The Wolfe conditions

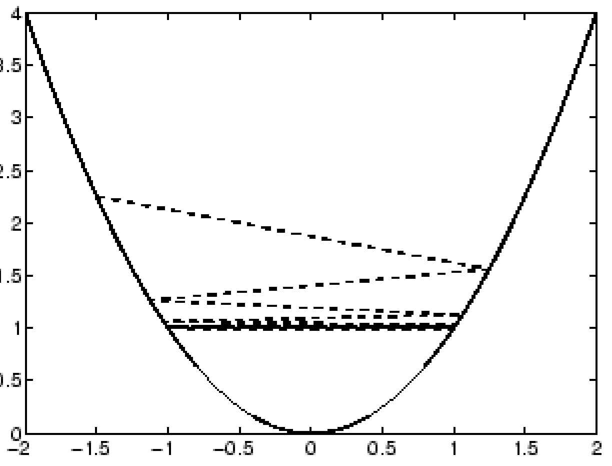
I have been discussing a general strategy for globalizing Newton's method; this strategy is based on one simple goal: reducing *f* at each step. Ideally, having determined a descent direction $p^{(k)}$ at $x^{(k)}$, α_k would be chosen to be the (global) minimizer of the function

$$\phi(\alpha) = f(x^{(k)} + {}^{(k)}),\ \alpha \geq 0.$$

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 ϕ (and, in fact, usually impossible to find the *global* minimizer of ϕ 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 essentially two ways that a line search can continually reduce *f* without reducing it enough to obtain convergence. The first way is illustrated in Figure 2. 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.

Figure 2: 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.

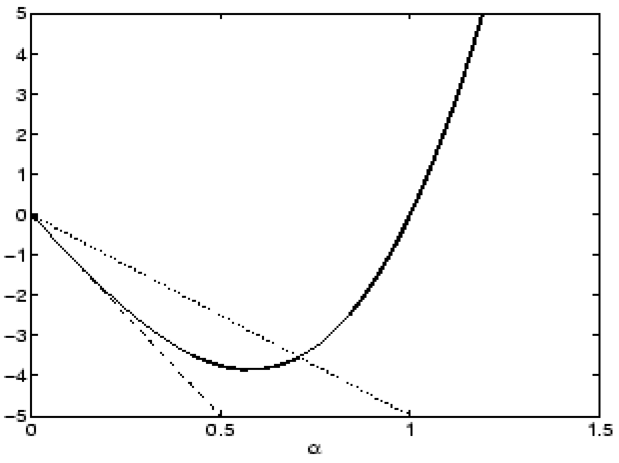


To avoid the problem illustrated by Figure 2, the following condition can be required of the step length α:

$$f(x^{(k)}+\alpha_k p^{(k)}) \le f(x^{(k)}) + c_1 \alpha_k \nabla f(x^{(k)}) \cdot p^{(k)}. \tag{1}$$

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 ϕ can curve upward as α increases from 0, *f* may not attain this decrease for any $\alpha_k > 0$. However, it is easy to show that, for any $c_1 \in (0,1)$, α_k satisfies (1) for all c_1 sufficiently small. On the other hand, (1) prevents the line search from passing over a minimum of *phi* and climbing too far up the other side of the ``valley'' (in such a situation, the predicted decrease in *f* is increasing with α, but the actual decrease is decreasing). The meaning of (1) is illustrated in Figure 3.

Figure: Condition (1) illustrated. The solid curve is ϕ, the dashed line is the tangent line to ϕ at $\alpha = 0$, and the dotted line is determined by (1) with c_1=0.5. In order for step length α to satisfy (1), it has to lie in the interval on which the graph of ϕ is below the graph of the dotted line. The reader should notice how (1) prevents the line search from taking overly long steps.



The second way that an algorithm can reduce *f* without reducing it sufficiently is to take steps that are too short. As a very simple example of this, suppose $f:\mathbf{R}\rightarrow\mathbf{R}$ 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 $\alpha_k=2^{-k-1}$ results in a decrease in *f*. However, $x^{(k)}\rightarrow 1$, 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 squence in the previous paragraph satisfies condition (1) assuming c_1 is chosen sufficiently small (for example, $c_1=0.5$ works, as can be easily shown). Thus condition (1) does not guard against overly short steps. The following condition prevents the step length from being too short:

$$\nabla f(x^{(k)}+\alpha_k p^{(k)})\cdot p^{(k)}\geq c_2\nabla f(x^{(k)})\cdot p^{(k)}. \tag{2}$$

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 (1) and (2) simultaneously. Since $\nabla f(x^{(k)}+\alpha_k p^{(k)})\cdot p^{(k)}$ and $\nabla f(x^{(k)})\cdot p^{(k)}$ are the derivative of ϕ at $\alpha=\alpha_k$ and $\alpha=0$, respectively, (2) simply guarantees that α_k is large enough that the slope of ϕhas increased by some fixed relative amount. This prevents the line search from taking steps that become too small. Condition (2) is satisfied if $\phi'(\alpha_k)$ is very small or positive, which means that α_k close to or beyond a local minimizer of ϕ would be satisfactory. This is another way to see that (2) prevents short steps.

Condition (2) can be written in the equivalent form

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

which implies that $y^{(k)} \cdot s^{(k)} > 0$ (using the familiar notation $s^{(k)}=x^{(k+1)}-x^{(k)}$, $y^{(k)} = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$). Therefore, (2) implies that the BFGS update will be well-defined, a side benefit of this condition on the line search.

Together (1) and (2) 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), the following stronger condition is sometimes used:

$$\left|\nabla f(x^{(k)} + \alpha_k p^{(k)}) \cdot p^{(k)}\right| \leq c_2 \left|\nabla f(x^{(k)}) \cdot p^{(k)}\right|. \tag{3}$$

Using (3), the line search is actually seeking a minimizer, and a small value of c_2 implies an accurate minimization of ϕ. However, as I mentioned above, it is not computationally efficient to perform an accurate minimization during the line search, so the weaker condition (2) is usually preferred in place of (3).

In fact, in the context of a backtracking line search, it is not even necessary to enforce (2) 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, (2) 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 α_k such that both (1) and (2) 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 (2) is not needed to guarantee the existence of the update), then the line search need not enforce (2).

There is also the practical issue of implementation of the line search. I will address the implementation issue first and leave the theoretical questions to another lecture.