $\mathbf{d}_1$. We move away from $\mathbf{x}_1$ along the descent direction $\mathbf{d}_1$ to find a new point $\mathbf{x}_2$ satisfying $f(\mathbf{x}_2) < f(\mathbf{x}_1)$ — this is called a *line search*. We then repeat the construction, choosing a new descent direction $\mathbf{d}_2$ at $\mathbf{x}_2$ and performing another line search to obtain a new point $\mathbf{x}_3$ satisfying $f(\mathbf{x}_3) < f(\mathbf{x}_2)$. Of course, we hope that the sequence of points $\mathbf{x}_1, \mathbf{x}_2, \ldots$ converges to a local minimum of $f(\mathbf{x})$, but any reduction can be valuable in applications. [For example, if the function being minimized is the electrical energy lost by heat dissipation in the National Grid, then even a small reduction saves vast amounts of money.] Most optimization methods apply this strategy, differing only in their choice of descent directions and the method chosen to perform the line search.

We need to study line searches in detail. To this end, define

$$\phi(\alpha) = f(\mathbf{a} + \alpha \mathbf{d}), \qquad \text{for } \alpha \geq 0. \tag{2.19}$$

Thus $\phi(\alpha)$ is simply the value of $f(\mathbf{x})$ when $\mathbf{x} = \mathbf{a} + \alpha \mathbf{d}$.

Proposition 2.4 The derivative of $\phi(\alpha)$ is given by

$$\phi'(\alpha) = \mathbf{d}^T \nabla f(\mathbf{a} + \alpha \mathbf{d}). \tag{2.20}$$

In particular, $\phi'(0) = \mathbf{d}^T \nabla f(\mathbf{a}) < 0$ when $\mathbf{d}$ is a descent direction for $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{a}$. Furthermore, the second derivative is

$$\phi''(\alpha) = \mathbf{d}^T D^2 f(\mathbf{a} + \alpha \mathbf{d}) \mathbf{d}. \tag{2.21}$$

Proof. Using a Taylor series for the univariate function $\phi(\alpha)$, we have

$$\phi(\alpha+h) = \phi(\alpha) + h\phi'(\alpha) + \frac{1}{2}h^2\phi''(\alpha) + O(h^3). \tag{2.22}$$

Using the Taylor series for the function $f(\mathbf{x})$ of n variables, we have

$$\begin{aligned}
\phi(\alpha+h) &= f(\mathbf{a} + (\alpha+h)\mathbf{d}) \\
&= f((\mathbf{a}+\alpha\mathbf{d}) + h\mathbf{d}) \\
&= f(\mathbf{a}+\alpha\mathbf{d}) + (h\mathbf{d})^T\nabla f(\mathbf{a}+\alpha\mathbf{d}) + \frac{1}{2}h^2\mathbf{d}^T D^2 f(\mathbf{a}+\alpha\mathbf{d})\mathbf{d} + \mathcal{O}(h^3) \\
&= \phi(\alpha) + h\mathbf{d}^T\nabla f(\mathbf{a}+\alpha\mathbf{d}) + \frac{1}{2}h^2\mathbf{d}^T D^2 f(\mathbf{a}+\alpha\mathbf{d})\mathbf{d} + \mathcal{O}(h^3). \qquad (2.23) \\
& \qquad (2.24)
\end{aligned}$$

We now equate the h and h^2 terms in (2.22) and (2.24). □

We shall now prove the sufficient condition for local minima of multivariate functions analogous to Theorem 2.1.

Theorem 2.5 Suppose $\nabla f(\mathbf{a}) = 0$ and

$$\mathbf{d}^T D^2 f(\mathbf{a})\mathbf{d} > 0 \tag{2.25}$$

for every nonzero vector $\mathbf{d} \in \mathbb{R}^n$. The $f(\mathbf{x})$ has a local minimum at $\mathbf{x} = \mathbf{a}$.

Proof. Using the previous proposition, we see that the line search function $\phi(\alpha)$ satisfies $\phi'(0) = 0$ and $\phi''(0) > 0$. Hence $\phi(\alpha)$ has a local minimum at $\alpha = 0$, by Theorem 2.1, for every nonzero vector $\mathbf{d} \in \mathbb{R}^n$. Hence $f(\mathbf{x})$ has a local minimum at $\mathbf{x} = \mathbf{a}$. □

The sufficient condition satisfied by the second derivative matrix in Theorem 2.5 is so important that it has its own name:

Definition 2.5 Let A be any matrix. We say that A is *positive definite* if

$$\mathbf{u}^T A\mathbf{u} > 0 \tag{2.26}$$

for every nonzero vector $\mathbf{u} \in \mathbb{R}^n$.

How do we know that a matrix is positive definite? If A is symmetric, then it's positive definite if and only if its eigenvalues are all positive numbers. A simpler condition for 2×2 matrices is given in the exercise sheets.

Exercise 2.6 (i) When is

$$\begin{pmatrix} P & 0 \\ 0 & Q \end{pmatrix}$$

positive definite?

(ii) Prove that

$$\begin{pmatrix} 4 & -1 \\ -1 & 4 \end{pmatrix}$$

is positive definite.

We now restate Theorem 2.5 in the new terminology.

Theorem 2.6 If $\nabla f(\mathbf{a}) = 0$ and the second derivative matrix $D^2 f(\mathbf{a})$ is positive definite, then $f(\mathbf{x})$ has a local minimum at $\mathbf{x} = \mathbf{a}$.

2.3. Exact line search

Given a point $\mathbf{a} \in \mathbb{R}^n$ and a descent direction $\mathbf{d}$, we know that we can reduce $f(\mathbf{x})$ by going along the half-line

$$L = \{\mathbf{a} + \alpha\mathbf{d} : \alpha \geq 0\}. \tag{2.27}$$

But how far do we go? An exact line search increases α until $\phi(\alpha)$ begins to increase again. In other words, we "walk" in the downhill direction $\mathbf{d}$ as far as possible, stopping when we begin to ascend. Informally, imagine a blind man walking over a hilly landscape. He walks downhill in a particular direction, stopping once the path begins to go uphill. He then chooses a new downhill direction and repeats the process. More formally, we apply the next definition.

Definition 2.6 If $\mathbf{d}$ is a descent direction for $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{a}$, then an *exact line search* chooses the smallest positive number α^* for which $\phi'(\alpha^*) = 0$.

Exercise 2.7 Prove that $\phi'(\alpha) < 0$ for $0 \leq \alpha < \alpha^*$ when $\phi'(0) < 0$ and α^* is as defined above.

Example 2.6 We can calculate exact line searches analytically for quadratic functions, as we shall now illustrate. Let $f(x, y) = x^2 + 3y^2$ and $\mathbf{a} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. We have

$$\nabla f(x, y) = \begin{pmatrix} 2x \\ 6y \end{pmatrix},$$

and the steepest descent direction is

$$\mathbf{d} = -\nabla f(\mathbf{a}) = \begin{pmatrix} -2 \\ -6 \end{pmatrix}.$$

Then

$$L = \{\mathbf{a} + \alpha\mathbf{d} : \alpha \geq 0\} = \{\begin{pmatrix} 1 \\ 1 \end{pmatrix} + \alpha \begin{pmatrix} -2 \\ -6 \end{pmatrix} = \begin{pmatrix} 1 - 2\alpha \\ 1 - 6\alpha \end{pmatrix} : \alpha \geq 0\}$$

and $\phi(\alpha) = f(\mathbf{a} + \alpha\mathbf{d}) = f(1 - 2\alpha, 1 - 6\alpha) = (1 - 2\alpha)^2 + 3(1 - 6\alpha)^2$. Thus $\phi'(\alpha^*) = 0$ if and only if $\alpha^* = 5/28$, which implies

$$\mathbf{b} = \begin{pmatrix} 9/14 \\ -1/14 \end{pmatrix}.$$

Example 2.7 Exact line searches can fail! For example, if $\phi(\alpha) = \exp(-\alpha)$, then ϕ does not attain its minimum value. Good software packages will terminate the line search, but poorer programs will continue until floating point overflow occurs! In most theoretical work we assume that this doesn't happen, as I shall do in these notes.

How do we perform an exact line search for a general function? Good algorithms are rather fiddly and would take too long to explain in this short course. Here is a very simple algorithm based on the bisection method for finding a root of a nonlinear equation.

Algorithm 2.1. (Bisection algorithm for exact line search.) Let $\phi'(0) < 0$ and find $A > 0$ such that $\phi'(A) > 0$. Let $\epsilon > 0$ be a chosen tolerance and set $L = 0$, $R = A$.

```
While $R - L > \epsilon$
```

If $\|\phi'(\frac{L+R}{2})\| < \epsilon$ then STOP.
If $\phi'(\frac{L+R}{2}) < 0$, replace L by $(L+R)/2$.
If $\phi'(\frac{L+R}{2}) > 0$, replace R by $(L+R)/2$.

Exercise 2.8 Apply Algorithm 2.1 when $\phi(\alpha) = -\sin\alpha$ and $A = \pi$.

Good software packages contain faster reliable codes for exact line searches. You can learn more in *Numerical Recipes* or at `www.netlib.org`.

Proposition 2.7 Let $\mathbf{a} \in \mathbb{R}^n$ and let $\mathbf{d}$ be a descent direction for $f(\mathbf{x})$ at $\mathbf{x} = \mathbf{a}$. If we apply an exact line search to $\phi(\alpha)$, obtaining $\mathbf{b} = \mathbf{a} + \alpha^*\mathbf{d}$, then

$$\nabla f(\mathbf{b})^T\mathbf{d} = 0. \tag{2.28}$$

Proof. We know that, by (2.20),

$$0 = \phi'(\alpha^*) = \mathbf{d}^T\nabla f(\mathbf{a} + \alpha^*\mathbf{d}) = \mathbf{d}^T\nabla f(\mathbf{b}). \tag{2.29}$$

□

Exercise 2.9 Check that $\nabla f(\mathbf{b})^T\nabla f(\mathbf{a}) = 0$ in Example 2.6.

This simple geometrical property of exact line searches makes them much easier to understand. However, exact line searches are expensive — they consume computer time.

2.4. *Steepest Descent — a* **BAD** *method*

How do we construct descent directions? One way is to choose

$$\mathbf{d} = -\nabla f(\mathbf{a}), \tag{2.30}$$

because then we obtain

$$\mathbf{d}^T\nabla f(\mathbf{a}) = -\|\nabla f(\mathbf{a})\|^2 < 0,$$

as required (unless we're already at a stationary point). Further, if we ask how to choose a unit vector $\mathbf{d}$ that makes $\phi'(0)$ as negative as possible, we must obtain $\mathbf{d} = -\nabla f(\mathbf{a})$, as seen in the next exercise, so explaining the name "steepest descent".

Exercise 2.10 For any two vectors $\mathbf{v}$ and $\mathbf{w}$ in $\mathbb{R}^3$, we have

$$|\mathbf{v}^T\mathbf{w}| = |\mathbf{v}\cdot\mathbf{w}| = \|\mathbf{v}\|\|\mathbf{w}\||\cos\phi|,$$

where ϕ is the angle between the two vectors $\mathbf{v}$ and $\mathbf{w}$ and $\|\mathbf{v}\|$ is defined by (3.10). Thus

$$|\mathbf{v}^T\mathbf{w}| \le \|\mathbf{v}\|\|\mathbf{w}\|, \tag{2.31}$$

and the upper bound is attained when $\mathbf{w}$ is a multiple of $\mathbf{v}$. Thus, given a unit vector $\mathbf{d}$, we make $\mathbf{d}^T\nabla f(\mathbf{a})$ as negative as possible by choosing

$$\mathbf{d} = -\frac{\nabla f(\mathbf{a})}{\|\nabla f(\mathbf{a})\|}.$$

[(2.31) is called the Cauchy–Schwarz inequality, and its n-dimensional analogue can be found in the exercises.]

Algorithm 2.2. (Steepest descent with exact line searches.) Pick $\epsilon > 0$ and choose a starting point $\mathbf{x}_0 \in \mathbb{R}^n$. Set $k = 0$.

```
While $\|\nabla f(\mathbf{x}_k)\| > \epsilon$
    Perform exact line search along $\{\mathbf{x}_k - \alpha\nabla f(\mathbf{x}_k) : \alpha \geq 0\}$ to obtain $\mathbf{x}_{k+1}$.
    Increment $k$ by one.
```

IMPORTANT: As you will see, steepest descent with exact line searches is an atrociously inefficient method. Unfortunately, unsophisticated users tend to reinvent this algorithm when faced with an optimization problem for the first time, so sealing their fate. However, studying Algorithm 2.2 is extremely useful, because it points the way to many good ideas.

Why is steepest descent so bad? The easiest way to explain this is to apply it to minimize the quadratic function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x}, \qquad \mathbf{x} \in \mathbb{R}^n, \tag{2.32}$$

where A is an $n \times n$ symmetric positive definite matrix. Our first task is to compute the gradient of this function. The following notation is highly useful.

Definition 2.7 The *Kronecker delta* is defined by $\delta_{jk} = 0$ when $j \neq k$ and $\delta_{jj} = 1$.

Example 2.8 If I denotes the $n \times n$ identity matrix, then $I_{jk} = \delta_{jk}$.

Example 2.9 If $f(\mathbf{x}) = x_p$, for $\mathbf{x} \in \mathbb{R}^n$, then $\nabla f(\mathbf{x}) = \mathbf{e}_p$, where $(\mathbf{e}_p)_j = \delta_{jp}$.

Exercise 2.11 Show that

$$\frac{\partial x_j}{\partial x_k} = \delta_{jk}.$$

Proposition 2.8 If $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x}$, for $\mathbf{x} \in \mathbb{R}^n$, where A can be any $n \times n$ symmetric matrix, then

$$\nabla f(\mathbf{x}) = A\mathbf{x}. \tag{2.33}$$

Proof. Now

$$f(\mathbf{x}) = \frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n A_{jk} x_j x_k. \tag{2.34}$$

Using the Kronecker delta notation, and remembering that A is symmetric, we have

$$\begin{aligned}
\frac{\partial f}{\partial x_q} &= \frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n A_{jk}\frac{\partial}{\partial x_q}(x_j x_k) \\
&= \frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n A_{jk}\delta_{jq}x_k + \frac{1}{2}\sum_{j=1}^n \sum_{k=1}^n A_{jk}\delta_{kq}x_j \\
&= \frac{1}{2}\sum_{k=1}^n A_{qk}x_k + \frac{1}{2}\sum_{j=1}^n A_{jq}x_j \\
&= \sum_{k=1}^n A_{qk}x_k \\
&= (A\mathbf{x})_q.
\end{aligned}$$

□

If $\mathbf{x}_k$ is the current point, then our search direction is

$$\mathbf{d}_k = -\nabla f(\mathbf{x}_k) = -A\mathbf{x}_k. \tag{2.35}$$

The exact line search condition implies the equation

$$\nabla f(\mathbf{x}_k)^T \nabla f(\mathbf{x}_k + \alpha^* \mathbf{d}_k) = 0, \tag{2.36}$$

by Proposition 2.7, whence

$$(A\mathbf{x}_k)^T \left(A\left(\mathbf{x}_k - \alpha^* A\mathbf{x}_k\right)\right) = 0,$$

or

$$\mathbf{x}_k^T A^2 \mathbf{x}_k = \alpha^* \mathbf{x}_k^T A^3 \mathbf{x}_k.$$

Thus we obtain the useful explicit formula

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha^* \mathbf{d}_k = \mathbf{x}_k - \left(\frac{\mathbf{x}_k^T A^2 \mathbf{x}_k}{\mathbf{x}_k^T A^3 \mathbf{x}_k}\right) A\mathbf{x}_k. \tag{2.37}$$

To illustrate this in two dimensions, write

$$\mathbf{x}_k = \begin{pmatrix} u_k \\ v_k \end{pmatrix}, \tag{2.38}$$

and choose A to be the diagonal matrix

$$A = \begin{pmatrix} R & 0 \\ 0 & 1 \end{pmatrix}, \tag{2.39}$$

where R is a positive constant; then we obtain

$$\begin{pmatrix} u_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} u_k \\ v_k \end{pmatrix} - \left(\frac{R^2 u_k + v_k}{R^3 u_k + v_k}\right) \begin{pmatrix} R u_k \\ v_k \end{pmatrix}. \tag{2.40}$$

Choosing a random starting vector $\mathbf{x}_0$ and a positive number R, we can then apply this equation to generate the iterates of steepest descent with exact line search, and our findings are as illustrated in lectures. The conclusion is that steepest descent performs poorly even when applied to the model problem of minimizing a quadratic function.

Exercise 2.12 Set $R = 10^3$, $\mathbf{x}_0 = \begin{pmatrix} 10 \\ 1 \end{pmatrix}$, and apply (2.40) using, for instance, a spreadsheet. You should see that many iterations are needed to approach the minimum at the origin. Try larger values of R, such as $R = 10^6$.

Proposition 2.9 Let $F(\mathbf{x}) = a + \mathbf{b}^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T M \mathbf{x}$, for $\mathbf{x} \in \mathbb{R}^n$, where M is a symmetric matrix. Then

$$\nabla F(\mathbf{x}) = \mathbf{b} + M\mathbf{x}. \tag{2.41}$$

Proof. We have

$$F(\mathbf{x}) = a + \sum_{k=1}^n b_k x_k + \frac{1}{2} \sum_{j=1}^n \sum_{k=1}^n M_{jk} x_j x_k,$$

whence, as in Proposition 2.8,

$$\frac{\partial F}{\partial x_p} = b_p + (M\mathbf{x})_p.$$

□

2.5. The Conjugate Gradient (CG) algorithm — a GOOD method

A seemingly trivial change to the steepest descent method provides the following excellent method for minimizing a general function whose first partial derivatives are available.

Algorithm 2.3. (Fletcher–Reeves CG with exact line searches.) Pick $\epsilon > 0$ and choose a starting point $\mathbf{x}_0 \in \mathbb{R}^n$.

```
Set $k = 0$ and $\mathbf{d}_0 = \nabla f(\mathbf{x}_0)$.
While $\|\nabla f(\mathbf{x}_k)\| > \epsilon$
    Perform exact line search along $\{\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k) : \alpha \geq 0\}$ to obtain $\mathbf{x}_{k+1}$.
    Increment $k$ by one.
    Set
```

$$\mathbf{d}_k = -\nabla f(\mathbf{x}_k) + \Big(\frac{\|\nabla f(\mathbf{x}_k)\|^2}{\|\nabla f(\mathbf{x}_{k-1})\|^2}\Big)\mathbf{d}_{k-1}. \tag{2.42}$$

This algorithm has the remarkable property that, when applied to a quadratic function in n dimensions, it reaches the minimum in precisely n steps (ignoring computer rounding errors). Thus the seemingly innocuous difference between (2.42) and steepest descent leads to stunningly superior performance in practice.

2.6. Newton's method for minimizing functions of n variables

The steepest descent method uses only first partial derivatives of $f(\mathbf{x})$. What happens if second partial derivatives are also available? Given a point $\mathbf{x}_k \in \mathbb{R}^n$, we want to generate a new point $\mathbf{x}_{k+1}$ that is closer to a local minimum of $f(\mathbf{x})$. One way is to minimize the quadratic approximation

$$q(\mathbf{h}) = f(\mathbf{x}_k) + \mathbf{h}^T \nabla f(\mathbf{x}_k) + \frac{1}{2}\mathbf{h}^T D^2 f(\mathbf{x}_k)\mathbf{h},$$

which is possible if the second derivative matrix $D^2 f(\mathbf{x}_k)$ is positive definite — this will typically be the case if we are sufficiently close to a local minimum. Thus we solve

$$0 = \nabla q(\mathbf{h}_k) = \nabla f(\mathbf{x}_k) + D^2 f(\mathbf{x}_k)\mathbf{h}_k,$$

obtaining

$$\mathbf{h}_k = -\Big(D^2 f(\mathbf{x}_k)\Big)^{-1} \nabla f(\mathbf{x}_k)$$

and

$$\mathbf{x}_{k+1} \equiv \mathbf{x}_k + \mathbf{h}_k = \mathbf{x}_k - \Big(D^2 f(\mathbf{x}_k)\Big)^{-1} \nabla f(\mathbf{x}_k).$$

Practical Computation: We **DON'T** calculate the inverse matrix, because it is easier, and more accurate, to solve the linear system

$$D^2 f(\mathbf{x}_k)\mathbf{h}_k = -\nabla f(\mathbf{x}_k).$$

This algorithm converges quadratically when it converges. More precisely, if $\mathbf{x}^*$ is the local minimum, then we obtain

$$\|\mathbf{e}_{k+1}\| = \mathcal{O}(\|\mathbf{e}_k\|^2),$$

where $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$. Unfortunately, Newton's method has many disadvantages:

(i) It's completely unreliable. Specifically, there is no way to predict whether a given starting point will lead to convergence to the desired local minimum for a general function $f(\mathbf{x})$.
(ii) If one of the second derivative matrices $D^2 f(\mathbf{x}_k)$ is not positive definite, then the algorithm will fail. [Exercise: Why?]
(iii) Each step requires the solution of n linear equations in n unknowns. This requires $\mathcal{O}(n^3)$ operations — extremely expensive.
(iv) We must be able to calculate the (roughly) $\frac{1}{2}n^2$ second partial derivatives of $f(\mathbf{x})$. This is a long tedious calculation for a human. Further, although algebraic manipulation software (such as Maple or Mathematica) can sometimes be useful, it can be unreliable.

Exercise 2.13 A Pentium III 500 MHz processor can perform about 10^8 arithmetic operations per second. It can therefore solve n linear equations in n unknowns in (at most) $T(n) = 10^{-7}n^3$ seconds. Find $T(10^3)$ and $T(10^6)$. [One year contains about 3×10^7 seconds.]

Having said that Newton's method is unreliable, let me add that there are excellent reliable methods that preserve the quadratic convergence of Newton's method. These are the *variable metric* methods, where a positive definite symmetric matrix is used to approximate the second derivative matrix. We begin by letting B_0 be the identity matrix. Each subsequent step minimizes the quadratic approximation

$$Q(\mathbf{h}) = f(\mathbf{x}_k) + \mathbf{h}^T \nabla f(\mathbf{x}_k) + \frac{1}{2}\mathbf{h}^T B_k \mathbf{h}. \tag{2.43}$$

Setting $\nabla Q(\mathbf{h}_k) = 0$, we construct $\mathbf{h}_k$ by solving the linear system

$$B_k \mathbf{h}_k = -\nabla f(\mathbf{x}_k). \tag{2.44}$$

The matrix B_k is then modified to form the second derivative approximation at the new point, denoted B_{k+1}. Further details of variable metric methods are beyond the scope of this course. [But see Chapter 10 of *Numerical Recipes*.]

Another algorithm based on Newton's method is the *modified Newton method.* Here we let

$$\mathbf{d} = \pm\Big(D^2 f(\mathbf{x}_k)\Big)^{-1} \nabla f(\mathbf{x}_k) \tag{2.45}$$

be the search direction for an exact line search, choosing the sign in (2.45) to ensure that $\mathbf{d}$ is a descent direction. Unfortunately, modified Newton can also fail when the second derivative matrix isn't positive definite, as you will see in exercises.

2.7. Least squares problems

In many applications, we want to fit experimental data using a mathematical model depending on several parameters $\mathbf{x} \in \mathbb{R}^n$. More formally, we want to minimize the sum of squares

$$S(\mathbf{x}) = \sum_{\ell=1}^{M} (r_\ell(\mathbf{x}))^2, \tag{2.46}$$

where $r_\ell(\mathbf{x})$ is the ℓ^{th} residual. If every $r_\ell(\mathbf{x})$ is a linear function, then we say that minimizing (2.46) is a *linear least squares problem.* If some of the $r_\ell(\mathbf{x})$ are nonlinear functions, then we have a *nonlinear least squares problem.*

Example 2.10 If the data $(t_1, y_1), (t_2, y_2), \ldots, (t_M, y_M)$ lie approximately on a line, then we can estimate the linear equation by minimizing the sum of squares

$$S(\mathbf{x}) = \sum_{\ell=1}^{M} \Big(y_\ell - x_2 t_\ell - x_1 \Big)^2. \tag{2.47}$$

The resulting line has equation $y(t) = x_2^* t + x_1^*$ and the residual functions are

$$r_\ell(\mathbf{x}) = y_\ell - x_2 t_\ell - x_1, \qquad 1 \le \ell \le M. \tag{2.48}$$

This is a *linear* least squares problem.

Proposition 2.10 If $S(\mathbf{x})$ is defined by (2.46), then

$$\nabla S(\mathbf{x}) = 2 \sum_{\ell=1}^{M} r_\ell(\mathbf{x}) \nabla r_\ell(\mathbf{x}) \tag{2.49}$$

and

$$D^2 f(\mathbf{x}) = 2 \sum_{\ell=1}^{M} \Big[\nabla r_\ell(\mathbf{x}) \Big(\nabla r_\ell(\mathbf{x}) \Big)^T + r_\ell(\mathbf{x}) D^2 r_\ell(\mathbf{x}) \Big]. \tag{2.50}$$

Proof. We have

$$\frac{\partial S}{\partial x_k} = 2 \sum_{\ell=1}^{M} r_\ell \frac{\partial r_\ell}{\partial x_k}, \tag{2.51}$$

whence (2.50). Differentiating (2.51), we find

$$\frac{\partial^2 S}{\partial x_j \partial x_k} = 2 \sum_{\ell=1}^{M} \Big[\frac{\partial r_\ell}{\partial x_j} \frac{\partial r_\ell}{\partial x_k} + r_\ell \frac{\partial^2 r_\ell}{\partial x_j \partial x_k} \Big]. \tag{2.52}$$

Now, for any pair of column vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, Proposition 3.1 tells us that

$$\big(\mathbf{u}\mathbf{v}^T \big)_{jk} = u_j v_k, \qquad 1 \le j, k \le m.$$

Hence (2.52) implies (2.50). □

When the fit is good, each residual function $r_\ell(\mathbf{x})$ is small. If we have $r_\ell(\mathbf{x}) \approx 0$, for $1 \le \ell \le M$, then the formula for the second derivative matrix given in (2.50) can be replaced by the approximation

$$D^2 S(\mathbf{x}) \approx 2 \sum_{\ell=1}^{M} \nabla r_\ell(\mathbf{x}) \Big(\nabla r_\ell(\mathbf{x}) \Big)^T, \tag{2.53}$$

when $\mathbf{x}$ is sufficiently close to a local minimum $\mathbf{x}^*$. In other words, instead of computing the $M^2/2$ second partial derivatives, we can estimate them using only the first partial derivatives. Equation (2.53) can then be used in Newton's method, the combination of the two being called the Gauss–Newton method for nonlinear least squares problems.

Example 2.11 Returning to the linear least squares problem (2.47), we see that

$$\nabla r_\ell(\mathbf{x}) = \begin{pmatrix} -1 \\ -t_\ell \end{pmatrix} \qquad \text{and} \qquad D^2 r_\ell(\mathbf{x}) = 0. \tag{2.54}$$

Hence, by Proposition 2.10,

$$\nabla S(\mathbf{x}) = -2 \sum_{\ell=1}^{M} (y_\ell - x_2 t_\ell - x_1) \begin{pmatrix} 1 \\ t_\ell \end{pmatrix} \tag{2.55}$$

and

$$D^2 S(\mathbf{x}) = 2 \sum_{\ell=1}^{M} \begin{pmatrix} 1 \\ t_\ell \end{pmatrix} \begin{pmatrix} 1 & t_\ell \end{pmatrix}. \tag{2.56}$$

Thus the only stationary point occurs when

$$\sum_{\ell=1}^{N} (y_\ell - x_2 t_\ell - x_1) \begin{pmatrix} 1 \\ t_\ell \end{pmatrix} = 0, \tag{2.57}$$

that is,

$$\begin{pmatrix} M & \sum_{\ell=1}^M t_\ell \\ \sum_{\ell=1}^M t_\ell & \sum_{\ell=1}^M t_\ell^2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sum_{\ell=1}^M y_\ell \\ \sum_{\ell=1}^M y_\ell t_\ell \end{pmatrix}. \tag{2.58}$$

The stationary point is a local minimum because the second derivative matrix is positive definite. Indeed, for any vector $\mathbf{v} \in \mathbb{R}^2$, we have

$$\mathbf{v}^T D^2 S(\mathbf{x}) \mathbf{v} = 2 \sum_{\ell=1}^{M} \mathbf{v}^T \begin{pmatrix} 1 \\ t_\ell \end{pmatrix} \begin{pmatrix} 1 & t_\ell \end{pmatrix} \mathbf{v} = 2 \sum_{\ell=1}^{M} \left(\mathbf{v}^T \begin{pmatrix} 1 \\ t_\ell \end{pmatrix} \right)^2 \geq 0, \tag{2.59}$$

with equality if and only if $\mathbf{v}$ is orthogonal to every vector $\begin{pmatrix} 1 \\ t_\ell \end{pmatrix}$, for $1 \leq \ell \leq M$, which implies $\mathbf{v} = 0$ when the numbers $t_1, t_2, \ldots, t_M$ are all different.

2.8. Finite difference methods

Calculating first partial derivatives can be tedious. Can we do without them? One way is to approximate derivative by *finite differences*. For example, given a univariate function $g(x)$, we have

$$g(x+h) = g(x) + hg'(x) + \frac{1}{2}h^2 g^{(2)}(x) + \frac{1}{6}h^3 g^{(3)}(x) + \frac{1}{24}h^4 g^{(4)}(x) + \mathcal{O}(h^5)$$

and

$$g(x-h) = g(x) - hg'(x) + \frac{1}{2}h^2 g^{(2)}(x) - \frac{1}{6}h^3 g^{(3)}(x) + \frac{1}{24}h^4 g^{(4)}(x) + \mathcal{O}(h^5).$$

Therefore

$$\frac{g(x+h) - 2g(x) + g(x-h)}{h^2} = g^{(2)}(x) + \frac{1}{12}h^2 g^{(4)}(x) = g^{(2)} + \mathcal{O}(h^2). \tag{2.60}$$

Equation (2.60) is a called a *finite difference approximation* to the second derivative.

Exercise 2.14 Let $g(x) = x \cot(x/2)$. Use (2.60) to estimate $g^{(2)}(0)$.

There's a similar trick for first derivatives.

Exercise 2.15 Show that

$$g'(x) = \left(\frac{g(x+h) - g(x-h)}{2h}\right) + \mathcal{O}(h^2). \tag{2.61}$$

We can use (2.61) to construct approximations to $\nabla f(\mathbf{x})$ for a multivariate function $f(\mathbf{x})$. We use

$$\frac{\partial f}{\partial x_j}(\mathbf{a}) = \left(\frac{f(\mathbf{a} + h\mathbf{e}_j) - f(\mathbf{a} - h\mathbf{e}_j)}{2h}\right) + \mathcal{O}(h^2), \qquad 1 \le j \le n, \tag{2.62}$$

where $\mathbf{e}_j$ is the jth coordinate vector, that is $(\mathbf{e}_j)_k = \delta_{jk}$.

Finite difference approximations like (2.62) are also extremely important when testing that software for calculating partial derivatives is correct, because errors occur all too easily in algebra and programming.

3. Mathematical Background

3.1. Some linear algebra

We use A_{jk} to denote the element in row j and column k of the matrix A. If $\mathbf{x} \in \mathbb{R}^n$ is a (column) vector, we use both x_k and $(\mathbf{x})_k$ to denote the kth element (or component) of $\mathbf{x}$.

If

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{pmatrix} \tag{3.1}$$

is an $n \times n$ matrix and

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n \tag{3.2}$$

then their *matrix-vector* product $\mathbf{y} = A\mathbf{x}$ is given by the equations

$$\begin{aligned} y_1 &= A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n \\ y_2 &= A_{21}x_1 + A_{22}x_2 + \cdots + A_{2n}x_n \\ &\vdots \\ y_n &= A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nn}x_n. \end{aligned} \tag{3.3}$$

$$\tag{3.4}$$

It's boring to write these expressions in full, so we use a briefer notation:

$$y_j = \sum_{k=1}^{n} A_{jk}x_k, \qquad 1 \le j \le n, \tag{3.5}$$

or

$$(A\mathbf{x})_j = \sum_{k=1}^{n} A_{jk}x_k, \qquad 1 \le j \le n. \tag{3.6}$$

Example 3.1 If A is a $p \times q$ matrix and B is a $q \times r$ matrix, then their matrix product AB is given by

$$(AB)_{jk} = \sum_{\ell=1}^{q} A_{j\ell} B_{\ell k}, \qquad 1 \le j \le p, 1 \le k \le r.$$

Definition 3.1 The transpose A^T of the $p \times q$ matrix A is the $q \times p$ matrix defined by

$$\left(A^T\right)_{jk} = A_{kj}, \qquad 1 \le j \le q, 1 \le k \le p. \tag{3.7}$$

We say that A is *symmetric* when $A = A^T$.

Proposition 3.1 Let $\mathbf{u}$ and $\mathbf{v}$ be (column) vectors in $\mathbb{R}^n$. Then $M = \mathbf{u}\mathbf{v}^T$ is the $n \times n$ matrix given by

$$M_{jk} u_j v_k, \qquad \text{for } 1 \le j \le n, 1 \le k \le n. \tag{3.8}$$

Proof. This is just the usual definition of matrix multiplication, thinking of $\mathbf{u}$ as a $n \times 1$ matrix and $\mathbf{v}$ as a $1 \times n$ matrix. □

Example 3.2

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \begin{pmatrix} 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 4 & 5 & 6 \\ 8 & 10 & 12 \\ 12 & 15 & 18 \end{pmatrix}.$$

We shall also need to measure the *length* of a vector.

Definition 3.2 The *length*, or *norm*, of the vector

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ v_n \end{pmatrix} \in \mathbb{R}^n \tag{3.9}$$

is defined by the number ("norm $\mathbf{u}$")

$$\|\mathbf{u}\| = \sqrt{u_1^2 + u_2^2 + \cdots + u_n^2}. \tag{3.10}$$

Example 3.3 If $\mathbf{x} = \begin{pmatrix} 1 & -3 & 0 & 5 \end{pmatrix}^T$, then $\|\mathbf{x}\| = \sqrt{35}$.

Proposition 3.2 If A is any $n \times n$ symmetric matrix and $\mathbf{x} \in \mathbb{R}^n$, then the *quadratic function*

$$\mathbf{x}^T A \mathbf{x} = \sum_{j=1}^{n} \sum_{k=1}^{n} A_{jk} x_j x_k. \tag{3.11}$$

Proof. Setting $\mathbf{y} = A\mathbf{x}$, we see that

$$\mathbf{x}^T A \mathbf{x} = \mathbf{x}^T \mathbf{y} = \sum_{j=1}^{n} x_j y_j.$$

Further, using (3.6), we have

$$\mathbf{x}^T A \mathbf{x} = \sum_{j=1}^{n} x_j \sum_{k=1}^{n} A_{jk} x_k = \sum_{j=1}^{n} \sum_{k=1}^{n} A_{jk} x_j x_k.$$

□

Example 3.4 If

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 3 \end{pmatrix}$$

then

$$\begin{pmatrix} x \\ y \end{pmatrix}^T A \begin{pmatrix} x \\ y \end{pmatrix} = 2x^2 + 3y^2 - 2xy.$$

Exercise 3.1 Calculate

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}^T \begin{pmatrix} 3 & -1 & 2 \\ -1 & -4 & 5 \\ 2 & 5 & 6 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}.$$

3.2. Contours and gradients

Let $f(x,y)$ be a function of two variables. Then a *contour* of $f(x,y)$ is simply a curve on which $f(x,y)$ is constant. I'm sure you've all seen contours on maps — just think of $z = f(x,y)$ as the land's height at location (x,y).

Proposition 3.3 Gradient vectors are orthogonal to contours.

A fully rigorous proof is not needed. Here's a sketch. Let $\mathbf{r}$ and $\mathbf{r}+\mathbf{h}$ be close points on the same contour, so that $\|\mathbf{h}\|$ is small and $f(\mathbf{r}) = f(\mathbf{r}+\mathbf{h})$. Then we have

$$f(\mathbf{r}+\mathbf{h}) = f(\mathbf{r}) + \mathbf{h}^T \nabla f(\mathbf{r}) + \mathcal{O}(\|\mathbf{h}\|^2), \tag{3.12}$$

or

$$0 = \mathbf{h}^T \nabla f(\mathbf{r}) + \mathcal{O}(\|\mathbf{h}\|^2). \tag{3.13}$$

Now let's introduce the unit vector

$$\mathbf{u} = \frac{\mathbf{h}}{\|\mathbf{h}\|},$$

so that $\mathbf{h} = h\mathbf{u}$, where $h \equiv \|\mathbf{h}\|$. Then (3.13) becomes

$$0 = \mathbf{u}^T \nabla f(\mathbf{r}) + \mathcal{O}(h). \tag{3.14}$$

Further, as $h \to 0$, $\mathbf{u}$ tends to the tangent vector to the contour passing through the point $\mathbf{r}$; thus the contour's tangent is orthogonal to the gradient vector.

3.3. Ellipses

The equation

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = c^2, \tag{3.15}$$

where a, b are fixed positive numbers and $c \geq 0$, represents an *ellipse*. It's easy to sketch this curve if you substitute

$$X = \frac{x}{a} \qquad \text{and} \qquad Y = \frac{y}{b}, \tag{3.16}$$

because then the point (X, Y) satisfies

$$X^2 + Y^2 = c^2, \tag{3.17}$$

which you should recognize as the equation of a circle of radius c centred at the origin. Thus an ellipse is simply a squashed circle. This is useful when sketching contours of quadratic functions in two dimensions.