VI.
Nonlinear Optimization

Robert Fourer

Department of Industrial Engineering and Management Sciences
Northwestern University
Evanston, Illinois 60208-3119, U.S.A.

(847) 491-3151
4er@iems.northwestern.edu
http://www.iems.northwestern.edu/~4er/

Copyright © 1989–2004 Robert Fourer
17. Extending Linear to Nonlinear Optimization

For optimization involving nonlinear functions for which derivatives can be computed — what are called smooth functions — there is an extensive body of theory and methods. Smooth nonlinear optimization is harder than linear programming in some respects, but surprisingly easier in others.

The smooth nonlinear case tends to be harder than the linear case because it allows the decision variables to appear in a much broader variety of functions — such as sines, logarithms, powers, and ratios — in any combination. As a result the optimization is more complex mathematically and more difficult computationally. Many of the ideas and methods of linear programming can be extended to the nonlinear case, however; this chapter sketches an introduction to extensions of such kinds.

Because nonlinear functions encompass such a wide variety of forms, however, interesting nonlinear optimization problems need not have large numbers of variables and constraints. Indeed, optimizing a nonlinear function can be a challenge in the absence of any constraints at all. The following two chapters focus on unconstrained optimization of this kind. In that simpler context we can more easily introduce the fundamentals of nonlinear optimality conditions and methods. Finally we give a detailed presentation of nonlinear optimization with constraints in a later chapter (not ready yet).

17.1 Generalizing the linear optimality conditions

We turn first to the geometric view of optimality for linear programming that we developed in Part II, and consider how that view might be extended to encompass smooth nonlinear objective and constraint functions. The key is to consider the complementary slackness conditions for optimality that we derived in Part III.

Our geometric forms for a linear program and its dual (Section 15.2) are

\[
\begin{align*}
\text{Minimize} & \quad cx \\
\text{Subject to} & \quad Ax \geq b \\
\text{Maximize} & \quad \pi b \\
\text{Subject to} & \quad \pi A = c \\
& \quad \pi \geq 0
\end{align*}
\]

We let \(a^i\) denoting the \(i\)th row of \(A\). Then the complementary slackness property can be interpreted as saying that \(x^*\) is optimal for this linear program if and only if there also exists a \(\pi^*\) such that the following conditions are satisfied:

\[\begin{align*}
\triangleright & \quad \text{Primal feasibility: } a^i x^* \geq b_i, \text{ for each } i = 1, \ldots, m \\
\triangleright & \quad \text{Dual feasibility: } \sum_{i=1}^{m} \pi_i^* a^i = c, \quad \pi^* \geq 0 \\
\triangleright & \quad \text{Complementarity: } \\
& \quad \text{Either } a^i x^* = b_i \text{ or } \pi_i^* = 0 \text{ (or both), for each } i = 1, \ldots, m
\end{align*}\]

For each \(i\) having \(a^i x^* > b_i\), however, complementarity requires \(\pi_i^* = 0\), and so the corresponding \(\pi_i^* a^i\) term is zero in the sum. Thus if we let

\[B \equiv \{i = 1, \ldots, m : a^i x^* = b_i\}\]
denote the set of \textit{active} constraints at $x^*$, we can also think of the optimality conditions as
\[ a^i x^* \geq b_i, \text{ for each } i = 1, \ldots, m \]
\[ \sum_{i \in B} \pi^*_i a^i = c \text{ and } \pi^*_i \geq 0, \text{ for each } i \in B \]

We do not have to state explicitly that $\pi^*_i = 0$ for $i \notin B$, since we have left all such $\pi^*_i$ values out of the statement of the conditions entirely. Indeed, the inactive constraints must be irrelevant to the question of whether $x^*$ satisfies the optimality conditions, because the answer would be the same if all the inactive constraints were dropped from the problem. We should expect $\pi^*_i$ to play a significant role in the optimality conditions only for $i \in B$.

In the linear case we know that $i \in B$ can always be chosen to correspond to $m$ constraint rows $a^i$ that are linearly independent, so that $\sum_{i \in B} \pi^*_i a^i = c$ has a unique solution. Figure 17-1 illustrates these conditions for a two-variable example:

\[
\begin{align*}
\text{Minimize} & \quad 6x_1 + x_2 \\
\text{Subject to} & \quad 9x_1 - 6x_2 \geq -9 \\
& \quad 3x_1 + x_2 \geq 6 \\
& \quad -x_1 + 4x_2 \geq 0
\end{align*}
\]

In our geometric view of linear programming we say that $x^* = (1,3)$ is an extreme point of the feasible region. The line $6x_1 + x_2 = 9$ just passes through this point, demonstrating that it is optimal for the objective $6x_1 + x_2$. 

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure17-1.png}
\caption{Optimal extreme point of a linear program.}
\end{figure}
To confirm optimality algebraically in terms of the active constraints, we observe that \( x^* = (1, 3) \) satisfies \( a^i x^* \geq b_i, \ i = 1, \ldots, 3 \) with \( \mathcal{B} = \{1, 2\} \):

\[
\begin{align*}
a^1 x^* &= 9 \cdot 1 - 6 \cdot 3 = -9 \quad \Rightarrow \quad 1 \in \mathcal{B} \\
a^2 x^* &= 3 \cdot 1 + 1 \cdot 3 = 6 \quad \Rightarrow \quad 2 \in \mathcal{B} \\
a^3 x^* &= -1 \cdot 1 + 4 \cdot 3 > 0
\end{align*}
\]

Hence in this case \( \sum_{i \in \mathcal{B}} \pi_i^* a^i = c \) becomes \( \pi_1 a^1 + \pi_2 a^2 = c \), which is satisfied by \( (\pi_1^*, \pi_2^*) = (1/9, 5/3) \geq 0 \):

\[
\begin{bmatrix}
1/9 & 5/3 \\
9 & -6 \\
3 & 1
\end{bmatrix}
\begin{bmatrix}
9 \\
3 \\
1
\end{bmatrix}
= \begin{bmatrix}
6 \\
1
\end{bmatrix}.
\]

It follows that \( x^* = (1, 3) \) and \( \pi^* = (1/9, 5/3, 0) \) give a solution to our original optimality conditions. In the geometry of Figure 17–1, the vectors \( a^1 = (9, -6) \) and \( a^2 = (3, 1) \) are the directions of the “constraint normals” — the arrows perpendicular to the constraint lines in the figure. Similarly the vector \( c \) gives the direction of the “objective normal” represented by the arrow perpendicular to the objective line.

The conditions \( \pi_1 a^1 + \pi_2 a^2 = c \), \( \pi_1 \geq 0 \), \( \pi_2 \geq 0 \) thus assert that the objective normal lies in the cone defined by all nonnegative combinations of the constraint normals. It makes sense that \( a^3 \) does not figure in these conditions, because the optimality conditions are local to the point \( x^* = (1, 3) \). The constraint \( a^3 x \geq b_3 \) is not active at \( x^* \); it could be dropped from the linear program without any effect on the optimality of \( x^* \), and we would not expect it to appear in any optimality conditions for that solution.

Consider now the example in Figure 17–2, which depicts the following optimization problem having the same linear objective but nonlinear constraints:

\[
\begin{align*}
\text{Minimize} & \quad cx = 6x_1 + x_2 \\
\text{Subject to} & \quad a^1(x) = 9x_1 - x_2^2 \geq 0 \\
& \quad a^2(x) = x_1x_2 \geq 3 \\
& \quad a^3(x) = -3 \log x_1 + 4x_2 \geq 0
\end{align*}
\]

As in the linear case, \( x^* = (1, 3) \) is an extreme point of the feasible region. In this context the set of active constraints is

\[
\mathcal{B} \equiv \{ i = 1, \ldots, m : a^i(x^*) = b_i \}
\]

and for \( x^* = (1, 3) \) we have again \( \mathcal{B} = \{1, 2\} \):

\[
\begin{align*}
a^1(1, 3) &= 9 \cdot 1 - 3^2 = 0 \quad \Rightarrow \quad 1 \in \mathcal{B} \\
a^2(1, 3) &= 1 \cdot 3 = 3 \quad \Rightarrow \quad 2 \in \mathcal{B} \\
a^3(1, 3) &= -3 \cdot \log 1 + 4 \cdot 3 > 0
\end{align*}
\]

The line \( 6x_1 + x_2 = 19 \) just passes through \( (1, 3) \), making it a locally optimal point for the objective \( 6x_1 + x_2 \).

Again we can confirm optimality by observing that the objective normal \( c \) lies within the cone defined by the normals to the active constraints. But now the constraints are nonlinear, so the normals are defined by their derivatives.
Figure 17–2. Minimization of a linear function subject to nonlinear constraints. A local optimum occurs at the extreme point \((1, 3)\), where the objective function gradient lies in the cone defined by the constraint function gradients.

Specifically, the normal to a nonlinear constraint curve at a point is the normal to its tangent line at that point, which is given by the vector of partial derivatives of the constraint function with respect to the variables. We denote this vector — the \textit{gradient} of \(a^i(x)\) with respect to \(x\) — by \(\nabla a^i(x)\); in our two-dimensional example,

\[
\nabla a^i(x) = \left( \frac{\partial a^i(x)}{\partial x_1}, \frac{\partial a^i(x)}{\partial x_2} \right).
\]

The optimality conditions are a straightforward generalization of the linear ones, but with \(a^i(x^*)\) and \(\pi_i^* a^i\) replaced by \(a^i(x^*)\) and \(\pi_i^* \nabla a^i(x^*)\):

\begin{itemize}
  \item \(a^i(x^*) \geq b_i\), for each \(i = 1, \ldots, m\)
  \item \(\sum_{i=1}^m \pi_i^* \nabla a^i(x^*) = c\), \(\pi_i^* \geq 0\)
  \item Either \(a^i(x^*) = b_i\) or \(\pi_i^* = 0\) (or both), for each \(i = 1, \ldots, m\)
\end{itemize}

These conditions in terms of the active set are

\begin{itemize}
  \item \(a^i(x^*) \geq b_i\), for each \(i = 1, \ldots, m\)
  \item \(\sum_{i \in B} \pi_i^* \nabla a^i(x^*) = c\) and \(\pi_i^* \geq 0\), for each \(i \in B\)
\end{itemize}

In our example, the gradients of the active constraints at \(x^*\) are

\[
\nabla a^1(x^*) = \left( \frac{\partial a^1(x^*)}{\partial x_1}, \frac{\partial a^1(x^*)}{\partial x_2} \right) = (9, -2x_2^*) = (9, -6),
\]
\[
\nabla a^2(x^*) = \left( \frac{\partial a^2(x^*)}{\partial x_1}, \frac{\partial a^2(x^*)}{\partial x_2} \right) = (x_2^*, x_1^*) = (3, 1),
\]
which are the same as $a^1$ and $a^2$ from the linear example, so we know that the conditions $\pi^*_1 \nabla a^1(x^*) + \pi^*_2 \nabla a^2(x^*) = c$ are satisfied by the same dual values $\pi^*_1 = 1/9$ and $\pi^*_2 = 5/3$. The value of $\nabla a^3(x^*)$ does not figure in the conditions because again constraint 3 is not active.

We can further extend this reasoning to the case of a nonlinear objective:

\[
\text{Minimize } \quad c(x) = 3x_1^2 + (x_2 - 5/2)^2 \\
\text{Subject to } \quad a^1(x) = 9x_1 - x_2^2 \geq 0 \\
\quad \quad \quad a^2(x) = x_1x_2 \geq 3 \\
\quad \quad \quad a^3(x) = -3\log x_1 + 4x_2 \geq 0
\]

In Figure 17–3 we see that the contour $c(x) = 3x_1^2 + (x_2 - 5/2)^2 = 3\frac{1}{4}$ — no longer a line as in the linear case — just touches the feasible region at $x^* = (1, 3)$, which must therefore be a locally optimal point. Now it is the objective normal, given by the objective gradient $\nabla c(x)$, that lies in the cone defined by the active constraint normals. In our example,

\[
\nabla c(x^*) = (\partial c(x^*)/\partial x_1, \partial c(x^*)/\partial x_2) = (6x_1, 2x_2 - 5) = (6, 1).
\]

Thus the objective normal at $x^*$ is the same as in the preceding linear-objective example, and $\pi^*_1 \nabla a^1(x^*) + \pi^*_2 \nabla a^2(x^*) = \nabla c(x)$ is satisfied by the same dual values $\pi^*_1 = 1/9$ and $\pi^*_2 = 5/3$. 

**Figure 17–3.** Minimization of a nonlinear function subject to nonlinear constraints. Again, a local optimum occurs at the extreme point $(1, 3)$, where the objective function gradient lies in the cone defined by the constraint function gradients.
17.2 General necessary conditions for nonlinear optimization

Substituting $\nabla c(x)$ for $c$ in our previous conditions, we now have a general optimality property for non-linearly-constrained nonlinear optimization. To state it properly we must formally define the kind of optimality involved:

**Definition 17–1** A point $x^*$ is **locally** optimal if it yields a better objective value than any other nearby feasible solution. Specifically, $x^*$ is locally optimal for $c(x)$ if there exists $\epsilon > 0$ such that $c(x) \geq c(x^*)$ for all feasible $x$ with $\|x - x^*\| \leq \epsilon$.

**Property 17–1 (Karush-Kuhn-Tucker)** If $x^*$ is locally optimal for

\[
\begin{align*}
\text{Minimize} & \quad c(x) \\
\text{Subject to} & \quad a_i(x) \geq b_i, \quad i = i, \ldots, m
\end{align*}
\]

then there exists a $\pi^*$ such that the following conditions are satisfied:

- $a^i(x^*) \geq b_i$, for each $i = 1, \ldots, m$
- $\sum_{i=1}^{m} \pi^*_i \nabla a^i(x^*) = \nabla c(x^*)$, $\pi^*_i \geq 0$
- Either $a^i(x^*) = b_i$ or $\pi^*_i = 0$ (or both), for each $i = 1, \ldots, m$

Equivalently,

- $a^i(x^*) \geq b_i$, for each $i = 1, \ldots, m$
- $\sum_{i \in B} \pi^*_i \nabla a^i(x^*) = \nabla c(x^*)$ and $\pi^*_i \geq 0$, for each $i \in B$

where $B \equiv \{i = 1, \ldots, m : a^i(x^*) = b_i\}$.

In contrast to the linear complementary slackness conditions, these are only **necessary** conditions for optimality. For $x^*$ to be locally optimal, these conditions must hold. But these conditions can also hold at points that are not locally optimal, as some of the examples in the next section will show. We will also see that a nonlinear optimization problem can have distinctly separate locally optimal solutions that achieve different objective values.

Variations on the problem form we have been using, such as equality constraints and nonnegative variables, lead to variations in the optimality conditions. The general rules come out to be much the same as for linear programming, except with the element of the gradients — the partial derivatives of the objective and the constraint functions — introduced in the appropriate places. The rules for a minimization are as follows:

<table>
<thead>
<tr>
<th>In the minimization problem:</th>
<th>In the necessary conditions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a^i(x^*) \geq b_i$</td>
<td>$\pi^*_i \geq 0$</td>
</tr>
<tr>
<td>$a^i(x^*) = b_i$</td>
<td>$\pi^*_i$ unrestricted</td>
</tr>
<tr>
<td>$a^i(x^*) \leq b_i$</td>
<td>$\pi^*_i \leq 0$</td>
</tr>
<tr>
<td>$x^*_j \geq 0$</td>
<td>$\sum_{i=1}^{m} \pi^<em>_i \partial a^i(x^</em>)/\partial x_j \leq \partial c(x^*)/\partial x_j$</td>
</tr>
<tr>
<td>$x^*_j$ unrestricted</td>
<td>$\sum_{i=1}^{m} \pi^<em>_i \partial a^i(x^</em>)/\partial x_j = \partial c(x^*)/\partial x_j$</td>
</tr>
<tr>
<td>$x^*_j \leq 0$</td>
<td>$\sum_{i=1}^{m} \pi^<em>_i \partial a^i(x^</em>)/\partial x_j \geq \partial c(x^*)/\partial x_j$</td>
</tr>
</tbody>
</table>
In addition there are the following complementarity conditions:

*If* $a^i(x) \geq b_i$ *or* $a^i(x) \leq b_i$ *in the constraints:*

Either $a^i(x^*) = b_i$ *or* $\pi^*_i = 0$ *(or both), for each* $i = 1, \ldots, m$.

*If* $x_j \geq 0$ *or* $x_j \leq 0$ *in the constraints:*

Either $x_j = 0$ *or* $\sum_{i=1}^m \pi^*_i \partial a^i(x)/\partial x_j = \partial c(x)/\partial x_j$ *(or both), for each* $j = 1, \ldots, n$.

Where the constraints have $a^i(x) = b_i$ *or* $x_j$ *unrestricted*, complementarity is automatically satisfied by $a^i(x) = b_i$ *or by* $\sum_{i=1}^m \pi^*_i \partial a^i(x)/\partial x_j = \partial c(x)/\partial x_j$, *and no explicit complementarity condition is needed.*

The rules for a maximization are the same, but with the inequalities reversed in the necessary conditions.

**Example 17-1.** In deriving interior-point methods for linear programming, we made use of the following log-barrier problem:

Minimize $cx - \mu \sum_{j=1}^n \log x_j$

Subject to $Ax = b$

In this case the gradient elements $\partial c(x)/\partial x_j = c_j - \mu/x_j$; since the constraints are linear, each $a^i(x) = a^i x$, and $\partial a^i(x)/\partial x_j = a^i_j$. The constraints are all equalities and the variables $x_j$ are all unrestricted, so the above rules says that the necessary conditions should contain

$a^i x^* = b_i$, for each $i = 1, \ldots, m$

$\sum_{i=1}^m \pi^*_i a^i_j = c_j - \mu/x^*_j$, for each $j = 1, \ldots, n$

and no explicit complementarity conditions. If we multiply through by $x^*_j$ in each condition of the second set, and define $s^*_j$ to be $c_j - \sum_{i=1}^m \pi^*_i a^i_j$, then we have the following equivalent form:

$a^i x^* = b_i$, for each $i = 1, \ldots, m$

$\sum_{i=1}^m \pi^*_i a^i_j + s^*_j = c_j$, for each $j = 1, \ldots, n$

$s^*_j x^*_j = \mu$, for each $j = 1, \ldots, n$

These are identical to the conditions that we stated in Section 15.2, except that there we used a more concise matrix terminology.

**Example 17-2.** A *quadratic program* is given by the following generalization of a linear program:

Minimize $\frac{1}{2} x^T Q x + cx$

Subject to $Ax = b$

$x \geq 0$

A small example makes it clear what the terms in $x^T Q x$ are like:
In general, $q_{jj}$ multiplies $x_j^2$. For any $j \neq k$, both $q_{jk}$ and $q_{kj}$ multiply $x_j x_k$, so nothing is lost in stipulating that $Q$ be a symmetric matrix — one that has all elements $q_{jk} = q_{kj}$. Then

$$
\frac{\partial c(x)}{\partial x_j} = \frac{1}{2} \frac{\partial}{\partial x_j} \left( q_{jj} x_j^2 + 2 \sum_{k \neq j} q_{jk} x_j x_k + c_j \right) = q_{jj} x_j + \sum_{k \neq j} q_{jk} x_k + c_j = q^j x + c_j
$$

where $q^j$ is the $j$th row of $Q$. Again the constraints are linear and $\partial a^i(x)/\partial x_j$ is $a^i_j$. The constraints are equalities but the variables are restricted to $x_j \geq 0$, so the rules give the necessary conditions as:

$$
a^i x^* = b_i, \quad \text{for each } i = 1, \ldots, m
$$
$$
\sum_{i=1}^m \pi^*_i a^i_j \leq q^j x^* + c_j, \quad \text{for each } j = 1, \ldots, n
$$
$$
x_j = 0 \text{ or } \sum_{i=1}^m \pi^*_i a^i_j = q^j x^* + c_j \text{ (or both)}, \quad \text{for each } j = 1, \ldots, n
$$

In the special case of all $q^j = 0$ these reduce to the complementary slackness conditions for linear programming. But in contrast to the LP case, here the "primal" variables $x_j^*$ appear in the "dual" constraints $\sum_{i=1}^m \pi^*_i a^i_j \leq q^j x^* + c_j$. A kind of simplex method can nevertheless be devised for certain quadratic programs as we will show later in this chapter. 

### 17.3 Complications introduced by nonlinearity

Our examples so far may suggest that the nonlinear case is a straightforward extension of the linear one, but in fact the nonlinearities introduce quite a few complications. This section illustrates the complications through a series of examples, most of them combining the constraints of Section 17.1 with different, quadratic objectives.

**Example 17–3.** We begin by observing that optimal solutions can occur anywhere in the feasible set, not only at extreme points. If we change our objective to, for example,

$$
\text{Minimize } c(x) = 9x_1^2 + 16x_2^2
$$
Figure 17–4. A local optimum of a nonlinear function at a point where only one constraint is active.

then, as seen in Figure 17–4, an objective contour just touches the feasible region at the local optimum \( x^* = (2, 1^{1/2}) \). But at this point there is only one active constraint:

\[
\begin{align*}
a^1(1, 3) &= 9 \cdot 2 - (1^{1/2})^2 > 0 \\
a^2(1, 3) &= 2 \cdot 1^{1/2} = 3 \\
a^3(1, 3) &= -3 \cdot \log 2 + 4 \cdot 1^{1/2} > 0
\end{align*}
\]

There is just one significant dual value, \( \pi^*_2 \), which must satisfy

\[
\begin{align*}
\pi^*_2 \nabla a^2(x^*) &= \nabla c(x^*) \\
\pi^*_2 \cdot (x^*_2, x^*_1) &= (18x^*_1, 32x^*_2) \\
\pi^*_2 \cdot (1^{1/2}, 2) &= (18 \cdot 2, 32 \cdot 1^{1/2}) \\
\pi^*_2 &= 24.
\end{align*}
\]

Here the "cone" of normals to the active constraints is just the half-line of points \( \pi^*_2 \cdot (1^{1/2}, 2) \), \( \pi^*_2 \geq 0 \), and \( \nabla c(x) = (36, 48) \) is in that cone because it equals \( 24 \cdot (1^{1/2}, 2) \).

A locally minimal solution may also occur in the interior of the feasible region, as happens for example when the objective is changed to

\[
\text{Minimize } c(x) = (x_1 - 3)^2 + (x_2 - 4)^2
\]

The point \( x^* = (3, 4) \) is clearly a minimum, since \( c(x^*) = 0 \) and cannot possibly take any negative value since it is a sum of squares. But no constraint is active at this point:
\[
\begin{align*}
a^1(1,3) &= 9 \cdot 3 - 4^2 > 0 \\
a^2(1,3) &= 3 \cdot 4 > 3 \\
a^3(1,3) &= -3 \cdot \log 3 + 4 \cdot 4 > 0
\end{align*}
\]

Since all constraints have \(a^i(x^*) > b_i\), the optimality conditions require that all \(\pi^*_i\) be zero. Thus the condition \(\sum_{i \in B} \pi^*_i \nabla a^i(x) = \nabla c(x)\) reduces to \(0 = \nabla c(x)\), which is indeed satisfied at \(x^* = (3,4)\):

\[
\nabla c(x^*) = \left(\frac{\partial c(x^*)}{\partial x_1}, \frac{\partial c(x^*)}{\partial x_2}\right) = (2 \cdot (x^*_1 - 3), 2 \cdot (x^*_2 - 4)) = (0,0).
\]

From a geometric perspective, the “cone” generated by the empty set of normals to the active constraints is the degenerate cone consisting of the one point \((0,0)\), and \(\nabla c(x^*)\) can be in this cone only if it equals \((0,0)\).

**Example 17-4.** Another significant difference, which we have touched upon briefly already, is that Property 17-1 gives only a necessary condition, which must be satisfied at any local minimum. Because it is not a sufficient condition, it may be satisfied at additional points that are not locally minimal. In the simplest situation, an interior point such as \(x^* = (3,4)\) may satisfy \(\nabla c(x^*) = 0\) even though it is locally maximal for \(c(x)\),

\[
\text{Minimize } c(x) = -(x_1 - 3)^2 - (x_2 - 4)^2
\]

or is neither minimal nor maximal:

\[
\text{Minimize } c(x) = (x_1 - 3)^2 - (x_2 - 4)^2
\]

In the latter case \(x^*\) is a “saddle point” of \(c(x)\) that is locally minimal in some directions and maximal in others.

Similar situations may occur at the boundary of the feasible region when the objective contours are tangent to the constraint contours. Figure 17-5 depicts the example in which the objective has been changed to:

\[
\text{Minimize } c(x) = -(x_1 - 11/4)^2 - (x_2 - 5/2)^2
\]

Here \(x^* = (2,11/2)\) is a feasible point at which only the second constraint is active:

\[
\begin{align*}
a^1(2,11/2) &= 9 \cdot 2 - (11/2)^2 > 0 \\
a^2(2,11/2) &= 2 \cdot 11/2 = 3 \\
a^3(2,11/2) &= -3 \cdot \log 2 + 4 \cdot 11/2 > 0
\end{align*}
\]

The constraint gradient \(\nabla a^2(x^*) = (11/2, 2)\) and the objective gradient \(\nabla c(x^*) = (11/2, 2)\), so the necessary conditions \(\pi^*_2 \nabla a^2(x^*) = \nabla c(x^*)\), \(\pi^* \geq 0\) are satisfied with \(\pi^*_2 = 1\). But Figure 17-5 shows that while the objective-function contour \(c(x) = c(x^*) = -19/16\) just touches the feasible region as we have come to expect at optimal points, \(c(x) = -19/16 - \epsilon\) also intersects the feasible region for any small enough \(\epsilon > 0\); the figure shows the contour for \(c(x) = -21/4\) as an example. The problem here is that, in contrast to the objective shown in Figure...
Figure 17–5. A point that satisfies the Karush-Kuhn-Tucker necessary conditions for a local optimum, but that is not locally optimal. The dashed circle is a contour for a smaller value of the objective, and does intersect the feasible region.

17–4, the objective here is curved in the same direction as the one active constraint, and in fact has a greater curvature than the constraint. But gradients alone do not capture any information about curvature. We’ll subsequently see that, except in special cases, sufficient conditions for optimality must involve the second derivatives of the objective and constraint functions.

A related phenomenon can occur at an extreme point of the feasible region in a “degenerate” situation where some of the $\pi^*_i$ values are zero. An example is given by changing the objective to

$$\text{Minimize } c(x) = -(x_1 - 11/2)^2 - 2(x_2 - 3/2)^2$$

At the extreme point $x^* = (1,3)$, the active set is $B = \{1, 2\}$, and the necessary conditions $\pi^*_1 \nabla a^1(x^*) + \pi^*_2 \nabla a^2(x^*) = \nabla c(x)$, $\pi^*_i \geq 0$ are satisfied by $\pi^*_1 = 1$ and $\pi^*_2 = 0$. But Figure 17–6 suggests that the contours for $c(x) = c(x^*) - \epsilon$ intersect the feasible region (“above” the extreme point) for any sufficiently small $\epsilon > 0$. Indeed, for all small enough values $\epsilon > 0$, the solution $(1 + 2/3\epsilon + 1/9\epsilon^2, 3 + \epsilon)$ satisfies the constraints and has a lower objective value $c(x^*) - 13/9\epsilon^2 - 4/27\epsilon^3 - 1/81\epsilon^4$. Hence $(1,3)$ is not a locally minimal solution for this objective.

Finally, the necessary conditions may not be sufficient at points where the active constraints’ gradients are not linearly independent. This has not been an issue in our examples where the optimum has been at $x^* = (1,3)$, because at that point the active constraints have linearly independent gradients $\nabla a^1(x^*) =$
Figure 17–6. A point that satisfies the Karush-Kuhn-Tucker necessary conditions for a local optimum, but that is not locally optimal. Here the objective normal at \((1, 3)\) is in the cone generated by the constraint normals, but the multiplier \(\pi_2^* = 0\).

\((9, -6)\) and \(\nabla a^2(x^*) = (3, 1)\). But consider the following problem depicted in Figure 17–7:

Minimize \(c'(x) = x_1\)

Subject to \(a^{1'}(x) = x_1 + x_2^2 \geq 4\)
\(a^{2'}(x) = x_1 - x_2 \geq 0\)
\(a^{3'}(x) = -4x_1 - x_2^2 \geq -16\)

At the feasible point \(x^* = (4, 0)\) there are two active constraints:

\(a^{1'}(4, 0) = 4 + 0^2 = 4 \quad \Rightarrow \quad 1 \in \mathcal{B}\)
\(a^{2'}(4, 0) = 4 - 0 > 0\)
\(a^{3'}(4, 0) = -4 \cdot 4 - 0^2 = -16 \quad \Rightarrow \quad 3 \in \mathcal{B}\)

Thus the necessary conditions for optimality are satisfied at \(x^*\) if there exist \(\pi^*_1\) and \(\pi^*_3\) such that

\[\pi^*_1 \nabla a^{1'}(x^*) + \pi^*_3 \nabla a^{3'}(x^*) = \nabla c'(x^*)\]

\[\Rightarrow \pi^*_1 \cdot (1, 2x^*_2) + \pi^*_3 \cdot (-4, -2x_2) = (1, 0)\]

\[\Rightarrow \pi^*_1 \cdot (1, 0) + \pi^*_3 \cdot (-4, 0) = (1, 0).\]

These conditions are in fact satisfied for any \(\pi^*_1 \geq 0, \pi^*_3 \geq 0\) such that \(\pi^*_1 - 4\pi^*_3 = 1\). But \((4, 0)\) is not locally minimal, because there are arbitrarily close points \((4 - \epsilon, \epsilon^2)\) that are also feasible in the constraints but that achieve a lower
Figure 17–7. A point that satisfies the Karush-Kuhn-Tucker necessary conditions for a local optimum, but that is not locally optimal. In this case, the difficulty stems from the linear dependence of the constraint normals at the point (4,0).

objective value $4 - \epsilon$. The requirement that the constraint gradients (or normals) be linearly independent is an example of what is called a constraint qualification.

Sufficient conditions for optimality in the general nonlinear case can be stated in various ways, but all involve requirements on the second derivatives of the objective and constraints as well as constraint qualifications. Beginning with the next chapter we will introduce some sufficient conditions, but starting with simpler cases in which they are easier to understand and to check.

One other complication introduced by nonlinearities has been the need to use the term locally optimal. In linear programming, a locally optimal solution is also globally optimal, in that it is minimal over all feasible solutions rather than only those in some neighborhood of the optimum. This is why, for example, the simplex method can test for global optimality by checking only the reduced costs at the current iterate. Locally optimal solutions are also guaranteed to be globally optimal for nonlinear problems that have certain properties of convexity, as we will observe in the next section. In general, however, a locally optimal solution need not be globally optimal.

Example 17–5. Using again our constraints from earlier in this chapter, the problem
Minimize $c(x) = -(x_1 - 3)^2 - (x_2 - 3)^2$
Subject to $a^1(x) = 9x_1 - x_2^2 \geq 0$
$a^2(x) = x_1x_2 \geq 3$
$a^3(x) = -3\log x_1 + 4x_2 \geq 0$

has solutions that satisfy the necessary conditions at two distinct extreme points:

- Where the active set $B = \{1, 2\}$, the solution $x^{*1} = (1, 3)$ satisfies
  $\pi_1^{*1}\nabla a^1(x^{*1}) + \pi_2^{*1}\nabla a^2(x^{*1}) = \nabla c(x^{*1})$, $\pi_1^{*1} \geq 0$, $\pi_2^{*1} \geq 0$
  with $\pi_1^{*1} = \frac{4}{27}$, $\pi_2^{*1} = \frac{8}{9}$.
- Where $B = \{2, 3\}$, the solution $x^{*2} = (3.32732, .901626)$ satisfies
  $\pi_2^{*2}\nabla a^2(x^{*2}) + \pi_3^{*2}\nabla a^3(x^{*2}) = \nabla c(x^{*2})$, $\pi_2^{*2} \geq 0$, $\pi_3^{*2} \geq 0$
  with $\pi_2^{*2} = .176390$, $\pi_3^{*2} = .902461$ (to 6 places).

Figure 17-8 illustrates this situation. Although the objective has different values at $x^{*1}$ and $x^{*2}$, its contours just touch the feasible region at both of these points, so that it is locally optimal at both. It has a lower value at $x^{*2}$, however, so that $x^{*1}$ cannot possibly be a global optimum.

Multiple locally optimal points are a common feature of nonlinear optimization problems of practical interest, especially those arising from physical problems. Sometimes it is possible to determine that a particular locally optimal solution is globally optimal — or is at least one of the better locally optimal solutions — by appealing to special properties of the problem at hand. In the
next several chapters we continue to focus on methods for local nonlinear optimization, however. When we subsequently turn to global optimization, which is a much harder problem, we will see that some approaches employ a series of local optimizations.

17.4 Convexity in optimization

There is one very important special case of nonlinear optimization in which local optimality implies global and the necessary conditions are also sufficient. This is where the objective and the feasible region are both “convex” in an appropriate sense.

Fundamentally, a function is convex if a line between any two points on its graph lies entirely on or above the graph, as in Figure 17–9, never falling below the graph as in Figure 17–10. Formally, this condition — and its opposite, where the line lies entirely on or below the graph — can be stated as follows:

**Definition 17–2** A function $f(x)$ is **convex** on a set $S$ if, for any $x_1 \in S$ and $x_2 \in S$ and for any $0 < \lambda < 1$,

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2).$$

**Definition 17–3** A function $f(x)$ is **concave** on a set $S$ if, for any $x_1 \in S$ and $x_2 \in S$ and for any $0 < \lambda < 1$,

$$f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda)f(x_2).$$

In follows immediately from the definitions that $f(x)$ is convex if and only if $-f(x)$ is concave, and $f(x)$ is concave if and only if $-f(x)$ is convex. A linear function $f(x)$ satisfies by definition $f(\lambda x_1 + (1-\lambda)x_2) = \lambda f(x_1) + (1-\lambda)f(x_2)$, so it is both convex and concave. There are many more convexity results for specific functions: in one dimension, for example, $x^2$ is convex; $\log x$ is concave on $(0, \infty)$; $1/x$ is convex on $(0, \infty)$ and concave on $(-\infty, 0)$. Simple convex functions can be combined into more complex ones by various rules, the most important of which are that the sum of convex functions is convex, and that a nonnegative multiple of a convex function is convex.

The strong convex equivalent of Property 17–1 has the following statement:

**Property 17–2** If $c(x)$ is convex and $a^i(x)$ is concave for every $i = 1, \ldots, m$, then $x^*$ is globally optimal for

Minimize $c(x)$

Subject to $a^i(x) \geq b_i$, $i = i, \ldots, m$

if and only if there exists a $\pi^*$ such that:

- $a^i(x^*) \geq b_i$, for each $i = 1, \ldots, m$
- $\sum_{i=1}^m \pi_i^* \nabla a^i(x^*) = \nabla c(x^*)$, $\pi^* \geq 0$
- Either $a^i(x^*) = b_i$ or $\pi_i^* = 0$ (or both), for each $i = 1, \ldots, m$
Taking negatives, this result is seen to apply as well to maximizing a concave function and to constraints $a^i(x) \leq b_i$ where $a^i(x)$ is convex. Since a linear $a^i(x)$ is both convex and concave, the constraints may contain either $a^i(x) \leq b_i$ or $a^i(x) \geq b_i$, or (if both hold) $a^i(x) = b_i$. Nonnegativity restrictions on the variables are merely a special case of linear constraints. Thus linear programming is a special case of Property 17–2; the necessary and sufficient conditions for a global optimum in this case reduce to the standard complementary slackness conditions.

**Example 17–1 (revisited).** Since $\log x_j$ is concave, $-\log x_j$ is convex and so is $-\mu \log x_j$. The function $cx$ is linear and hence also convex. Thus the objective of the barrier problem, $cx - \mu \sum_{j=1}^{n} \log x_j$ is a sum of convex functions, which is convex. The constraint functions are all linear, a special case of concave.

The barrier problem thus satisfies Property 17–2. It follows that, when an interior-point method takes a step toward the solution to the necessary conditions for this problem, it is stepping toward the problem’s global optimum. This property is one essential element of the method’s fast and reliable convergence.

**Example 17–2 (revisited).** If $x^TQx$ is a convex function of $x$, then so is $\frac{1}{2}x^TQx + cx$, and, since the constraints $Ax = b$ and $x \geq 0$ are linear, the quadratic program satisfies Property 17–2. A global solution is guaranteed by methods, such as we will outline in the next two sections, that find solutions to the necessary conditions.

By the definition of convexity, $x^TQx$ is convex if and only if it satisfies
Figure 17–10. A function that is not convex.

\[(\lambda x_1 + (1 - \lambda)x_2)^T Q(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda x_1^T Qx_1 + (1 - \lambda)x_2^T Qx_2.\]

Following some straightforward algebraic manipulation, this inequality becomes

\[\lambda(1 - \lambda)(x_1 - x_2)^T Q(x_1 - x_2) \geq 0.\]

Thus we can make the following assertion: If \(v^T Qv \geq 0\) for all vectors \(v\), then the objective of the quadratic program is convex and any solution to the necessary optimality conditions is a global minimum.

A symmetric matrix \(Q\) that has \(v^T Qv \geq 0\) for all \(v\) is said to be positive semi-definite. There are very efficient computational methods for testing the positive semi-definiteness of a matrix, so that a method for quadratic programming can test in advance whether the objective it has been passed is convex.

Our observations here have a nice generalization. Any function \(f(x)\) is convex on a set \(S\) if its symmetric matrix of second partial derivatives \(\frac{\partial^2 f(x)}{\partial x^2}\) is positive semi-definite at all \(x \in S\). In the quadratic case this matrix is \(Q\) everywhere, but in general its entries are dependent on \(x\). On account of this dependence, there is in general no way to efficiently prove convexity of a function in an optimization problem; at best we can prove convexity for particular functions by cleverly applying various properties of convexity and concavity such as those we have mentioned in this section.

\[\blacksquare\]

17.5 Generalization of interior-point methods

We have shown how interior-point methods for linear programming can be developed as methods for solving the complementary slackness conditions for
optimality. Analogous methods for nonlinear programming can be developed by applying an analogous approach to the generalization of the complementary slackness conditions embodied in Property 17–1. A complete derivation of such a method requires tools not needed in the linear case, however, which we will introduce only in subsequent chapters. Here we only outline the nonlinear interior-point method for the purpose of contrasting it to the linear method.

Recall that for a linear program in equality form with nonnegative variables,

\[
\begin{align*}
\text{Minimize} & \quad c^T x \\ 
\text{Subject to} & \quad Ax = b \\ & \quad x \geq 0
\end{align*}
\]

each step of the simplest interior-point method attempted to work toward solving the conditions

\[
\begin{align*}
Ax &= b \\
A^T \pi + \sigma &= c \\
X\Sigma e &= 0 \\
&\quad x \geq 0, \quad \sigma \geq 0
\end{align*}
\]

Beginning from some \( \bar{x} > 0, \bar{\pi}, \) and \( \bar{\sigma} > 0, \) each iteration solved the equations

\[
\begin{bmatrix}
-X^{-1}\Sigma & A^T \\
A & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \pi
\end{bmatrix} =
\begin{bmatrix}
c - A^T \bar{\pi} \\
b - A\bar{x}
\end{bmatrix},
\Delta \sigma = -\bar{\sigma} - X^{-1}\Sigma \Delta x,
\]

and then tried to step as far as possible along the line toward

\((\bar{x} + \Delta x, \bar{\pi} + \Delta \pi, \bar{\sigma} + \Delta \sigma)\)

while keeping all elements of \( x \) and \( \sigma \) at least some small amount above zero.

Suppose now that we have a nonlinear program in a similar equality form,

\[
\begin{align*}
\text{Minimize} & \quad c(x) \\ 
\text{Subject to} & \quad a^i(x) = b_i, \quad i = i, \ldots, m \\ & \quad x_j \geq 0, \quad j = i, \ldots, n
\end{align*}
\]

Then by the rules set forth in Section 17.2, the necessary conditions for optimality are

\[
\begin{align*}
&\forall \quad a^i(x) = b_i, \quad i = i, \ldots, m \quad \text{and} \quad x_j \geq 0, \quad j = 1, \ldots, n \\
&\forall \quad \sum_{i=1}^m \pi \partial a^i(x)/\partial x_j \leq \partial c(x)/\partial x_j, \quad j = 1, \ldots, n \\
&\forall \quad \text{Either} \quad x_j = 0 \quad \text{or} \quad \sum_{i=1}^m \pi \partial a^i(x)/\partial x_j = \partial c(x)/\partial x_j \quad \text{(or both),} \quad j = i, \ldots, n
\end{align*}
\]

When slack variables \( \sigma \) are added, these conditions can be expressed in terms of \( m + 2n \) equations in \( m + 2n \) variables:

\[
\begin{align*}
a^i(x) &= b_i, \quad i = 1, \ldots, m \\
\sum_{i=1}^m \pi \partial a^i(x)/\partial x_j + \sigma_i &= \partial c(x)/\partial x_j, \quad j = 1, \ldots, n \\
x_j \sigma_j &= 0, \quad j = 1, \ldots, n
\end{align*}
\]
together with \( x_j \geq 0 \) and \( \sigma_j \geq 0 \) for all \( j = 1, \ldots, n \). Equivalently, writing \( A(x) \) for \((a^1(x), \ldots, a^m(x))\) and \( \nabla A(x) \) for the matrix whose \((i, j)\) entry is \( \partial a^i(x)/\partial x_j \), the optimization problem is

\[
\text{Minimize } \ c(x) \\
\text{Subject to } \ A(x) = b \\
x \geq 0
\]

and the optimality conditions have the form:

\[
A(x) = b \\
\nabla A(x)^T \pi + \sigma = \nabla c(x) \\
X \Sigma e = 0 \\
x \geq 0, \sigma \geq 0
\]

In this terminology, the analogy between the linear and nonlinear cases is easy to see.

If we were to proceed with the development of the interior-point method as in the linear case, we would replace \((x, \pi, \sigma)\) by \((\bar{x} + \Delta x, \bar{\pi} + \Delta \pi, \bar{\sigma} + \Delta \sigma)\) in the above conditions, then throw away all high-order terms and solve the resulting linear equations for the step direction \((\Delta x, \Delta \pi, \Delta \sigma)\). Identifying and discarding “high-order terms” involves the approximation of the objective and constraint functions by linear functions, which in this more general setting gives rise to what is known as Newton’s method. We will introduce Newton’s method for single-variable functions in the next chapter, and will consider its use to solve nonlinear equations in many variables in Chapter 3.

Thus here we only state the results of applying Newton’s method to the above optimality conditions. The equations for the step direction involve the first derivatives \( \nabla c(x) \) and \( \nabla A(x) \) and also corresponding second derivatives \( \nabla^2 c(x) \) and \( \nabla^2 a^i(x) \), whose \((i, j)\) entries are \( \partial^2 c(x)/\partial x_i \partial x_j \) and \( \partial^2 a^i(x)/\partial x_i \partial x_j \), respectively. If we also define

\[
\nabla^2 \Pi(\bar{x}) = \nabla^2 c(\bar{x}) - \sum_{i=1}^m \nabla^2 a^i(\bar{x}) \bar{\pi}_i,
\]

then the equations are

\[
\begin{bmatrix}
-\bar{X}^{-1} \Sigma - \nabla^2 \Pi(\bar{x}) \\
\nabla A(\bar{x})^T
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \pi
\end{bmatrix}
= 
\begin{bmatrix}
\nabla c(\bar{x}) - \nabla A(\bar{x})^T \bar{\pi} \\
b - A(\bar{x})\bar{\pi}
\end{bmatrix},
\]

\[
\Delta \sigma = -\bar{\sigma} - \bar{X}^{-1} \Sigma \Delta x.
\]

In the linear case these are exactly the equations we had before, with \( \nabla^2 \Pi(\bar{x}) \) vanishing because the second derivatives of linear functions are all zero. The second-derivative term significantly complicates the solving of these equations, however. Simply forming \( \nabla^2 \Pi(\bar{x}) \) can be expensive; in the most general setting, where the nonlinear functions have no special forms that can be exploited, this term is a sum of \( m + 1 \) symmetric \( m \times m \) matrices whose entries are functions of \( \bar{x} \).

 Whereas \( -\bar{X}^{-1} \Sigma \) is a diagonal matrix, moreover, \( -\bar{X}^{-1} \Sigma - \nabla^2 \Pi(\bar{x}) \) need have no simple structure. Implementations for linear programming can eliminate
\[ \Delta x \] to produce a positive semi-definite system of the form \( A(\bar{X}^{-1})A^T\Delta \pi = \ldots \) that is amenable to many computational efficiencies. This approach does not extend to the nonlinear case, however, because in general the inverse of 
\[ -\bar{X}^{-1}\Sigma - \nabla^2 \Pi(\hat{x}) \] is not efficiently computable. Implementations for nonlinear programming must instead address the whole system in \((\Delta x, \Delta \pi)\). Besides being larger, this system is much harder; while it is symmetric, it is typically not semidefinite.

Given a step direction, the computation of a step length is also not so easy as in the linear case. The objective does not vary linearly with the distance moved from \((\hat{x}, \hat{\pi}, \hat{\sigma})\) to \((\hat{x} + \Delta x, \hat{\pi} + \Delta \pi, \hat{\sigma} + \Delta \sigma)\), and consequently there is no simple formula for the step length. Instead the step length has to be chosen by some kind of efficient one-dimensional minimization algorithm. Such algorithms have been studied extensively in the context of nonlinear optimization, and we will describe them in some detail in Chapter 15.2.

**Example ?? (revisited).** When \( c(x) \) is the quadratic objective function \( x^T Q x + c x \) and \( a^i(x) \) is the linear function \( a^i x \), the second derivatives \( \nabla^2 c(\hat{x}) = Q \) and \( \nabla^2 a^i(\hat{x}) = 0 \) at any \( \hat{x} \). Together with the first derivatives that we already know, this gives the simplified step equations

\[
\begin{bmatrix}
-\bar{X}^{-1}\Sigma - Q & A^T \\
A & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta \pi
\end{bmatrix}
= 
\begin{bmatrix}
Q \hat{x} + c - A^T \hat{\pi} \\
b - A \hat{x}
\end{bmatrix}.
\]

There are correspondingly simpler procedures for the choice of the step length.

If the objective is only a sum of squares — \( c(x) = \sum_{i=1}^m q_{ii} x^2_i \) — then \( Q \) is diagonal. The upper left-hand block is then a diagonal matrix as in the linear case. If also all \( q_{ii} \geq 0 \) — so that the objective is convex — then the system that results upon eliminating \( \Delta x \) has coefficient matrix \( A(\bar{X}^{-1}\Sigma + Q)^{-1}A^T \), which is positive semidefinite. Thus all the efficiencies of computing the step directions for linear programs carry over to convex sum-of-squares minimization.

### 17.6 Generalization of simplex methods

Simplex methods generalize to reduced gradient methods, which will eventually be described here.
18. Unconstrained One-Variable Nonlinear Optimization with Derivatives

Even the unconstrained minimization or maximization of a nonlinear function of a single variable can be a nontrivial problem. We begin in this chapter by considering the univariate case, where a function $f$ of a single variable $x$ is to be minimized.

**Definition 18–1** $x^*$ is a *local minimum* [local maximum] of $f$ if there is some $\delta > 0$ so that $f(x^*) \leq [\geq] f(x^* + \Delta x)$ for all $-\delta \leq \Delta x \leq \delta$.

**Definition 18–2** $x^*$ is a *global minimum* [global maximum] of $f$ if $f(x^*) \leq [\geq] f(x^* + \Delta x)$ for any value of $\Delta x$.

This chapter and the next are concerned with methods that can be guaranteed to find only local minimums and maximums. Global optimization is a harder problem that must be attacked with substantially different methods. Fortunately, for practical purposes a local optimization method often suffices. Particular properties of the function of interest often can be used to show that a local optimum is in fact global; an example is given by certain “convex” cases to be considered later.

18.1 Analytical optimization

For functions whose derivatives can be computed — what are called smooth functions — unconstrained optimization is a standard topic in the study of elementary calculus. In the univariate case, to find a potential minimum or maximum, you set the first derivative to zero and solve for the variable. If the second derivative at a solution is positive or negative then you have a local minimum or maximum, respectively.

Derivatives are such an important concept that there are several independent notations for them. Perhaps the most common in calculus texts uses $df(x)/dx$ for the first derivative of $f(x)$ with respect to $x$, and $d^2f(x)/dx^2$ for the second derivative. We adopt the following alternative, however, which is more common in studies of optimization — perhaps because it is more convenient when derivatives are part of larger expressions:

$f'(x) =$ first derivative of $f$ with respect to $x$

$f''(x) =$ second derivative of $f$ with respect to $x$

Third and higher derivatives can be denoted by adding more ’ characters after $f$ or by writing $f^{(k)}$ for the $k$th derivative. We say that a function is differentiable at a value $\bar{x}$ if $f^{(k)}$ can be computed for any $k \geq 1$.

The relationships between derivatives and optimization derive from the so-called Taylor series for smooth functions. If $f$ is a univariate function differentiable at $\bar{x}$,
\[
f(\hat{x} + \Delta x) = \sum_{k=0}^{\infty} \frac{1}{k!} f^{(k)}(\hat{x}) \Delta x^k
\]

\[
= f(\hat{x}) + f'(\hat{x}) \Delta x + \frac{1}{2} f''(\hat{x}) \Delta x^2 + \frac{1}{6} f'''(\hat{x}) \Delta x^3 + \frac{1}{24} f''''(\hat{x}) \Delta x^4 + \cdots
\]

for all \( \Delta x \) of sufficiently small magnitude. We do not work directly with this infinite sum, but rather rely on the ability of finite numbers of terms to provide good approximations to \( f \) in the vicinity of \( \hat{x} \):

**Property 18–1 (Taylor approximation)** For any univariate function \( f \) differentiable at \( \hat{x} \), the first \( t \) terms of the Taylor series approximate \( f \) to within a multiple of \( \Delta x^t \) for all sufficiently small \( \Delta x \). Specifically, there exist a multiple \( \beta_t > 0 \) and a radius \( \delta_t > 0 \), depending in general on \( f \) and \( \hat{x} \), such that

\[
|f(\hat{x} + \Delta x) - \sum_{k=0}^{t-1} \frac{1}{k!} f^{(k)}(\hat{x}) \Delta x^k| \leq \beta_t |\Delta x|^t
\]

for all \( |\Delta x| \leq \delta_t \). \( \square \)

Of particular interest are the linear approximation corresponding to \( t = 2 \),

\[
|f(\hat{x} + \Delta x) - f(\hat{x}) - f'(\hat{x}) \Delta x| \leq \beta_2 |\Delta x|^2
\]

for all \( |\Delta x| \leq \delta_2 \), and the quadratic approximation given by \( t = 3 \),

\[
|f(\hat{x} + \Delta x) - f(\hat{x}) - f'(\hat{x}) \Delta x - \frac{1}{2} f''(\hat{x}) \Delta x^2| \leq \beta_3 |\Delta x|^3
\]

for all \( |\Delta x| \leq \delta_3 \), where the \( \beta_t \) and \( \delta_t \) values depending on \( f \) and \( \hat{x} \).

A necessary condition for optimality is one that must be true at any optimum. Conditions involving both first and second derivatives are a consequence of the Taylor series approximations:

**Property 18–2 (First-order necessary condition)** If \( x^* \) is a local minimum or local maximum of \( f \) then \( f'(x^*) = 0 \).

\textbf{Proof} for the local minimum case: Substituting \( x^* \) and \( x - x^* \) for \( \hat{x} \) and \( \Delta x \) in the linear Taylor series approximation, we can deduce that

\[
f(x) \leq f(x^*) + f'(x^*)(x - x^*) + \beta_2(x - x^*)^2
\]

for all \( x \) such that \( |x - x^*| \leq \delta_2 \), where \( \beta_2 \) and \( \delta_2 \) depend only on \( f \) and \( x^* \).

If \( f'(x^*) < 0 \), we observe that the last two terms of the approximation satisfy

\[
f'(x^*)(x - x^*) + \beta_2(x - x^*)^2 < 0
\]

for \( x^* < x < x^* - f'(x^*)/\beta_2 \), a non-empty interval. Hence \( f(x) < f(x^*) \) for all sufficiently small \( x > x^* \), so \( x^* \) cannot be a local minimum.

Similarly, if \( f'(x^*) > 0 \), we observe that the last two terms of the approximation satisfy the same inequality for \( x^* > x > x^* - f'(x^*)/\beta_2 \), also a non-empty interval. Hence again \( f(x) < f(x^*) \) for all sufficiently small \( x < x^* \), so that \( x^* \) cannot be a local minimum in this case either.

Thus when \( x^* \) is a local minimum the only possibility is that \( f'(x^*) = 0 \). \( \square \)
Property 18–3 (Second-order necessary condition) If \( x^* \) is a local minimum [local maximum] of \( f \) then \( f''(x^*) \geq [\leq] 0 \).

**Proof** for the local minimum case: Substituting \( x^* \) and \( x - x^* \) for \( \bar{x} \) and \( \Delta x \) in the quadratic Taylor series approximation, and setting \( f'(x^*) = 0 \) as required by the first-order optimality condition, we can deduce that

\[
f(x) \leq f(x^*) + \frac{1}{2} f''(x^*)(x - x^*)^2 + \beta_3 |x - x^*|^3
\]

for all \( x \) such that \( |x - x^*| \leq \delta_3 \), where \( \beta_3 \) and \( \delta_3 \) depend only on \( f \) and \( x^* \).

If \( f''(x^*) < 0 \), we observe that the last two terms of the approximation satisfy

\[
\frac{1}{2} f''(x^*)(x - x^*)^2 + \beta_3 |x - x^*|^3 < 0
\]

for all \( x \neq x^* \) such that \( |x - x^*| < \frac{1}{2} f'(x^*)/\beta_3 \). Hence \( f(x) < f(x^*) \) for all \( x \) sufficiently close to \( x^* \), so that \( x^* \) cannot be a local minimum.

It follows that \( x^* \) can only be a local minimum when \( f''(x^*) \geq 0 \). □

A sufficient condition is one that implies optimality. The Taylor series gives a condition of this kind as well:

Property 18–4 (Second-order sufficient condition) If \( f'(x^*) = 0 \) and if \( f''(x^*) > [<] 0 \) then \( x^* \) is a local minimum [maximum] of \( f \).

**Proof** for the local minimum case: Substituting \( x^* \) and \( x - x^* \) for \( \bar{x} \) and \( \Delta x \) in the quadratic Taylor series approximation, we can deduce that

\[
f(x) \geq f(x^*) + f'(x^*)(x - x^*) + \frac{1}{2} f''(x^*)(x - x^*)^2 - \beta_3 |x - x^*|^3
\]

for all \( x \) such that \( |x - x^*| \leq \delta_3 \), where \( \beta_3 \) and \( \delta_3 \) depend only on \( f \) and \( x^* \).

The hypothesis \( f'(x^*) = 0 \) implies

\[
f'(x^*)(x - x^*) = 0
\]

for any \( x \). Moreover, given \( f''(x^*) > 0 \), the last two terms

\[
\frac{1}{2} f''(x^*)(x - x^*)^2 - \beta_3 |x - x^*|^3 \geq 0
\]

for all \( x \) such that \( |x - x^*| \leq \frac{1}{2} f'(x^*)/\beta_3 \). Thus \( f(x) \geq f(x^*) \) for all \( x \) sufficiently close to \( x^* \), implying that \( x^* \) is a local minimum of \( f \). □

These conditions motivate searching for locally optimal points by solving \( f'(x) = 0 \), which is a necessary condition for either a local minimum or a local maximum. If a solution \( x^* \) satisfies \( f''(x^*) > 0 \) or \( f''(x^*) < 0 \), then local minimality or maximality of \( x^* \) is confirmed. In the case where also \( f''(x^*) = 0 \), however, nothing definite can be said without looking at higher-order derivatives (see Example 18–4).

**Example 18–1.** Here are two optimization problems defined on a box that has a square base:
Optimization Methods — §18.1

Maximize the volume, given that the surface area must be 10 cm$^2$.

Minimize the surface area, given that the volume must be 10 cm$^3$.

Writing $x$ for the side of the base and $y$ for the height, we can express these problems mathematically as follows:

Maximize $x^2y$  
Minimize $2x^2 + 4xy$

Subject to $2x^2 + 4xy = 10$  
$x \geq 0$, $y \geq 0$

Subject to $x^2y = 10$  
$x \geq 0$, $y \geq 0$

We can eliminate the equality constraints by solving for $y = (10 - 2x^2)/4x$ and $y = 10/x^2$, respectively, and substituting these expressions for $y$ into the objectives. Then we have:

Maximize $v(x) = 5/2x - 1/2x^3$  
Minimize $s(x) = 2x^2 + 40/x$

We have set aside $x \geq 0$ and $y \geq 0$ so as to apply the calculus-based “set the derivative to zero” approach to the resulting unconstrained problems in one variable. So long as we are only interested in positive solutions, we can ignore any that come out less than or equal to zero.

We proceed to find potentially optimal solutions by setting the derivatives of $v(x)$ and $s(x)$ to zero and solving for $x$. We call the resulting solutions $x_v$ and $x_s$:

$v'(x) = (5/2) - (3/2)x^2 = 0 \implies x_v = \pm \sqrt{5/3}$

$s'(x) = 4x - 40/x^2 = 0 \implies x_s = \sqrt[3]{10}$

In the first problem we ignore the negative solution. Substituting back into the expressions for $y$, we have corresponding values of $y$:

$y_v = (10 - 2x^2)/4x_v = (10 - 2(5/3))/4\sqrt{5/3} = \sqrt{5/3}$

$y_s = 10/x_s^2 = 10/((\sqrt[3]{10})^2) = \sqrt[3]{10}$

So we see that in both cases the optimum is achieved by a box that is a cube. Checking the second derivatives, we have

$v''(x_v) = -3x_v = -3\sqrt{5/3} < 0$

$s''(x_s) = 80/x^3 = 8 > 0$

So $x_v$ is a maximum and $x_s$ is a minimum, as desired.

**Example 18-2.** For $f(x) = x^6 - 3x^5 + 2x^4 - 7x^3 - 6x^2 + x + 1$ (Figure 18-1), the necessary condition is

$f'(x) = 6x^5 - 15x^4 + 8x^3 - 21x^2 - 12x + 1 = 0$.

There are at most 5 roots of this polynomial, hence at most 5 local minima or maxima. But there exists no general formula for 5th-order roots. For this case there is no way to get around using an iterative computational method such as those we are about to describe.
Figure 18–1. Plot of $f(x) = x^6 - 3x^5 + 2x^4 - 7x^3 - 6x^2 + x + 1$.

Example 18–3. The function $f(x) = \sum_{i=1}^{n} |x - a_i|$ (Figure 18–2) is continuous, but derivatives cannot be taken where $x = a_i$. Hence the necessary and sufficient conditions for smooth functions can’t be applied to this case.

Example 18–4. For $f(x) = x^3$ (Figure 18–3), we have $f'(x) = 3x^2$ and $f''(x) = 6x$. Thus $f'(0) = 0$, but also $f''(0) = 0$, and so although $x^* = 0$ satisfies the necessary conditions for optimality, it does not satisfy the 2nd-order sufficient conditions. In fact 0 is neither a minimum nor a maximum for $f$.

For $f(x) = x^4$, 0 is clearly a local minimum (indeed a global minimum, since $f(x)$ is positive for any other value of $x$). But $f'(0) = f''(0) = f'''(0) = 0$ in this case, while the local minimum is implied by $f''''(0) = 24 > 0$. For $f(x) = -x^4$, the situation is the same, except that $f''''(0) = -24 < 0$ implies that 0 is a local maximum.

Figure 18–2. Plot of $f(x) = \sum_{i=1}^{n} |x - a_i|$ for $n = 2$, $a_1 = 0.25$, $a_2 = 2.25$. 
18.2 Rates and orders of convergence

We will take an iterative approach to minimizing or maximizing \( f(x) \), or equivalently to solving the necessary conditions \( f'(x) = 0 \). We start with some guess at the solution, \( x^{(0)} \), and generate subsequent iterates \( x^{(1)}, x^{(2)}, x^{(3)}, \ldots \) according to a certain formula. When the formula is properly chosen, the sequence of \( x^{(k)} \) values converges to the desired solution \( x^* \) in the sense that, if we iterate long enough, \( x^{(k)} \) will come as close to \( x^* \) as we like.

As a practical matter, the limited precision of numbers stored in the computer places a limit on how closely an iterative method can come to computing a solution. Typically the method is stopped when the progress from one iteration to the next is sufficiently small relative to the magnitude of the current iterate; that is, when

\[
|x^{(k+1)} - x^{(k)}| \leq \epsilon |x^{(k)}|
\]

for some small value \( \epsilon \). Alternatively, the method can stop when \( f'(x) \) is sufficiently close to zero:

\[
|f'(x)| \leq \epsilon.
\]

For the precision of arithmetic on most current computers, a value of \( \epsilon \) around \( 10^{-12} \) is appropriate.

**Example 18-5.** A fast and elementary iterative algorithm for finding square roots is based on the observation that

\[
x = \sqrt{a} \iff x^2 = a \iff x = a/x
\]

If our initial guess \( x^{(0)} = a/x^{(0)} \), then we have guessed the square root correctly. Otherwise we take the average of \( x^{(0)} \) and \( a/x^{(0)} \) and make that our new guess, \( x^{(1)} \). Then again if \( x^{(1)} = a/x^{(1)} \) we have the correct square root, or else we take the average of \( x^{(1)} \) and \( a/x^{(1)} \) as our next guess, \( x^{(2)} \). The general formula for the iterations is
\[ x^{(k+1)} = \frac{1}{2}(x^{(k)} + a/x^{(k)}), \quad k = 0, 1, 2, \ldots \]

Since we are solving the equation \( x^2 - a = 0 \), we can take \( x^2 - a \leq \epsilon \) as our stopping criterion.

Applying this algorithm to the case of \( a = 5/3 \) from Example 18–1, with a starting guess of 1.5 and tolerance \( \epsilon = 1e-12 \), we see that it converges quite rapidly:

<table>
<thead>
<tr>
<th>( k )</th>
<th>( x(a/x) )</th>
<th>( x^2 - a )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.50000000000000</td>
<td>1.11111111111111</td>
</tr>
<tr>
<td>1</td>
<td>1.30555555555556</td>
<td>1.27659574468085</td>
</tr>
<tr>
<td>2</td>
<td>1.2910765011820</td>
<td>1.29091325246052</td>
</tr>
<tr>
<td>3</td>
<td>1.2909445128936</td>
<td>1.290944618225</td>
</tr>
<tr>
<td>4</td>
<td>1.2909444873581</td>
<td>1.2909444873581</td>
</tr>
</tbody>
</table>

The computed square root is \( x^* = 1.2909444873581 \). The first iterate \( x^{(1)} \) matches this value to about 2 digits; \( x^{(2)} \) matches it to about 4 digits, \( x^{(3)} \) to about 8 digits. So the accuracy is doubling every step. The exponents of \( |x^2 - a| \) show this doubling as well.  

**Example 18–6.** We think of applying this same idea to computing the cube root, observing that

\[ x = \sqrt[3]{a} \iff x^3 = a \iff x = a/x^2 \]

The iterations can be the same as before, except that \( x \) and \( a/x^2 \) are averaged:

\[ x^{(k+1)} = \frac{1}{2}(x^{(k)} + a/[x^{(k)}]^2), \quad k = 0, 1, 2, \ldots \]

Applying this algorithm to the case of \( a = 10 \) from Example 18–1, with a starting guess of 3 and the same tolerance \( \epsilon = 1e-12 \), we observe a far slower convergence than in the square-root case:

| \( k \) | \( x \) | \( a/x^2 \) | \( |x^3 - a| \) |
|--------|----------------|----------------|----------------|
| 0      | 2.00000000000000 | 2.50000000000000 | 2.000e+00 |
| 1      | 2.25000000000000 | 1.97530864197531 | 1.391e+00 |
| 2      | 2.1126532098765 | 2.240905448358 | 5.706e-01 |
| 3      | 2.17657243291173 | 2.11083239777431 | 3.114e-01 |
| 4      | 2.14370241534302 | 2.17606069923053 | 1.487e-01 |
| 5      | 2.1598155728677 | 2.14358212920111 | 7.604e-02 |
| 6      | 2.15173184324394 | 2.15985057322262 | 3.759e-02 |
| 7      | 2.15579120823328 | 2.15724213833388 | 1.890e-02 |
| 8      | 2.15375771103358 | 2.15578928646880 | 9.424e-03 |
| 9      | 2.15477349875119 | 2.15375723240403 | 4.719e-03 |
| 10     | 2.15426536557761 | 2.15477337868799 | 2.358e-03 |
| 11     | 2.15451937222280 | 2.15426533563508 | 1.179e-03 |
| 12     | 2.1543923592894 | 2.15451936473364 | 5.895e-04 |
| 13     | 2.15445585933129 | 2.15439235205709 | 2.948e-04 |
| 14     | 2.15442410569419 | 2.15445585886327 | 1.474e-04 |
| 15     | 2.15443998227873 | 2.15442410557719 | 7.369e-05 |
The computed cube root is $2.15443469003184$. About 3 steps on average are needed to get $x$ to match this value in one more decimal place, no matter how many places are already matched, and the values of $|x^3 - a|$ fall by only about half at each step.

There are more precise terms than “fast” and “slow” to characterize the speed of convergence. First we make precise what it means for a sequence of values to converge:

**Definition 18–3** A series of values $x^{(k)}$, $k = 0, 1, 2, \ldots$ converges to a value $x^*$ if, given any tolerance $\epsilon > 0$, there exists a $k_\epsilon$ such that $|x^{(k)} - x^*| < \epsilon$ for all $k \geq k_\epsilon$.

This captures the informal idea that the convergent sequence $x^{(k)}$ comes arbitrarily close to $x^*$ for all large enough values of $k$. In mathematical notation the fact that $x^{(k)}$ converges to $x^*$ is written \( \lim_{k \to \infty} x^{(k)} = x^* \), or

\[ x^{(k)} \to x^* \]

where it is clear from the context that $x^{(k)}$ represents a sequence of values $x^{(0)}$, $x^{(1)}$, $x^{(2)}$, \ldots.

We can now define the concepts of convergence that distinguish Example ?? from Example ??:
Definition 18–4 The order of convergence of a series $x^{(k)} \to x^*$ is the largest number $q$ such that for all $p < q$,

$$
\lim_{k \to \infty} \frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|^p} < \infty.
$$

A series that converges with order at least 2 has quadratic convergence. A series such that

$$
\lim_{k \to \infty} \frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|} = 0
$$

has superlinear convergence. A series such that

$$
\lim_{k \to \infty} \frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|} = r, \text{ where } 0 < r < 1,
$$

has linear convergence, and $r$ is its ratio of convergence.

Superlinear implies convergence of at least order 1, and linear implies convergence of exactly order 1. It is possible to have convergence for which the ratio $r$ in the definition is equal to 1, in which case convergence is order 1 but not linear — very slow indeed. For $r > 1$ the series must diverge, however, since then $|x^{(k+1)} - x^*|$ becomes larger than $|x^{(k)} - x^*|$ at every step as $k \to \infty$.

The results for Example ?? suggest that the convergence there is quadratic. The quantity $|x^{(k+1)} - x^*|/|x^{(k)} - x^*|^p$ stays at a finite value for $p = 2.00$, but steadily increases for, say, $p = 2.25$:

\begin{tabular}{ccc}
  $k$ & $x^{(k)}$ & $p = 2.00$ & $p = 2.25$
  \hline
  0 & 1.50000000000000 & 3.333e-01 & 4.930e-01
  1 & 1.30555555555556 & 3.830e-01 & 1.102E+00
  2 & 1.29107565011820 & 3.873e-01 & 4.080E+00
  \hline
\end{tabular}

In fact this is a special case of a more general quadratically convergent method, as we will see in the next section.

The results for Example ?? show linear convergence with ratio 1/2:

\begin{tabular}{ccc}
  $k$ & $x^{(k)}$ & $|x^{(k+1)} - x^*|/|x^{(k)} - x^*|$
  \hline
  0 & 2.00000000000000 & 6.188E-01
  1 & 2.25000000000000 & 4.372E-01
  2 & 2.11265432098765 & 5.299E-01
  3 & 2.17657243291173 & 4.848E-01
  4 & 2.14370241534302 & 5.075E-01
  5 & 2.15988155728677 & 4.962E-01
  6 & 2.15173184324394 & 5.019E-01
  7 & 2.15579120823328 & 4.991E-01
  8 & 2.15375771103358 & 5.005E-01
  9 & 2.15477349875119 & 4.998E-01
  10 & 2.15426536557761 & 5.001E-01
  \hline
\end{tabular}
The linear convergence is a characteristic and elementary property of most algorithms of this kind, as we will show in the next section; we will also see that the ratio of convergence depends only on the function and on the solution to which the iterates converge. Section 18.4 will then explain the approach that must be taken to achieve quadratic convergence.

A helpful way to distinguish different orders and rates of convergence is to think in terms of the number of zeroes after the decimal point in \( |x^{(k)} - x^*| \). Roughly speaking, this is the number of correct digits in \( x^{(k)} \) relative to \( x^* \). If it equals \( d \), then \( |x^{(k)} - x^*| \) must be less than \( 10^{-d} \) but greater than \( 10^{-(d+1)} \), and so will have an exponent of \(- (d + 1)\) in scientific notation. For instance, here are the numbers of correct digits in the \( \sqrt{10} \) iterates from Example ??:

| \( k \) | \( x^{(k)} \) | \( |x^{(k)} - x^*| \) | correct digits |
|---|---|---|---|
| 0 | 1.50000000000000 | 0.20900555126419 | 2.090E-01 | 0 |
| 1 | 1.30555555555556 | 0.01456110681975 | 1.456E-02 | 1 |
| 2 | 1.29107565011820 | 0.00008120138240 | 8.120E-05 | 4 |
| 3 | 1.29099445128936 | 0.00000002553555 | 2.554E-09 | 8 |
| 4 | 1.29099444873581 | | | |

The exponent can be approximated by the logarithm of the number to the base 10, so we can say that

\[
\text{number of correct digits in } x^{(k)} \approx - \log_{10} |x^{(k)} - x^*| - 1.
\]

Now applying the log to the ratios of Definition 18–4, we have

\[
- \log_{10} \frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|}^p
= (- \log_{10} |x^{(k+1)} - x^*| - 1) - p(- \log_{10} |x^{(k)} - x^*| - 1)
\approx (\text{correct digits in } x^{(k+1)}) - p \cdot (\text{correct digits in } x^{(k)})
\]

So when \( \lim_{k \to \infty} |x^{(k+1)} - x^*|/|x^{(k)} - x^*|^p = r < \infty \), the result in terms of correct digits is

\[
(\text{correct digits in } x^{(k+1)}) \approx p \cdot (\text{correct digits in } x^{(k)}) - \log_{10} r \quad \text{as } k \to \infty.
\]

This can be interpreted as telling us how fast the number of correct digits increased from one iterate to another.

For quadratic convergence \((p = 2)\), eventually the \( \log_{10} r \) term becomes small relative to the others, and the number of correct digits roughly doubles at each iteration. This is what we observed in Example ?? by the same reasoning, if convergence is order \( p \) then the number of correct digits should eventually increase by roughly a factor of \( p \) at each iteration.

For superlinear convergence \((p = 1, r \to 0)\), the gain in correct digits tends to get larger at each step, since \( - \log_{10} r \to \infty \) as \( r \to 0 \).

For linear convergence \((p = 1, 0 < r < 1)\), the number of correct digits tends to increase by \( - \log_{10} r \) at each step. This is what we observed in Example ??, where \( r = \frac{1}{2} \). Each step increases the number of correct digits by \( - \log_{10} \frac{1}{2} = \).
so that the number of steps needed to get one more correct digit is a little more than 3. No matter how many correct digits have been found so far, the work to determine one more is then same; this is the undesirable property that distinguishes linear from superlinear convergence.

In the next section we show that naive algorithms are generally linear, and in the following section that properly constructed algorithms are superlinear and indeed quadratic if second derivatives are employed. These results apply only to the univariate case, but the following chapter will show that similar observations hold for the multivariate case as well.

### 18.3 Convergence of iterative algorithms

We can think of each iterative algorithm of the sort we’ve been considering as being defined by a function \( A \). Given an initial guess \( x^{(0)} \), the algorithm generates iterates

\[
x^{(k+1)} = A(x^{(k)}), \quad k = 0, 1, 2, \ldots
\]

By evaluating the derivative of \( A \) at the point of convergence \( x^* \), we can determine whether the convergence is linear or superlinear, and the ratio of convergence for the former case:

**Property 18–5 (Linear convergence)** If an algorithm \( x^{(k+1)} = A(x^{(k)}) \) generates \( x^{(k)} \to x^* \) with \( A \) continuous at \( x^* \), then

\[
x^* = A(x^*) \quad \text{and} \quad 0 \leq |A'(x^*)| \leq 1.
\]

If \( 0 < |A'(x^*)| < 1 \) then the convergence to \( x^* \) is linear with ratio \( |A'(x^*)| \), and if \( |A'(x^*)| = 0 \) then the convergence is superlinear.

**Proof:** Continuity of \( A \) implies that \( \lim_{k \to \infty} A(x^{(k)}) = A(\lim_{k \to \infty} x^{(k)}) \). Combining this with the definition of the algorithm, we have

\[
x^* = \lim_{k \to \infty} x^{(k+1)} = \lim_{k \to \infty} A(x^{(k)}) = A(\lim_{k \to \infty} x^{(k)}) = A(x^*).
\]

Continuity of \( A \) then also implies that the formula for the ratio of convergence is an expression for the derivative of \( A \) at \( x^* \):

\[
\lim_{k \to \infty} \frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|} = \lim_{k \to \infty} \frac{|A(x^{(k)}) - A(x^*)|}{|x^{(k)} - x^*|} = A'(x^*).
\]

It follows that \( 0 \leq |A'(x^*)| \leq 1 \), since as previously remarked any higher value of the ratio-of-convergence formula would not allow the iterates to converge. The rest of the property follows directly by substituting \( |A'(x^*)| \) for \( \lim_{k \to \infty} |x^{(k+1)} - x^*|/|x^{(k)} - x^*| \) in Definition 18–4. \(

In the case of our cube-root algorithm (Example ??), we have

\[
A(x) = \frac{1}{2}(x + a/x^2) \quad \text{and} \quad A'(x) = \frac{1}{2}(1 - 2a/x^3).
\]

\( A(x) \) is continuous at any nonzero \( x \), and by Property 18–5 the algorithm defined by \( A \) can only converge with

\[
x^* = A(x^*) = \frac{1}{2}(x^* + a/(x^*)^2) \Rightarrow x^* = \sqrt[3]{a}.
\]
Thus $A$ is continuous at $x^*$ for any nonzero $a$, and
\[ |A'(x^*)| = |A'(\frac{3}{2}a)| = |\frac{1}{2}(2a/\sqrt{3}a^3)| = \frac{1}{2}, \]
confirming linear convergence with the ratio that we observed. In fact this shows that the algorithm can only achieve linear convergence with ratio $\frac{1}{2}$ for any choice of $a \neq 0$.

In the case of our square-root algorithm (Example ??), by contrast,
\[ A(x) = \frac{1}{2}(x + a/x), \quad A'(x) = \frac{1}{2}(1 - a/x^2). \]
Again $A(x)$ is continuous at any nonzero $x$, and the algorithm defined by this $A$ must converge to
\[ x^* = A(x^*) = \frac{1}{2}(x^* + a/x^*) \implies x^* = \pm \sqrt{a}. \]
So again $A$ is continuous at $x^*$ for any nonzero $a$. But now we observe that
\[ |A'(x^*)| = |A'(\sqrt{10})| = |\frac{1}{2}(1 - 10/(\sqrt{10})^2)| = 0. \]
Hence Property 18–5 says convergence must be superlinear for any $a \neq 0$, and in fact it turned out to be quadratic in our example.

We are thus motivated to turn next to the derivation of an approach that can yield a similarly fast algorithm for smooth functions in general.

### 18.4 Newton’s method

The key to high-order convergence of iterative methods for smooth nonlinear optimization is approximation. Though in general we have no formula for a local optimum of $f(x)$, we can construct a simpler function $f^{(k)}(x)$ that approximates $f(x)$ well in the vicinity of the current iterate $x^{(k)}$ and that is easy to optimize. The minimum of this approximating function gives us the next iterate:
\[ x^{(k+1)} = \min_x f^{(k)}(x). \]
A similar method works for solving a nonlinear equation, and indeed is the same method when applied to solving the first-order optimality condition $f'(x) = 0$.

The desired approximation is provided by the Taylor series of Property 18–1. For $x$ in the vicinity of $x^{(k)}$,
\[
\begin{align*}
    f(x^{(k)} + \Delta x) &= f(x^{(k)}) + f'(x^{(k)})\Delta x + \frac{1}{2}f''(x^{(k)})\Delta x^2 \\
    &\quad + \frac{1}{6}f'''(x^{(k)})\Delta x^3 + \frac{1}{24}f''''(x^{(k)})\Delta x^4 + \cdots
\end{align*}
\]
As we have already observed in motivating the necessary and sufficient optimality conditions in Section 18.1, the second-order term dominates the sum of all the remaining terms for all $\Delta x$ close enough to zero. Hence it makes sense to use as our approximation the function that remains when all higher-order terms are dropped:
\[
    f^{(k)}(x^{(k)} + \Delta x) = f(x^{(k)}) + f'(x^{(k)})\Delta x + \frac{1}{2}f''(x^{(k)})\Delta x^2
\]
For any \( x^{(k)} \) such that \( f''(x^{(k)}) \neq 0 \), this is a quadratic function in \( \Delta x \). It has a unique optimum where its derivative with respect to \( \Delta x \) is zero:

\[
\frac{df^{(k)}(x^{(k)} + \Delta x)}{d(\Delta x)} = f'(x^{(k)}) + f''(x^{(k)}) \Delta x = 0 \quad \Rightarrow \quad \Delta x = -\frac{f'(x^{(k)})}{f''(x^{(k)})}.
\]

The resulting algorithm is defined by setting \( x^{(k+1)} \) to \( x^{(k)} + \Delta x \), that is,

\[
x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}.
\]

This is known as Newton’s method, after its inventor who is better known for his contributions to physics but who introduced many fundamentals of calculus in the process. (How much of calculus he independently invented is subject to some dispute.)

**Example 18–1 (revisited).** To minimize \( s(x) = 2x^2 + 40/x \) iteratively, Newton’s method computes

\[
x^{(k+1)} = x^{(k)} - \frac{s'(x^{(k)})}{s''(x^{(k)})} = x^{(k)} - \frac{4x^{(k)} - 40/[x^{(k)}]^2}{4 + 80/[x^{(k)}]^3} = \frac{30x^{(k)}}{[x^{(k)}]^3 + 20}.
\]

From an initial guess of \( x^{(0)} = 1.5 \) it quickly converges to a point where the first-order necessary condition \( s'(x) = 0 \) is satisfied within a very small tolerance:

<table>
<thead>
<tr>
<th>k</th>
<th>( x^{(k)} )</th>
<th>( s'(x^{(k)}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.50000000000000</td>
<td>-1.178E+01</td>
</tr>
<tr>
<td>1</td>
<td>1.92513689839574</td>
<td>-3.092E+00</td>
</tr>
<tr>
<td>2</td>
<td>2.12841001068192</td>
<td>-3.161E-01</td>
</tr>
<tr>
<td>3</td>
<td>2.15411780671898</td>
<td>-3.803E-03</td>
</tr>
<tr>
<td>4</td>
<td>2.15443464341878</td>
<td>-5.594E-07</td>
</tr>
<tr>
<td>5</td>
<td>2.15443469003188</td>
<td>-8.882E-15</td>
</tr>
</tbody>
</table>

Quadratic convergence to \( x^* = 2.15443469003188 \) is again seen in the tendency toward doubling of the correct digits at each step. Since \( s''(x^*) = 4 + 80/[x^*]^3 = 12.000 \) is positive, \( x^* \) is indeed a local minimum.

**Example 18–2 (revisited).** To find locally optimal points of \( f(x) = x^6 - 3x^5 + 2x^4 - 7x^3 - 6x^2 + x + 1 \), Newton’s method iterates

\[
x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})} = x^{(k)} - \frac{6x^5 - 15x^4 + 8x^3 - 21x^2 - 12x + 1}{30x^4 - 60x^3 + 24x^2 - 42x - 12}.
\]

Here there are several locally optimal points for \( f \), and the method converges to different ones depending on the starting point:

<table>
<thead>
<tr>
<th>k</th>
<th>( x^{(k)} )</th>
<th>( f'(x^{(k)}) )</th>
<th>( f''(x^{(k)}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.00000000000000</td>
<td>2.350E+02</td>
<td>8.880E+02</td>
</tr>
<tr>
<td>1</td>
<td>2.73536036036036</td>
<td>5.384E+01</td>
<td>5.042E+02</td>
</tr>
<tr>
<td>2</td>
<td>2.62857826674412</td>
<td>6.485E+00</td>
<td>3.859E+02</td>
</tr>
<tr>
<td>3</td>
<td>2.61177415225887</td>
<td>1.427E-01</td>
<td>3.690E+02</td>
</tr>
<tr>
<td>4</td>
<td>2.61138729612799</td>
<td>7.432E-05</td>
<td>3.686E+02</td>
</tr>
<tr>
<td>5</td>
<td>2.61138709450272</td>
<td>2.014E-11</td>
<td>3.686E+02</td>
</tr>
<tr>
<td>6</td>
<td>2.61138709450266</td>
<td>1.101E-13</td>
<td>3.686E+02</td>
</tr>
</tbody>
</table>
The sign of the second derivative shows that the first and third solutions are local minima while the second is a local maximum.

Newton's method converges to a particular local minimum or maximum when started from any sufficiently close \( x^{(0)} \). From other starting points its behavior can be hard to predict. Here are a few examples:

<table>
<thead>
<tr>
<th>( k )</th>
<th>( x^{(k)} )</th>
<th>( f'(x^{(k)}) )</th>
<th>( f''(x^{(k)}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.000000000000000</td>
<td>-9.100E+01</td>
<td>0.000E+00</td>
</tr>
<tr>
<td>1</td>
<td>1.999900000000000</td>
<td>-9.100E+01</td>
<td>-2.940E-02</td>
</tr>
<tr>
<td>1</td>
<td>-3093.64245508552000</td>
<td>-1.702E+18</td>
<td>2.750E+15</td>
</tr>
<tr>
<td>2</td>
<td>-2474.8139418283000</td>
<td>-5.576E+17</td>
<td>1.126E+15</td>
</tr>
<tr>
<td>3</td>
<td>-1979.75123297364000</td>
<td>converges to -0.4839 . . . after 42 iterations</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2.000100000000000</td>
<td>-9.100E+01</td>
<td>2.940E-02</td>
</tr>
<tr>
<td>1</td>
<td>3096.83390208620000</td>
<td>1.708E+18</td>
<td>2.757E+15</td>
</tr>
<tr>
<td>2</td>
<td>2477.56715266503000</td>
<td>5.596E+17</td>
<td>1.129E+15</td>
</tr>
<tr>
<td>3</td>
<td>1982.15375988928000</td>
<td>converges to 2.4691 . . . after 39 iterations</td>
<td></td>
</tr>
</tbody>
</table>

In general, whenever \( f''(x^{(k)}) \approx 0 \) but \( f'(x^{(k)}) \neq 0 \), the step \( f'(x^{(k)})/f''(x^{(k)}) \) will be very large and the future behavior of the method will have little to do with \( x^{(k)} \).

Example 18–7. An example of nonconvergence is provided by \( f(x) = \ln x + 1/x^2 \) (Figure ??). The formula for the iterations is

\[
x^{(k+1)} = x^{(k)} - f'(x^{(k)})/f''(x^{(k)}) = x^{(k)} - \frac{1/x^{(k)} - 2/[x^{(k)}]^3}{-1/[x^{(k)}]^2 + 6/[x^{(k)}]^4}.
\]

When the starting point is close to the minimum, the iterates converge quickly:
Figure 18-4. Plot of $f(x) = \ln x + 1/x^2$.

<table>
<thead>
<tr>
<th>k</th>
<th>$x^{(k)}$</th>
<th>$f'(x^{(k)})$</th>
<th>$f''(x^{(k)})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.50000000000000</td>
<td>7.407E-02</td>
<td>7.407E-01</td>
</tr>
<tr>
<td>1</td>
<td>1.40000000000000</td>
<td>-1.458E-02</td>
<td>1.052E+00</td>
</tr>
<tr>
<td>2</td>
<td>1.41386138613861</td>
<td>-3.524E-04</td>
<td>1.001E+00</td>
</tr>
<tr>
<td>3</td>
<td>1.41421334319673</td>
<td>-2.192E-07</td>
<td>1.000E+00</td>
</tr>
<tr>
<td>4</td>
<td>1.41421356237301</td>
<td>-8.493E-14</td>
<td>1.000E+00</td>
</tr>
</tbody>
</table>

But when the starting point is too large, the iterates instead fly off to infinity:

<table>
<thead>
<tr>
<th>k</th>
<th>$x^{(k)}$</th>
<th>$f'(x^{(k)})$</th>
<th>$f''(x^{(k)})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.00000000000000</td>
<td>2.593E-01</td>
<td>-3.704E-02</td>
</tr>
<tr>
<td>1</td>
<td>10.00000000000000</td>
<td>9.800E-02</td>
<td>-9.400E-03</td>
</tr>
<tr>
<td>2</td>
<td>20.42553191489360</td>
<td>4.872E-02</td>
<td>-2.362E-03</td>
</tr>
<tr>
<td>3</td>
<td>41.04975463703470</td>
<td>2.433E-02</td>
<td>-5.913E-04</td>
</tr>
<tr>
<td>4</td>
<td>82.19730020015760</td>
<td>1.216E-02</td>
<td>-1.479E-04</td>
</tr>
<tr>
<td>5</td>
<td>164.44330705293200</td>
<td>6.081E-03</td>
<td>-3.697E-05</td>
</tr>
<tr>
<td>6</td>
<td>328.91094399675500</td>
<td>3.040E-03</td>
<td>-9.243E-06</td>
</tr>
<tr>
<td>7</td>
<td>657.83405001466800</td>
<td>1.520E-03</td>
<td>-2.311E-06</td>
</tr>
</tbody>
</table>

You can see from Figure ?? that as $x \to \infty$ the slopes are decreasing, as if $f(x)$ might be approaching a maximum. But in fact $f'(x) > 0$ for any finite $x$; intuitively, the iterates "converge" to the "maximum at infinity" rather than to some finite value.

18.5 Newton’s method for solving equations

The formula for Newton’s method makes no reference to $f(x^{(k)})$. In fact what it finds are solutions to $f'(x) = 0$, the first-order necessary conditions for a local minimum or maximum. It works just as well for computing zeroes of any nonlinear function, and can readily be derived as a zero-finding method.

Specifically, a smooth function $g$ can be approximated near an iterate $x^{(k)}$ by a function $g^{(k)}$ consisting of the first two terms of the Taylor series for $g$: 
Figure 18–5. A step of Newton’s method for solving $g(x) = 0$.

$$g^{(k)}(x^{(k)} + \Delta x) = g(x^{(k)}) + g'(x^{(k)})\Delta x.$$  

Provided $g'(x^{(k)}) \neq 0$, this approximation is a linear function that is zero at a unique value of $\Delta x$:

$$g(x^{(k)}) + g'(x^{(k)})\Delta x = 0 \implies \Delta x^{(k)} = -\frac{g(x^{(k)})}{g'(x^{(k)})}.$$  

As before the resulting algorithm is defined by setting $x^{(k+1)}$ to $x^{(k)} + \Delta x^{(k)}$, or

$$x^{(k+1)} = x^{(k)} - \frac{g(x^{(k)})}{g'(x^{(k)})},$$

Figure 18.4 shows graphically how Newton’s method works in this case. The graph of $g^{(k)}$ is the line tangent to the graph of $g$ at $x^{(k)}$. The point where this line intersects the horizontal axis is the solution to $g^{(k)}(x) = 0$, and becomes $x^{(k+1)}$. Near a solution to $g(x) = 0$ the tangent lines are excellent approximations to $g$, and intuitively this is the reason why convergence is so fast once the iterates get near a solution.

Example 18–5 (revisited). To compute the square root of $a$ we can apply Newton’s method to solving $g(x) = x^2 - a = 0$. The derivative is $g'(x) = 2x$, so the formula for the iterates is

$$x^{(k+1)} = x^{(k)} - \frac{[x^{(k)}]^2 - a}{2x^{(k)}} = \frac{1}{2} (x^{(k)} + a/x^{(k)}),$$

which is exactly the formula for the intuitive square root algorithm that we earlier observed to have quadratic convergence.

Example 18–6 (revisited). To compute the cube root of $a$ we can apply Newton’s method to solving $g(x) = x^3 - a = 0$. The derivative is $g'(x) = 3x^2$, so the formula for the iterates is

$$x^{(k+1)} = x^{(k)} - \frac{[x^{(k)}]^3 - a}{3[x^{(k)}]^2} = \frac{2}{3} x^{(k)} + \frac{1}{3} (a/[x^{(k)}]^2),$$

This is much like the formula of $\frac{1}{2} x^{(k)} + \frac{1}{2} (a/[x^{(k)}]^2)$ that was our intuitive extension of the square root method. But when we use it to compute $\sqrt[3]{10}$ from a starting point of 2.0 as before, convergence to within a tolerance of $1e-12$ is achieved in 4 iterations rather than 42:
A seemingly small difference between the two formulas makes for a huge difference in the speed of convergence.

18.6 Quadratic convergence of Newton’s method

The fast convergence of Newton’s method derives, like the method itself, from the Taylor series approximation. We first state and prove the simplest version, for solving equation systems.

Property 18–6 (Quadratic convergence of Newton’s method) Given a smooth function \( g \) such that 
\[
|g'(x_1) - g'(x_2)| < \gamma |x_1 - x_2|
\]
for some \( \gamma \) and any \( x_1, x_2 \), if \( g(x^*) = 0 \), \( g'(x^*) \neq 0 \), and some iterate \( x^{(k)} \) lies within a sufficiently small interval around \( x^* \), then Newton’s method converges to \( x^* \) with order two.

Proof: For a smooth function \( g \) the condition \( g'(x^*) \neq 0 \) implies
\[
\frac{1}{|g'(x)|} \leq \beta_1
\]
for all \( x \) such that \( |x - x^*| < \delta_1 \). Also the condition that \( |g'(x_1) - g'(x_2)| < \gamma |x_1 - x_2| \) enables us to state the following more general version of the linear Taylor approximation:
\[
|g(x) - g(x^*) - g'(x)(x - x^*)| \leq \beta_2(x - x^*)^2
\]
for all \( x \) such that \( |x - x^*| < \delta_2 \).

Multiplying the left sides and the right sides of these inequalities and putting \( g(x^*) = 0 \), we have
\[
\frac{1}{|g'(x)|} |g(x) - g'(x)(x - x^*)| = \left| x - \frac{g(x)}{g'(x)} - x^* \right| \leq \beta_1 \beta_2(x - x^*)^2
\]
for all \( x \) such that \( |x - x^*| < \delta = \min(\delta_1, \delta_2) \). Taking \( x^{(k)} \) for \( x \) and substituting the Newton iteration \( x^{(k+1)} = x^{(k)} - g(x^{(k)})/g'(x^{(k)}) \), we have
\[
|x^{(k+1)} - x^*| \leq \beta_1 \beta_2(x^{(k)} - x^*)^2.
\]

If also any iterate satisfies \( |x^{(k)} - x^*| < \theta/\beta_1 \beta_2 \) for some \( \theta < 1 \), then
\[
|x^{(k+1)} - x^*| \leq \theta|x^{(k)} - x^*|,
\]
so all subsequent iterates also satisfy $|x - x^*| < \delta$. It follows that $|x^{(k+1)} - x^*|/|x^{(k)} - x^*|^2$ is bounded by $\beta_1\beta_2$ for all subsequent $k$, and convergence with order 2 is established.

We can expect that $|g'(x_1) - g'(x_2)| < \gamma(x_1 - x_2)$ will hold in practical situations, since it says only that there is some bound on how fast the derivative of $g$ can change.

The requirement that $g'(x^*) \neq 0$ is not always satisfied, however, and where it fails the convergence may be only linear. If Newton's method for the square root, $x^{(k+1)} = \frac{1}{2}(x^{(k)} + a/x^{(k)})$, is applied at $a = 0$, for example, it becomes $x^{(k+1)} = \frac{1}{2}x^{(k)}$, which clearly has only linear convergence to $x^* = 0$ with ratio $\frac{1}{2}$. The problem here is that $g(x) = x^2 - a$, so $g'(x^*) = 2x^* = 0$.

The analogous property of Newton's algorithm for minimizing $f(x)$ is the same, but with $f'$ and $f''$ replacing $g$ and $g'$. Quadratic convergence can be achieved at points where $f''(x^*) = 0$, provided $f'''(x^*) \neq 0$. These points may or may not be locally minimal or maximal.