

# Globalizing Newton's method: Descent Directions (I)

Mark S. Gockenbach

## 1 Introduction

If  $x^*$  satisfies the sufficient conditions for a local minimizer of  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , namely,  $\nabla f(x^*) = 0$  and  $\nabla^2 f(x^*)$  positive definite, then Newton's method is guaranteed to produce rapid local convergence. However, Newton's method may fail to be satisfactory if the starting point  $x^{(0)}$  is not sufficiently close to  $x^*$ . This can happen for one of several reasons:

1. If  $\nabla^2 f(x^{(k)})$  is singular, then the Newton step is not even defined, and obviously some modification of the algorithm is necessary.
2. Even if  $\nabla^2 f(x^{(k)})$  is nonsingular, so that the Newton step is defined, there is no guarantee that it points in a good direction. For example, the Newton step may point uphill, and taking the step might increase the value of  $f$  (not a good thing when the goal is to minimize  $f$ !).
3. Finally, even if the Newton step points downhill, so that its direction is promising, the length of the Newton step might not be appropriate. For example, the step can be too long, and the algorithm might end up stepping over a "valley" in the graph of  $f$  and climbing up the opposite slope.

I will postpone a discussion of the third issue and focus now on modifying Newton's method to eliminate the first two difficulties.

## 2 Descent Directions

The reader will recall that  $p \in \mathbb{R}^n$  is a *descent direction* for  $f$  at  $x^{(k)}$  if

$$\nabla f(x^{(k)}) \cdot p < 0.$$

This condition implies that

$$f(x^{(k)} + \alpha p) < f(x^{(k)}) \text{ for all } \alpha > 0 \text{ sufficiently small.}$$

Indeed, if  $\phi$  represents the one-dimensional "slice" of  $f$  in the direction of  $p$ ,

$$\phi(\alpha) = f(x^{(k)} + \alpha p)$$

then the descent condition implies that  $\phi'(0) < 0$  and hence that  $\phi(\alpha) < \phi(0)$  for all  $\alpha > 0$  sufficiently small (see Figure 1).

Given  $x^{(k)}$  and a descent direction  $p$ , it is possible to reduce  $f$  by moving in the direction of  $p$ , that is, by choosing an appropriate  $\alpha > 0$  and defining  $x^{(k+1)} = x^{(k)} + \alpha p$ . A procedure for choosing  $\alpha$  is referred to as a *line search* (since  $x^{(k+1)}$  is found on the (half-)line parametrized as  $x^{(k)} + \alpha p$ ). I will discuss line searches (a solution to the third difficulty described above in the Introduction) later. For now, I want to concentrate on methods for producing descent directions.

Newton's method produces the direction

$$p = -\nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}).$$

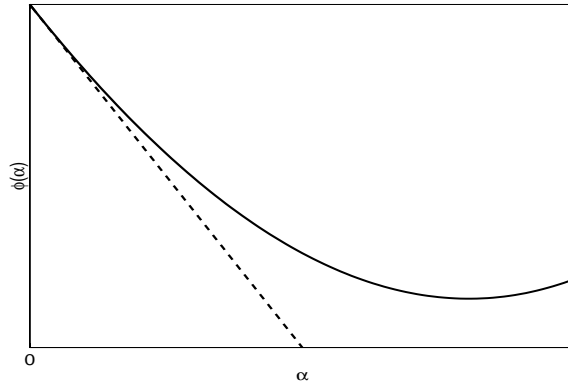


Figure 1: The function  $\phi(\alpha) = f(x^{(k)} + \alpha p)$ , where  $p$  is a descent direction..

This is a descent direction if

$$\nabla f(x^{(k)}) \cdot p < 0,$$

that is, if

$$-\nabla f(x^{(k)}) \cdot \nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}) < 0,$$

that is, if

$$\nabla f(x^{(k)}) \cdot \nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}) > 0. \quad (1)$$

Condition (1) will hold if  $\nabla^2 f(x^{(k)})^{-1}$  is positive definite. The reader may recall that the eigenvalues of  $H^{-1}$  are simply the reciprocals of the eigenvalues of  $H$ ,<sup>1</sup> and therefore  $\nabla^2 f(x^{(k)})^{-1}$  is positive definite if and only if  $\nabla^2 f(x^{(k)})$  is positive definite. Of course, if  $\nabla^2 f(x^{(k)})$  is positive definite, then it is nonsingular, and in this case the first two difficulties mentioned in the Introduction disappear. *If  $\nabla^2 f(x^{(k)})$  is positive definite, then the Newton step is well-defined and represents a descent direction.*

The following observation is essential: If  $H$  is *any* symmetric positive definite matrix, then  $p = -H^{-1} \nabla f(x^{(k)})$  is a descent direction for  $f$  at  $x^{(k)}$ . This suggests the following modification of Newton's method:

$$x^{(k+1)} = x^{(k)} - \alpha_k H_k^{-1} \nabla f(x^{(k)}), \quad (2)$$

where  $H_k$  is a positive definite approximation of  $\nabla^2 f(x^{(k)})$  and  $\alpha_k$  is a *step-length parameter* that is chosen by a line search. An iteration of the form (2) is referred to as a *quasi-Newton* iteration. In order for the rapid local convergence of Newton's method to be preserved,  $H_k$  should be a good approximation to (or equal to)  $\nabla^2 f(x^{(k)})$  when  $x^{(k)}$  is close to  $x^*$ .

I will now describe two specific quasi-Newton methods.

### 3 The steepest-descent method

The simplest descent direction for  $f$  at  $x^{(k)}$  is  $p = -\nabla f(x^{(k)})$ , which is referred to as the *steepest-descent* direction. The resulting iteration is

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

This can be viewed as a quasi-Newton iteration with  $H_k = I$  for all  $k$ . That this is a poor choice of Hessian approximation can be deduced from the fact that it is completely independent of  $f$ , and specifically of the curvature of  $f$ .

I will show later that (with an appropriate line search) the steepest-descent algorithm has good global convergence properties, but the convergence can be quite slow.

<sup>1</sup>The proof is simple:

$$Hx = \lambda x \Leftrightarrow x = H^{-1}(\lambda x) = \lambda H^{-1}x \Leftrightarrow \lambda^{-1}x = H^{-1}x.$$

## 4 Direct modification of the Hessian

A popular strategy for defining a quasi-Newton method is to replace  $\nabla^2 f(x^{(k)})$  by

$$H_k = \nabla^2 f(x^{(k)}) + E_k,$$

where  $E_k$  is a symmetric positive definite matrix with the property that  $H_k$  is positive definite. If  $\lambda_{\min}(A)$  denotes the smallest eigenvalue of a symmetric matrix  $A$ , then<sup>2</sup>

$$y \cdot Ay \geq \lambda_{\min}(A)\|y\|^2 \text{ for all } y \in \mathbb{R}^n.$$

Therefore,

$$y \cdot H_k y = y \cdot \nabla^2 f(x^{(k)})y + y \cdot E_k y \geq \left( \lambda_{\min}(\nabla^2 f(x^{(k)})) + \lambda_{\min}(E_k) \right) \|y\|^2,$$

which shows that  $H_k$  is positive definite if

$$\lambda_{\min}(E_k) > \left| \lambda_{\min}(\nabla^2 f(x^{(k)})) \right|.$$

### 4.1 Strategy 1

In order that  $H_k$  be as close as possible to  $\nabla^2 f(x^{(k)})$ ,  $E_k$  should be chosen as small as possible. A natural choice for  $E_k$ , then, is

$$E_k = \tau I,$$

where

$$\tau > \left| \lambda_{\min}(\nabla^2 f(x^{(k)})) \right|. \quad (3)$$

There is another reason for choosing  $E_k = \tau I$ : For each  $\tau$  satisfying (3), there exists  $\Delta > 0$  such that

$$p = -(\nabla^2 f(x^{(k)}) + \tau I)^{-1} \nabla f(x^{(k)})$$

is the solution of

$$\begin{aligned} \min \quad & f(x^{(k)}) + \nabla f(x^{(k)}) \cdot p + \frac{1}{2} p \cdot \nabla^2 f(x^{(k)}) p \\ \text{s.t.} \quad & \|p\| \leq \Delta. \end{aligned}$$

This means that  $p$  minimizes the best local quadratic approximation to  $f$  subject to the constraint that the step not be longer than  $\Delta$ . Although the relationship between  $\tau$  and  $\Delta$  is nonlinear and complicated, it is easy to show that:

- $\Delta \rightarrow 0$  as  $\tau \rightarrow \infty$ ;
- the direction of  $p = -(\nabla^2 f(x^{(k)}) + \tau I)^{-1} \nabla f(x^{(k)})$  converges to the steepest descent direction as  $\tau \rightarrow \infty$ .

Therefore, at the worst, choosing a large  $\tau$  will produce a short step (nearly) in the steepest descent direction, which is a reasonable thing to do when  $\nabla^2 f(x^{(k)})$  does not define a meaningful step. (However, if  $\tau$  is consistently chosen *too* large, then the short steps will slow the convergence of the algorithm.)

Various strategies for choosing  $\tau$  have been proposed and implemented, and it is not clear which is best.<sup>3</sup> However, it is essential that any such strategy choose  $\tau = 0$  in the case that  $\nabla^2 f(x^{(k)})$  is positive definite. In this way, the rapid local convergence of Newton's method is assured (since  $H_k = \nabla^2 f(x^{(k)})$  will be used when  $x^{(k)}$  is close enough to  $x^*$ , assuming  $\nabla^2 f(x^*)$  is positive definite).

<sup>2</sup>To prove this, express  $y$  in terms of an orthonormal basis of eigenvectors of  $A$ .

<sup>3</sup>A simple and effective strategy is given as Algorithm 6.3 on page 145 in Nocedal and Wright [1].

## 4.2 Strategy 2

A second strategy for modifying the Hessian when it fails to be positive definite is to take  $H_k = \nabla^2 f(x^{(k)}) + E_k$ , where  $E_k$  is a general diagonal matrix produced “on the fly” when trying to factor  $\nabla^2 f(x^{(k)})$ . In order to describe this method, I need to explain some numerical linear algebra.

### 4.2.1 The Cholesky factorization

I often write formulas involving the inverse of a matrix, for example,

$$x^{(k+1)} = x^{(k)} - \nabla^2 f(x^{(k)})^{-1} \nabla f(x^{(k)}). \quad (4)$$

In practice, however, it is rarely necessary or desirable to compute a matrix inverse. Rather, the presence of a matrix inverse in a formula such as the Newton iteration is a sign that a linear system must be solved. For example, (4) should be interpreted as specifying a two-step procedure:

1. Solve the linear system  $\nabla^2 f(x^{(k)})p = -\nabla f(x^{(k)})$  to get the Newton step  $p$ .
2. Define  $x^{(k+1)} = x^{(k)} + p$ .

I will now explain how a general linear system  $Ax = b$  is solved, with emphasis on the case in which  $A$  is symmetric and positive definite.

The reader will recall the Gaussian elimination algorithm for solving  $Ax = b$ , which is usually carried out as follows:

1. Form the “augmented matrix”  $[A|b]$ .
2. Perform row operations systematically to reduce the augmented matrix to “row echelon form,”  $[U|\tilde{b}]$ . If  $A$  is nonsingular, then  $U$  is an upper triangular matrix.
3. Perform more row operations to reduce  $[U|\tilde{b}]$  to  $[I|x]$ , where  $x$  is the solution. This stage is referred to “back substitution” because it amounts to solving the last equation for  $x_n$ , then substituting its value into the next-to-last equation to find  $x_{n-1}$ , and so forth.

Some authors refer to the second step alone as Gaussian elimination and to the entire algorithm as Gaussian elimination with back substitution. Others just refer to the entire algorithm as Gaussian elimination.

In modern software,<sup>4</sup> the process of reducing  $A$  to upper triangular form is usually carried out first, without any reference to the right-hand side  $b$ . If no row interchanges are required when reducing  $A$  to  $U$ , then the row operations that are performed can be recognized as multiplication by  $L^{-1}$ , where  $L$  is a certain lower triangular matrix:  $U = L^{-1}A$ . It follows that  $A = LU$ , which is referred to (unimaginatively) as the LU factorization of  $A$ . Solving  $Ax = b$  is then performed by a two-step process:

1. Factor  $A = LU$
2. Solve  $Ly = b$  to get  $y = L^{-1}b$  and then solve  $Ux = y$  to get  $x = U^{-1}y = U^{-1}L^{-1}b = A^{-1}b$ .

Commonly the first step is performed by one subroutine and the second step by another. If row interchanges are necessary when reducing  $A$  to  $U$ , then certain permutation matrices must be introduced, complicating the formulas considerably and the software to some degree.

When  $A$  is symmetric and positive definite, the above algorithm simplifies in two ways. First of all, it can be proved that row interchanges are never necessary, and so  $A$  is guaranteed to have an  $LU$  factorization. Second, because of the symmetry of  $A$ ,  $U = L^T$ . The resulting factorization,  $A = LL^T$ , is referred to as the *Cholesky factorization* of  $A$ . The Cholesky factorization can also be written as  $A = LDL^T$  (different  $L!$ ), where  $L$  is a lower triangular matrix with ones on the diagonal and  $D$  is a diagonal matrix with positive diagonal entries.

---

<sup>4</sup>The most popular software for numerical linear algebra is LAPACK.

I will now derive the factorization  $A = LL^T$ . By the definition of matrix-matrix multiplication,

$$A = LL^T \Rightarrow A_{ij} = \sum_{k=1}^n L_{ik}L_{jk}, \quad i, j = 1, 2, \dots, n.$$

Since  $L$  is lower triangular by hypothesis, the index  $k$  can be restricted so that the above summation does not involve any elements from the strict upper triangle of  $L$ :

$$A_{ij} = \sum_{k=1}^{\min\{i,j\}} L_{ik}L_{jk}, \quad i, j = 1, 2, \dots, n.$$

In particular,

$$A_{11} = L_{11}^2,$$

which shows that  $L_{11} = \sqrt{A_{11}}$ . It can be shown that all the diagonal entries of a positive definite matrix are positive, so the square root produces a real number. On the other hand, if  $A_{11}$  fails to be positive, then the algorithm can detect that  $A$  is not positive definite. Having computed  $L_{11}$ , it is possible to compute the remainder of the first column of  $L$ :

$$L_{i1}L_{11} = A_{i1} \Rightarrow L_{i1} = L_{11}^{-1}A_{i1}, \quad i = 2, 3, \dots, n.$$

In general, having computed the first  $j - 1$  columns of  $L$ , the  $j$ th column can be computed as follows. First of all,

$$\sum_{k=1}^j L_{jk}L_{jk} = A_{jj} \Rightarrow L_{jj} = \sqrt{A_{jj} - \sum_{k=1}^{j-1} L_{jk}L_{jk}}.$$

It can be shown that if  $A$  is positive definite, then

$$A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2 > 0$$

must hold, and therefore, once again, taking the square root is allowed. On the other hand, if

$$A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2 \leq 0$$

is detected, this demonstrates the  $A$  is not positive definite. Finally, having computed  $L_{jj}$ , the remainder of column  $j$  of  $L$  can be computed as follows:

$$\sum_{k=1}^j L_{ik}L_{jk} = A_{ij} \Rightarrow L_{ij} = L_{jj}^{-1} \left( A_{ij} - \sum_{k=1}^{j-1} L_{ik}L_{jk} \right), \quad i = j + 1, j + 2, \dots, n.$$

If the above algorithm can be carried out to produce  $A = LL^T$ , then it is easy to see that  $A$  is positive definite. The matrix  $L$  must be nonsingular, since it is lower triangular with positive diagonal entries. But then

$$y \neq 0 \Rightarrow y \cdot Ay = y \cdot LL^T y = (L^T y) \cdot L^T y = \|L^T y\|^2 > 0.$$

(Since  $L^T$  is nonsingular,  $L^T y = 0$  is only possible if  $y$  is the zero vector.) Thus  $A$  is positive definite if and only if the above algorithm can be carried out to completion, which means that all of the quantities

$$A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2, \quad j = 1, 2, \dots, n,$$

are positive. Here is the crucial observation: If, in the course of trying to factor  $A$ , it is observed that some

$$A_{jj} - \sum_{k=1}^{j-1} L_{jk}^2$$

fails to be positive, then  $E_{jj}$  can be chosen to be positive and large enough that

$$A_{jj} + E_{jj} - \sum_{k=1}^{j-1} L_{jk}^2 > 0$$

holds. Since this is the only point in the algorithm at which  $A_{jj}$  is used, the effect is the same as if  $E$  were added to  $A$  before the factorization began. In the end, a diagonal matrix  $E$  and a lower triangular matrix  $L$  are computed to satisfy

$$A + E = LL^T.$$

It should be noticed that, if  $A$  is positive definite, then  $E$  will be zero and  $A$  will not be modified.

Using  $A = \nabla^2 f(x^{(k)})$ , this is a popular method for modifying a Hessian that fails to be positive definite. I have not presented the crucial detail of how  $E_{jj}$  is chosen. In numerical mathematics, if a zero quantity will cause an algorithm to be undefined, a very small quantity will usually render the algorithm numerical unstable. Therefore, it is not enough that

$$A_{jj} + E_{jj} - \sum_{k=1}^{j-1} L_{jk}^2$$

be positive; it is necessary to ensure that this quantity is not too small. Details can be found in Algorithm 6.5 (page 148) of Nocedal and Wright [1].

## References

- [1] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.