\( x^* \in \mathbb{R}^n \) satisfies

1. \( \nabla f(x^*) = 0; \)
2. \( \nabla^2 f(x^*) \) is positive definite (and hence, in particular, nonsingular);
3. \( \nabla^2 f \) is Lipschitz continuous on a neighborhood of \( x^* \).

Then \( x^* \) is a strict local minimizer of \( f \) and, for any \( H_0 \) sufficiently close to \( \nabla^2 f(x^*) \) and any \( x^{(0)} \) sufficiently close to \( x^* \), Newton’s method defines a sequence that converges superlinearly to \( x^* \).

As mentioned above, it is usual to take \( H_0 \) as a multiple of the identity, so it is not clear that the hypothesis of the above theorem (that \( H_0 \) is sufficiently close to \( \nabla^2 f(x^*) \)) would hold in practice. The tolerance for “sufficiently close” is typically unknown in practice, so the final arbiter must be computational experience, which has shown that the BFGS method is both effective and efficient. Although the convergence is not as fast as the quadratic convergence of Newton’s method, in many cases secant methods such as BFGS may still be more efficient (namely, when it is expensive to compute \( \nabla^2 f(x^{(k)}) \)).

### 3.3 Line search algorithms and acceptable steps

I have now explained two quasi-Newton algorithms. The first is based on a direct modification of the Hessian, and the second uses the BFGS update. Both yield Hessian approximations \( H_k \) that are always positive definite and therefore always produce descent directions. At the \( k \)th iteration, the descent direction for \( f \) at \( x^{(k)} \) is \( p^{(k)} = -H_k^{-1} \nabla f(x^{(k)}) \). I will now explain how to use this descent direction.

The obvious algorithm is simply to use the descent direction itself as the step (the quasi-Newton step):

\[
x^{(k+1)} = x^{(k)} + p^{(k)}.
\]

This is in fact an excellent strategy near the solution \( x^* \), since both Newton’s method and the variants described above converge superlinearly once \( x^{(k)} \) is sufficiently close to \( x^* \). However, the quasi-Newton step may be a bad step when \( x^{(k)} \) is not near the solution. The reason is simple: Though a positive definite Hessian approximation \( H_k \) guarantees that \( p^{(k)} \) is a descent direction, there is no guarantee (unless \( x^{(k)} \) is close to \( x^* \)) that the length of \( p^{(k)} \) is any good.

**Example 3.5.** Let \( f : \mathbb{R} \to \mathbb{R} \) be defined by

\[
f(x) = \frac{1}{2} x^2 + 2x + e^{-16x^2} + 2e^{-5x^2}.
\]

The function \( f \), together with the best local quadratic approximation \( Q \) near \( x = 1 \), is graphed in Figure 3.2. As the graph shows, the step from \( x = 1 \) to the minimizer of the quadratic approximation (that is, the Newton step) is much longer than the step from \( x = 1 \) to the minimizer \( x^* \) of \( f \). The problem is that \( f \) is very nonquadratic between \( x = 1 \) and \( x = x^* \).
The remedy for the situation illustrated by the previous example is to define

\[ x^{(k+1)} = x^{(k)} - \alpha_k H^{-1}_k \nabla f(x^{(k)}), \]

where \( \alpha_k \) is chosen by minimizing (at least approximately) the function

\[ \alpha \mapsto f(x^{(k)} + \alpha p^{(k)}), \quad p^{(k)} = -H^{-1}_k \nabla f(x^{(k)}), \]

on the interval \( 0 < \alpha < \infty \). An algorithm for solving such a (one-dimensional) minimization problem is referred to as a line search. The variable \( \alpha \) is called the step length parameter.

### 3.3.1 Acceptable steps

In the quasi-Newton framework, it is known that step length one is suitable near the solution; indeed, to attain rapid local convergence, it is important to choose \( \alpha_k = 1 \) when \( x^{(k)} \) is near \( x^* \). On the other hand, since \( p^{(k)} \) is always a descent direction, choosing \( \alpha \) small enough must lead to a reduction in \( f \) (and hence to progress in minimizing \( f \)) even when step length one is a bad choice. For these reasons, the most popular line search algorithms are of the backtracking type. A backtracking line search first tries \( \alpha = 1 \) and then, if this is unacceptable, reduces \( \alpha \) until an acceptable step length is found. Before I can describe backtracking algorithms in detail, I need to define what constitutes an acceptable step.

#### The Wolfe conditions

Ideally, having determined a descent direction \( p^{(k)} \) at \( x^{(k)} \), \( \alpha_k \) would be chosen to be the (global) minimizer of the function

\[ \phi(\alpha) = f(x^{(k)} + \alpha p^{(k)}), \quad \alpha \geq 0. \]
In this way, the algorithm would reduce \( f \) as much as possible in going from \( x^{(k)} \) to \( x^{(k+1)} \). However, it is usually expensive to accurately compute a minimizer of \( \phi \) (and, in fact, usually impossible to find the global minimizer of \( \phi \) given the available information). Computational experience has shown conclusively that computing even a local minimizer is not worth the expense. It is more efficient to settle for a step that can be computed inexpensively, provided it gives “sufficient” decrease in \( f \).

There are two ways that a line search can continually reduce \( f \) without reducing it enough to obtain convergence. The first way is illustrated in Figure 3.3. In this case, the iterations repeatedly go from one side of a valley to the other, always reducing \( f \) but never by much. The problem is that the reduction in each step is very little compared to the length of the steps—the steps are too long.

![Graph](image-url)

**Figure 3.3.** A sequence of steps that reduces a function \( f \) at every step and yet does not converge to a minimizer of \( f \). The problem with this sequence is that each step achieves very little reduction in \( f \) relative to the length of the step.

To avoid the problem illustrated by Figure 3.3, the following condition can be required of the step length \( \alpha \):

\[
f(x^{(k)} + \alpha_k p^{(k)}) \leq f(x^{(k)}) + c_1 \alpha_k \nabla f(x^{(k)}) \cdot p^{(k)}. \tag{3.18}
\]

The quantity \( \alpha_k \nabla f(x^{(k)}) \cdot p^{(k)} \) is the decrease in \( f \) predicted by the slope of \( f \) at \( x^{(k)} \) in the direction of \( p^{(k)} \) (the reader should notice that \( \alpha_k \nabla f(x^{(k)}) \cdot p^{(k)} < 0 \) since \( p^{(k)} \) is a descent direction). Of course, since the graph of \( \phi \) can curve upward as \( \alpha \) increases from 0, \( f \) may not attain this decrease for any \( \alpha_k > 0 \). However, it is easy to show that, for any \( c_1 \in (0,1) \), \( \alpha_k \) satisfies (3.18) for all \( c_1 \) sufficiently small. On the other hand, (3.18) prevents the line search from passing over a minimum of \( \phi \) and climbing too far up the other side of the “valley.” The meaning of (3.18) is illustrated in Figure 3.4.

The second way that an algorithm can reduce \( f \) without reducing it sufficiently is to take steps that are too short. As a simple example of this, suppose \( f : \mathbb{R} \rightarrow \mathbb{R} \) is defined by \( f(x) = x^2 \). Beginning from \( x^{(0)} = 2 \), \( p^{(k)} = -1 \) is always a descent
direction at \( x^{(k)} = 1 + 2^k \) and \( \alpha_k = 2^{-k-1} \), which yields \( x^{(k+1)} = 1 + 2^{-(k+1)} \), results in a decrease in \( f \). However, \( x^{(k)} \to 1 \), while the minimizer is \( x^* = 0 \). This example, simple though it is, shows that \( x^{(k)} \) can be headed in the correct direction and \( f(x^{(k)}) \) can be decreasing, and yet the solution may never be approached if the steps are too small.

Moreover, the sequence in the previous paragraph satisfies condition (3.18) assuming \( c_1 \) is chosen sufficiently small (for example, \( c_1 = 0.5 \) works, as can be easily shown). Thus condition (3.18) does not guard against overly short steps. The following condition prevents the step length from being too short:

\[
\nabla f(x^{(k)} + \alpha_k p^{(k)}) \cdot p^{(k)} \geq c_2 \nabla f(x^{(k)}) \cdot p^{(k)}. \tag{3.19}
\]

The parameter \( c_2 \) must lie in \((0,1)\), like \( c_1 \), and, as I will discuss below, it must be greater than \( c_1 \) in order to ensure that it is possible to satisfy both (3.18) and (3.19) simultaneously. Since \( \nabla f(x^{(k)} + \alpha_k p^{(k)}) \cdot p^{(k)} \) and \( \nabla f(x^{(k)}) \cdot p^{(k)} \) are the derivatives of \( \phi \) at \( \alpha = \alpha_k \) and \( \alpha = 0 \), respectively, (3.19) simply guarantees that \( \alpha_k \) is large enough that the slope of \( \phi \) has increased by some fixed relative amount. This prevents the line search from taking steps that become too small. Condition (3.19) is satisfied if \( \phi'(\alpha_k) \) is very small or positive, which means that \( \alpha_k \) close to or beyond a local minimizer of \( \phi \) would be satisfactory. This is another way to see that (3.19) prevents short steps.

Condition (3.19) can be written in the equivalent form

\[
(\nabla f(x^{(k+1)}) - \nabla f(x^{(k)})) \cdot (x^{(k+1)} - x^{(k)}) \geq (c_2 - 1) \nabla f(x^{(k)}) \cdot p^{(k)} > 0,
\]

which implies that \( y^{(k)} \cdot s^{(k)} > 0 \) (using the familiar notation \( s^{(k)} = x^{(k+1)} - x^{(k)} \),

Figure 3.4. Condition (3.18) illustrated. The solid curve is \( \phi \), the dashed line is the tangent line to \( \phi \) at \( \alpha = 0 \), and the dotted line is determined by (3.18) with \( c_1 = 0.5 \). In order for step length \( \alpha \) to satisfy (3.18), it has to lie in the interval on which the graph of \( \phi \) is below the graph of the dotted line. The reader should notice how (3.18) prevents the line search from taking overly long steps.
\[ y^{(k)} = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}) \]. Therefore, (3.19) implies that the BFGS update will be well-defined, a side benefit of this condition on the line search.

Together (3.18) and (3.19) are referred to as the Wolfe conditions or sometimes the Armijo-Goldstein conditions. The first condition is also called the sufficient decrease condition and the second the curvature condition. In place of (3.20), the following stronger condition is sometimes used:

\[
\left| \nabla f(x^{(k)} + \alpha_k p^{(k)}) \cdot p^{(k)} \right| \leq c_2 \left| \nabla f(x^{(k)}) \cdot p^{(k)} \right|. \tag{3.20}
\]

Using (3.20), the line search is actually seeking a minimizer, and a small value of \( c_2 \) implies an accurate minimization of \( \phi \). However, as mentioned above, it is not computationally efficient to perform an accurate minimization during the line search, so the weaker condition (3.19) is usually preferred in place of (3.20).

In fact, in the context of a backtracking line search, it is not even necessary to enforce (3.19) in order to avoid overly short steps. The backtracking strategy ensures that a sufficiently long step will be taken whenever possible. However, in the context of the BFGS method, (3.19) is necessary to ensure that the Hessian update is well-defined.

There are now several theoretical results to prove:

1. It is always possible to choose a step length \( \alpha_k \) such that both (3.18) and (3.19) are satisfied.

2. Under certain conditions on the Hessian approximations \( H_k \), the quasi-Newton algorithm with a line search that satisfies the Wolfe conditions is globally convergent.

3. If the line search is of the backtracking type and the Hessian is approximated directly (rather than through BFGS updating, so that (3.19) is not needed to guarantee the existence of the update), then the line search need not enforce (3.19).

There is also the practical issue of implementation of the line search. I will address the implementation issue first and leave the theoretical questions for later sections.

### 3.4 Practical line search algorithms

#### 3.4.1 Backtracking algorithms

A backtracking line search can be described as follows. Given \( \overline{\alpha} > 0 \) (\( \overline{\alpha} = 1 \) in the quasi-Newton framework), \( c_1 \in (0, 1) \), and \( \gamma_1, \gamma_2 \) satisfying \( 0 < \gamma_1 < \gamma_2 < 1 \):

1. Set \( \alpha = \overline{\alpha} \).

2. While \( f(x^{(k)} + \alpha p^{(k)}) > f(x^{(k)}) + c_1 \alpha \nabla f(x^{(k)}) \cdot p^{(k)} \)

   (a) Replace \( \alpha \) by a new value in \( [\gamma_1 \alpha, \gamma_2 \alpha] \).

3. Define \( \alpha_k = \alpha \).