It can be shown that if \( x_0 \rightarrow x^* \) superlinearly, then the ratio in this expression converges to 1. If we adjust the choice (3.60) by setting \[ a_0 \leftarrow \min(1, 1.01 a_0) \],

we find that the unit step length \( a_0 = 1 \) will eventually always be tried and accepted, and the superlinear convergence properties of Newton and quasi-Newton methods will be observed.

### A LINE SEARCH ALGORITHM FOR THE WOLFE CONDITIONS

The Wolfe (or strong Wolfe) conditions are among the most widely applicable and useful termination conditions. We now describe in some detail a one-dimensional search useful for some problems. Let us first consider the Wolfe conditions (3.7). Procedure that is guaranteed to find a step length satisfying the Wolfe conditions (3.7) is.

1. Before the Wolfe conditions, we assume that \( p \) is a descent direction and that \( f \) is bounded below along the direction \( p \).

2. The Wolfe algorithm has two stages. This first stage begins with a trial estimate \( a_1 \), and keeps increasing it until it finds either an acceptable step length or an interval that brackets the desired step lengths. In the latter case, the second stage is invoked by calling a function called zoom (Algorithm 3.6, below), which successively decreases the size of the interval until an acceptable step length is identified.

A formal specification of the line search algorithm follows. We refer to (3.7a) as the sufficient decrease condition and to (3.7b) as the curvature condition. The parameter \( 
\text{Algorithm 3.5 (Line Search Algorithm).} 
\begin{align*}
\text{Set } a_0 &\leftarrow 0, \text{ choose } \alpha_{\text{min}} > 0 \text{ and } a_1 \in (0, \alpha_{\text{min}}); \\
n &\leftarrow 1; \\
\text{repeat} & \\
\text{Evaluate } \phi(a_n); \\
\text{if } \phi(a_n) > \phi(0) + \alpha_n \phi'(0) \text{ or } [\phi(a_n) \geq \phi(a_{n-1}) \text{ and } n > 1] & \\
\text{Set } a_n \leftarrow \text{zoom}(a_{n-1}, a_n) \text{ and stop}; \\
\text{Evaluate } \phi'(a_n); \\
\text{if } [\phi'(a_n) \leq c \phi'(0) \\
\text{set } a_n \leftarrow a_n \text{ and stop}; \\
\text{if } \phi'(a_n) \geq 0 \\
\text{Set } a_n \leftarrow \text{zoom}(a_{n-1}, a_n) \text{ and stop}; \\
\text{Choose } \alpha_{n+1} \in (a_n, \alpha_{\text{max}}); \\
n &\leftarrow n + 1; \\
\text{end (repeat)} 
\end{align*}

Note that the sequence of trial step lengths \( \{a_n\} \) is monotonically increasing, but that the order of the arguments supplied to the zoom function may vary. The procedure uses the knowledge that the interval \([a_{n-1}, a_n]\) contains step lengths satisfying the strong Wolfe conditions if one of the following three conditions is satisfied:

1. \( a_n \) violates the sufficient decrease condition;
2. \( \phi(a_n) \leq \phi(a_{n-1}) \);
3. \( \phi'(a_n) \geq 0 \).

The last step of the algorithm performs extrapolation to find the next optimal value \( a_{n+1} \). To implement this step, we can use approaches like the interpolation techniques above, or we can simply set \( a_{n+1} \) to some constant multiple of \( a_n \). Whichever strategy we use, it is important that the successive steps increase quickly enough to reach the upper limit \( a_{\text{max}} \) at a finite number of iterations.

We now specify the function \( \text{zoom} \), which requires a little explanation. The order of its input arguments is such that each call has the form \( \text{zoom}(a_{n-1}, a_n) \), where

1. \( a_n \) is the interval bounded by \( a_{n-1} \) and \( a_n \), contains step lengths that satisfy the strong Wolfe conditions;
2. \( a_{n-1} \) is, among all step lengths generated so far and satisfying the sufficient decrease condition, the one giving the smallest function value; and
3. \( a_n \) is chosen so that \( \phi'(a_n)/(a_n - a_{n-1}) < 0 \).

Each iteration of \( \text{zoom} \) generates an iterate \( a_n \) between \( a_{n-1} \) and \( a_n \) and then replaces one of these end points by \( a_n \) in such a way that the properties (a), (b), and (c) continue to hold.

\textbf{Algorithm 3.6 (zoom).} 

repeat 
\begin{align*}
\text{Interpolate (using quadratic, cubic, or bisection) to find a step length } a_{n+1} \text{ between } a_{n-1} \text{ and } a_n; \\
\text{Evaluate } \phi(a_{n+1}); \\
\text{if } \phi(a_{n+1}) > \phi(0) + c a_{n+1} \phi'(0) \text{ or } [\phi(a_{n+1}) \geq \phi(a_n) \text{ and } n > 1] & \\
\text{go to step 2; } \\
\text{Evaluate } \phi'(a_{n+1}); \\
\text{if } [\phi'(a_{n+1}) \leq c \phi'(0) \\
\text{set } a_{n+1} \leftarrow a_n \text{ and stop}; \\
\text{if } \phi'(a_{n+1}) \geq 0 \\
\text{Set } a_{n+1} \leftarrow \text{zoom}(a_{n-1}, a_n) \text{ and stop; } \\
\text{Choose } \alpha_{n+1} \in (a_{n+1}, \alpha_{\text{max}}); \\
n &\leftarrow n + 1; \\
\text{end (repeat)} 
\end{align*}
3.5. Step-Length Selection Algorithms

If the new estimate \( \alpha_i \) happens to satisfy the strong Wolfe conditions, then zoom has served its purpose of identifying such a point, so it terminates with \( \alpha_i = \alpha_j \). Otherwise, if \( \alpha_i \) is too small or too large, then we continue with \( \alpha_i \) to maintain condition (i), so \( \alpha_{i+1} = \alpha_i \) to maintain condition (b). If this setting results in a violation of condition (c), we increase \( \alpha_i \) by a factor of 0.1 to satisfy condition (a). If this new value is too large, then the line search is stopped and the current value of \( \alpha_i \) is returned. Readers should try to work through these examples to see for themselves how zoom works.

As explained earlier, the interpolation step that determines \( \alpha_i \) should be safeguarded to ensure that the new step length is not too close to the endpoints of the interval. Practical experience shows that line search algorithms also make use of the properties of the interpolating polynomials to make educated guesses of where the next step length should lie; see [99, 216]. A problem that arises is that the optimization algorithm approach the solution, and it is possible to choose a step size that is too small, leading to a violation of the Wolfe conditions. Therefore, the line search algorithm must include a stopping test to ensure that it cannot attain a lower value. Some procedures also consider the value of \( \alpha_i \) to be close to machine precision, or to some user-specified value. This approach is limited by the fact that the relative change in \( \alpha_i \) is close to machine precision, or to some user-specified value.

A line search algorithm that incorporates all these features is difficult to code. We advocate the use of one of the several good software implementations available in the public domain. See Dennis and Schnabel [92], Luenberger [119], Fletcher [101], More and Thuente [216] (in particular), and Hager and Zhang [98].

One may ask how much more expensive it is to require the strong Wolfe conditions instead of the regular Wolfe conditions. Our experience suggests that for a “loose” line search (with parameters such as \( \xi = 10^{-1} \) and \( \eta = 0.9 \)), both strategies require a similar amount of work. The strong Wolfe conditions have the advantage that by decreasing \( \eta \), we can directly control the quality of the search, forcing the accepted value of \( \alpha_i \) to be close to the local minimum. This feature is important in steepest descent or nonlinear conjugate gradient methods, and therefore a step selection routine that enforces the strong Wolfe conditions has widespread applicability.

NOTES AND REFERENCES

For an extensive discussion of line search termination conditions see Ortega and Rheinboldt [210]. Also, see [20] for a probabilistic analysis of the steepest descent method with exact line searches on quadratic functions. He shows that when \( n > 2 \), the worst-case bound (3.29) can be expected to hold for most starting points. The case \( n = 2 \) can be studied in closed form (see Barzilai and Borwein [114]). Theorem 3.6 is due to Dennis and More.

Some line search methods (see Golub and Stewart [132] and Moré and Sorensen [213]) choose a direction of negative curvature, whenever it exists, to prevent the iteration from converging to minimizing stationary points. A direction of negative curvature is one that satisfies the following condition: a direction of negative curvature \( p_i \) is one that satisfies the condition \( p_i^T \nabla^2 f(x_i) p_i < 0 \). These algorithms generate a search direction by combining \( p_i \) with the steepest descent direction \( -\nabla f(x_i) \), often performing a curvilinear backtracking line search.

It is difficult to determine the relative contributions of the steepest descent and negative curvature directions. Because of this fact, the approach fell out of favor after the introduction of trust-region methods.

For a more thorough treatment of the modified Cholesky factorization see Gill, Murray, and Wright [130] or Dennis and Schnabel [92]. A modified Cholesky factorization based on Gershgorin disk estimates is described in Schnabel and Eskow [276]. The modified indefinite factorization is from Chen and Hjalmar [118].

Another strategy for implementing a line search Newton method when the Hessian contains negative eigenvalues is to compute a direction of negative curvature and use it to define the search direction (see More and Sorensen [213] and Goldfarb [132]). Derivative-free line search algorithms include golden section and Fibonacci search.

Some share some of the features with the line search method given in this chapter. They typically store three trial points that determine an interval containing a one-dimensional minimizer. Golden section and Fibonacci differ in the way in which the trial step lengths are generated; see, for example, [79, 99].

Our discussion of interpolation follows Dennis and Schnabel [92], and the algorithm for finding a step length satisfying the strong Wolfe conditions can be found in Fletcher [101].

Exercises

3.1. Program the steepest descent and Newton algorithms using the backtracking line search, Algorithm 3.1. Use them to minimize the Rosenbrock function (2.22). Set the initial step length \( \alpha_0 = 1 \) and print the step length used by each method at each iteration. First try the initial point \( x_0 = (1, 1, 1)^T \) and then the more difficult starting point \( x_0 = (1, 1, 1, 1)^T \).

3.2. Show that if \( 0 < \xi < 1 \), there may be no step lengths that satisfy the Wolfe conditions.

3.3. Show that the one-dimensional minimizer of a strongly convex quadratic function is given by (3.35).

3.4. Show that the one-dimensional minimizer of a strongly convex quadratic function always satisfies the Goldenstein conditions (3.11).

3.5. Prove that \( \| r(x) \| \geq \| r(x) \| B^{-1} \) for any nonsingular matrix \( B \). Use this fact to explain (3.19).

3.6. Consider the steepest descent method with exact line searches applied to the convex quadratic function (3.24). Using the properties given in this chapter, show that if the initial point is such that \( -\nabla f(x) \) is parallel to an eigenvector of \( Q \), then the steepest descent method will find the solution in one step.