

# Globalizing Newton's method: line searches

Mark S. Gockenbach

## 1 Introduction

I have now explained two quasi-Newton algorithms<sup>1</sup> that always produce descent directions. In such an algorithm, at the  $k$ th iteration, a descent direction  $p^{(k)} = -H_k^{-1}\nabla f(x^{(k)})$  for  $f$  at  $x^{(k)}$  is known. 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 that I have described 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 1.1** *To illustrate that the length of the Newton step can be too long in some cases, I define  $f : \mathbb{R} \rightarrow \mathbb{R}$  by*

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

*The function  $f$ , together with the best local quadratic approximation  $Q$  near  $x = 1$ , is graphed in Figure 1. As the graph shows, the step from  $x = 1$  to the minimizer of the quadratic approximation (that is, the Newton step) is not a good estimate of the step from  $x = 1$  to the minimizer  $x^*$  of  $f$ . The problem is that  $f$  is very nonquadratic between  $x = 1$  and  $x = x^*$ .*

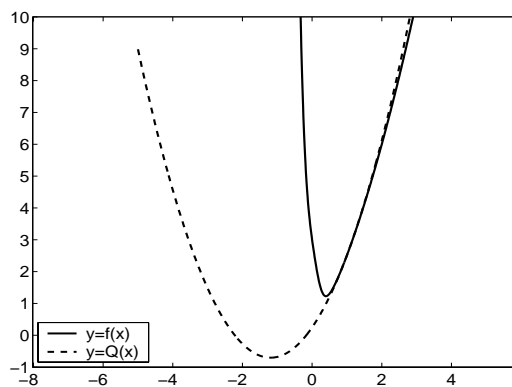


Figure 1: The function  $f$  from Example 1.1 (solid curve) and a quadratic approximation  $Q$  (dashed curve).

---

<sup>1</sup>One algorithm is based on a direct modification of the Hessian, while the other uses the BFGS update. Both yield Hessian approximations  $H_k$  that are always positive definite.

The remedy for the situation illustrated by the previous example is to define

$$x^{(k+1)} = x^{(k)} - \alpha_k H_k^{-1} \nabla f(x^{(k)}),$$

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

$$\alpha \mapsto f(x^{(k)} + \alpha p^{(k)}), \quad p^{(k)} = -H_k^{-1} \nabla f(x^{(k)}),$$

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.

## 2 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.

### 2.1 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  $x^{(k)}$ ,  $\alpha_k$  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 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.

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

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

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  $\phi$  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 (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  $\alpha$ , but the actual decrease is decreasing). The meaning of (1) is illustrated in Figure 3.

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  $f : \mathbb{R} \rightarrow \mathbb{R}$  is defined by  $f(x) = x^2$ .

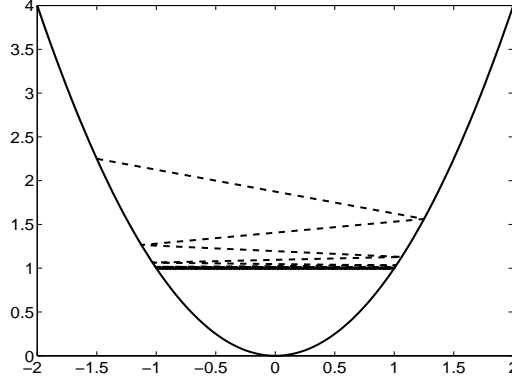


Figure 2: A sequence of steps that reduces a function  $f$  at every step and yet does not converge to a minimizer of  $f$ . The problem with this sequence is that each step achieves very little reduction in  $f$  relative to the length of the step.

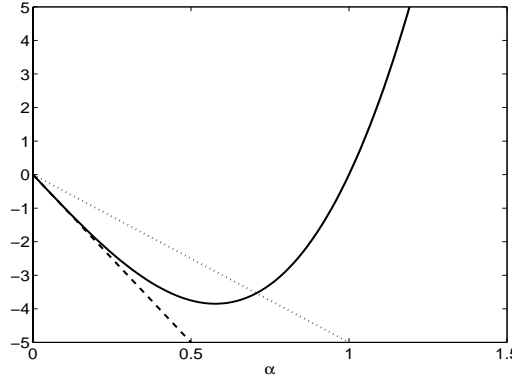


Figure 3: Condition (1) illustrated. The solid curve is  $\phi$ , the dashed line is the tangent line to  $\phi$  at  $\alpha = 0$ , and the dotted line is determined by (1) with  $c_1 = 0.5$ . In order for step length  $\alpha$  to satisfy (1), it has to lie in the interval on which the graph of  $\phi$  is below the graph of the dotted line. The reader should notice how (1) prevents the line search from taking overly long steps.

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 sequence 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:

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

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  $\phi$  at  $\alpha = \alpha_k$  and  $\alpha = 0$ , respectively, (2) 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 (2) 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 (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:

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

Using (3), the line search is actually seeking a minimizer, and a small value of  $c_2$  implies an accurate minimization of  $\phi$ . 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  $\alpha_k$  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.

### 3 Backtracking algorithms

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

1. Set  $\alpha = \bar{\alpha}$ .
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  $\alpha_k = \alpha$ .

The new value of  $\alpha$  in Step 2a is usually chosen by minimizing a quadratic or cubic polynomial interpolating the function  $\phi$  at two or three points. I will explain an interpolation algorithm below. The use of  $\gamma_1, \gamma_2$  is referred to as *safeguarding* the step; typical values are  $\gamma_1 = 0.1, \gamma_2 = 0.5$ . Safeguarding ensures that a reasonable new trial value for  $\alpha$  is chosen even if  $f$  is so nonlinear that

interpolation works poorly. The reader should notice that, as discussed above, only the first Wolfe condition, namely, the sufficient decrease condition, is actually enforced by the above algorithm.

Now I explain how an backtracking algorithm might choose a new value of  $\alpha$  if the current value of  $\alpha$ , say  $\alpha = \hat{\alpha}$ , produces insufficient decrease in  $f$ :

$$f(x^{(k)} + \hat{\alpha}p^{(k)}) > F(x^{(k)}) + c_1\hat{\alpha}\nabla f(x^{(k)}) \cdot p^{(k)}.$$

A simple strategy is to repeatedly replace  $\hat{\alpha}$  by  $\hat{\alpha}/2$  until the sufficient decrease condition is satisfied. (This assumes that  $\gamma_1 \leq 1/2 \leq \gamma_2$ .) In spite of its simplicity, this strategy is fairly effective.

Instead of simply halving  $\hat{\alpha}$ , interpolation can be used. At the beginning of the line search, the values of  $\phi(0)$  and  $\phi'(0)$  are known. In order to test the sufficient decrease condition,  $\phi(\hat{\alpha})$  must also be computed. These three pieces of information determine a quadratic polynomial  $p$  satisfying

$$p(0) = \phi(0), \quad p'(0) = \phi'(0), \quad p(\hat{\alpha}) = \phi(\hat{\alpha}).$$

This polynomial is easily determined to be

$$p(\alpha) = \phi(0) + \phi'(0)\alpha + \frac{\phi(\hat{\alpha}) - \phi(0) - \phi'(0)\hat{\alpha}}{\hat{\alpha}^2}\alpha^2.$$

The unique minimizer of  $p$  is

$$\alpha = -\frac{\phi'(0)\hat{\alpha}}{2(\phi(\hat{\alpha}) - \phi(0) - \phi'(0)\hat{\alpha})}. \quad (4)$$

It is easy to see that the denominator in the above fraction is positive (otherwise, the sufficient decrease condition would have held at  $\hat{\alpha}$ ) and therefore that this new value of  $\alpha$  is positive. If  $\gamma_1 < 0.5$ , then it can be shown that the value of  $\alpha$  given by (4) belongs to the interval  $(0, 1)$ . The updating of  $\hat{\alpha}$  is safeguarded, as mentioned above, so that if the trial value of  $\alpha$  given by (4) lies outside of the interval  $[\gamma_1\hat{\alpha}, \gamma_2\hat{\alpha}]$ , then the new value of  $\hat{\alpha}$  is taken to be  $\gamma_1\hat{\alpha}$  (if  $\alpha < \gamma_1\hat{\alpha}$ ) or  $\gamma_2\hat{\alpha}$  (if  $\alpha > \gamma_2\hat{\alpha}$ ).

If the quadratic interpolation fails to produce a step length satisfying the sufficient decrease condition, then cubic interpolation can be used. The cubic polynomial interpolating  $\phi(0)$ ,  $\phi'(0)$ ,  $\phi(\hat{\alpha})$ , and  $\phi(\bar{\alpha})$  is determined, where  $\hat{\alpha} < \bar{\alpha}$  are the two most recent values of  $\alpha$ . I leave it as an exercise to show that the cubic interpolant has a local minimizer in the interval  $[0, \hat{\alpha}]$  and to derive a formula for this minimizer.

## 4 Satisfying the curvature condition

If the curvature condition (2) or the stronger version (3) must be satisfied (because BFGS updating is to be used, for example), then a simple backtracking line search is not sufficient. This is because the interval  $0 \leq \alpha \leq 1$  may contain no step length satisfying the curvature condition. It would still be usual to begin with step length 1, but then the line search must be prepared to increase  $\alpha$  until an interval is identified that is guaranteed to contain an acceptable point. The details become rather complicated and I refer the reader to the text by Nocedal and Wright [2] (pages 58–61) and the references contained therein (page 61). In particular, Dennis and Schnabel [1] contains pseudo-code that completely specifies such an algorithm.

## References

- [1] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, 1983.
- [2] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.