Numerical Optimization Ch.3

This is a note doc of *Numerical Optimization*.

- Algorithm 3.6 (zoom).

Line Search Methods

Each iteration of a line search method computes a search directoin p_k and then decides how far to move along that direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k$$

where α_k is called the step length.

Most line search method require p_k to be a *desent directoin* -- one for which $p_k^T \nabla f_k < 0$ -- because this property guarantees that the function f can be reduced along this direction. Morever, the search direction often has the form

$$p_k = -B_k^{-1} \nabla f_k$$

where B_k is a symmtric and nonsingular matrix. Here are some B_k:

stepest descent method	Newton's method	quasi-Newton method
I	$\nabla^2 f_k$	Approximation of Hessian

Step Length

When compute α_k, we face a tradeoff: we would like to choose α_k to give a substantial reduction of f, but we do not want to spend too much time. The ideal choice would be the global minimizr of the univariate function $\phi(\cdot)$ defined by

$$\phi(\alpha) = f(x_k + \alpha p_k), \quad \alpha > 0$$

But in general, it is too expensive to get the value. More practical strategies prform an *inexact* line search to identify the α_k.

Typical line search algorithm:

1. A bracketing phase finds an interval containing desirable step lengths
2. A bisection or interpolation phase computes a good step length within this interval

[Sophisticated line search algorithms will be described later.]

The effective step lengths need not lie near minimizers of $\phi(\alpha)$

A simple condition we could impose on α_k is to require a reduction in f, i.e. $f(x_k + \alpha_k p_k) < f_k$, however, it is insufficient. In practical use, we need to enforce a **sufficient decrease** condition.

The Wolfe Conditions

A popular inexact line search condition stipulates that α_k should first of all give *sufficient decrease* in the objective function f, as measured by the following inequality:

$$f(x_k + \alpha p_k) \leq f_k + c_1 \alpha \nabla f_k^T p_k$$

for some constant $c_1 \in (0, 1)$. In other words, the reduction in f should be proportional to both the step length α_k and the directional derivative $\nabla f_k^T p_k$. This inequality is sometimes called the **Armijo condition**.

Denote the RHS as $l(\alpha)$, since $c_1 \in (0, 1)$, it lies above the $\phi(\alpha)$ for some small positive values of α. The sufficient decrease conditoin states that α is acceptable only if $\phi(\alpha) \leq l(\alpha)$. In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$.

The sufficient decrease condditon is not enoug by itself to nsure that the algorithm sufficient because it holds for all sufficiently small values of α. To rule out unacceptably short steps we introduce a second requirement, called the *curvature conditioon*, which requires α_k to satisfy

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k$$

for some constant $c_2 \in (c_1, 1)$. Note that the LHS is simply $\phi'(\alpha_k)$, so the curvature condition ensures that the slope of ϕ at α_k is greater than c_2 times the initial slope $\phi'(0)$, i.e. $\phi'(\alpha) \geq c_2 \phi'(0)$.
This makes sense because if the slope $\phi'(\alpha)$ is strongly negative, we have an indication that we can reeduce f significantly by moving further along the chosen direction. On the other hand, if $\phi'(\alpha)$ is only slightly negative or even positive, it is a sign that we cannot expect much more decrease in f in this direction, so it makees snse to terminate the line search.

The sufficient decrease and curvature conditions are known collectively as the **Wolfe Conditions**:

$$f(x_k + \alpha_k p_k) \leq f_k + c_1 \alpha_k \nabla f_k^T p_k, (\phi(\alpha_k) \leq l(\alpha) = f_k + \alpha c_1 \phi'(0))$$

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f_k^T p_k, (\phi'(\alpha) \geq c_2 \phi'(0))$$

with $0 < c_1 < c_2 < 1$.

A step length may satisfy the Wolfe conditions without being particularly close to a minimizer of ϕ. We can, however, modify the curvature condition to force α_k to lie in at least a broad neighborhood of a local minimizer or stationary point of ϕ.
The **strong Wolfe condition** requires α_k satisfy:

$$f(x_k + \alpha_k p_k) \leq f_k + c_1 \alpha_k \nabla f_k^T p_k,$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq c_2 |\nabla f_k^T p_k|,$$

with $0 < c_1 < c_2 < 1$. The only difference with the wolfe conditions is that we no longer allow the derivative $\phi'(\alpha)$ to be too positive. Hence, we exclude points that are far from stationary points of ϕ.

It is not difficult to prove there exist α_k satisfy the Wolfe conditions for every f that is smooth and bounded below.

Proof

Note $\phi(\alpha) = f(x_k + \alpha p_k)$ is bounded below $\forall \alpha > 0$ while $l(\alpha) = f_k + \alpha c_1 \nabla f_k^T p_k$ is unbounded below and must there exists $\alpha' > 0$ s.t. $\phi(\alpha') = l(\alpha')$.
Notice $\phi(0) = l(0)$, we have $\phi(\alpha') - \phi(0) = l(\alpha') - l(0) = \alpha' c_1 \nabla f_k^T p_k$
Meanwhile, $\phi(\alpha') - \phi(0) = \alpha' \phi'(\alpha'') \Rightarrow \phi'(\alpha'') = c_1 \nabla f_k^T p_k > c_2 \nabla f_k^T p_k$
Therefore, α'' satisfies the conditions.

The Goldstein Conditions

The **Goldstein Conditions** ensure that the α_k achieves sufficient decrease but is not too short. The Goldstein conditions can be stated as a pair of inequalities as:

$$f_k + (1 - c)\alpha_k \nabla f_k^T p_k \leq f(x_k + \alpha_k p_k) \leq f_k + c\alpha_k \nabla f_k^T p_k$$

with $0 < c < 1/2$. The second inequality is the Armijo condition, whereas the first inequality is introduced to control the step length from below.

A *disadvantage* of the Goldsein conditions v.s. the Wolfe conditions is that the first inequality may exclud all minimizer of ϕ. However, they have much in common, and their convergence theories are quite similar. The Goldstein conditions are often used in Newton-type meehods but are not well suited for quasi-Newton methods that maintain a positive definite Hessian approximation.

Sufficient Decrease and Backtracking

We have mentioned that the Armijo condition alonee is not sufficient to ensure the algorithm makes reasonable progress along the given search direction.
However, if the line search algorithm chooses its candidate step lengths appropriately, by using a so-called **backtracking** approach, we can dispense with the extra condition and use just the sufficient decrease condition to terminate the line search procedure. In most basic form, backtracking proceeds as follows.

> Backtracking Line Search

```
Choose alpha0 > 0,  0<rho<1, 0<c<1; set alpha <- alpha0;
while f(xk + alpha*pk) > f(xk) + c*alpha*f'(xk)**pk
    alpha <- rho*alpha;
end
return alphak = alpha
```

In this procedure, the initial step length α_0 is chosen to be 1 in Newton and quasi-Newton methods, but can have differnt values in other algorithms.
In practice, the contraction factor ρ is often allowed to vary at each iteration of the line search. For example, it can be chosen by safeguarded interpolation. We need ensure only that at each iteration we have $\rho \in [\rho_l, \rho_h]$, for some fixed constants $0 < \rho_l < \rho_h < 1$.
The backtracking approach ensures either that the selected step length is some fixed value(α_0), or else taht is short enough to satisfy the sufficient decrease condition but not too short.

This simple and popular strategy for terminating a line search is well suited for Newton methods but is less appropriate for quasi-Newton and conjugate gradient methods.

Convergence of Line Search Methods

To obtain *global convergence*, we must not only have well chosen step lengths α_k but also well chosen search directions p_k.
This section, we will focus on one key property: the angle θ_k between p_k and the steepest descent direction, defined by

$$\cos\theta_k = \frac{-\nabla f_k^T p_k}{\|\nabla f_k\|\|p_k\|}$$

The following theorem quantifies the effect of properly α_k and **the steepest descent method is globally convergent**. For other algorithms, it describes how far p_k can deviate from the steepest descent direction and still produce a globally convergent iteration.

Theorem 3.2(Zoutendijk)

For a line search method which is satisfied Wolfe conditions, suppose that f is bounded below in $\mathrm{R^n}$ and that f is continuously differentiable in an open set $\mathcal{N}$ containing the level set $\mathcal{L} \overset{def}{=} \{x : f(x) \le f(x_0)\}$, where x_0 is the starting point of the current iteration. Assume also that the gradient ∇f is Lipschitz continuous on $\mathcal{N}$, that is, there exists a constant $L > 0$ s.t.

$$\|\nabla f(x) - \nabla f(\bar{x})\| \le L\|x - \bar{x}\|, \quad \forall x, \bar{x} \in \mathcal{N}$$

Then

$$\sum_{k\ge 0} \cos^2\theta_k \|\nabla f_k\|^2 < \infty$$

Proof

From the Wolfe condtions 2, we have

$$(\nabla f_{k+1} - \nabla f_k)^T p_k \geq (c_2 - 1)\nabla f_k^T p_k,$$

while the Lipschitz condition implies that

$$(\nabla f_{k+1} - \nabla f_k)^T p_k \leq \alpha_k L \|p_k\|^2.$$

Then we have

$$\alpha_k \geq \frac{c_2 - 1}{L} \frac{\nabla f_k^T p_k}{\|p_k\|^2}$$

Substituing this into the Wolfe conditions 1, we obtain

$$f_{k+1} \leq f_k - c_1 \frac{c_2 - 1}{L} \frac{(\nabla f_k^T p_k)^2}{\|p_k\|^2}$$

Using the definition of θ_k, we can write this relation as

$$f_{k+1} \leq f_k - c \cos^2 \theta_k \|\nabla f_k\|^2$$

where $c = c_1(c_2 - 1)/L$. By summation, we obtain

$$f_{k+1} \leq f_0 - c \sum_{j=0}^{k} \cos^2 \theta_j \|\nabla f_j\|^2$$

Since f is bounded below, we have that $f_0 - f_{k+1}$ is less than some positive constant, for all k. Hence, by taking limis, we obtain

$$\sum_{k=0}^{\infty} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty.$$

q.e.d.

Similar results to this theorem hold when the Goldstein conditions or strong Wolfe conditions are used in place of the Wolfe conditions.
Note that the *assumptions* we used are actually not too restrictive:

1. If the function f were not bounded below, the optimization problem would not be well defined.
2. The smoothness assumption--Lipschitz continuity of the gradient--is implied by many of the smoothness conditions that are used in local convergence theorems and are often satisfied in practice.

The Zoutendijk condition($\sum_{k=0}^{\infty} \cos^2 \theta_k \|\nabla f_k\|^2 < \infty$) implies that

$$\cos^2 \theta_k \|\nabla f_k\|^2 < \infty \to 0$$

This limit can be used in turn to derive global convergence results for line search algorithms.
If p_k ensures that θ_k is bounded away from 90°, i.e. there is a positive constant δ such that

$$\cos\theta_k \geq k > 0, \quad \forall k$$

Then we immediately get the result:

$$\lim_{k\to\infty} \|\nabla f_k\| = 0$$

In other words, we can be sure that the gradient norms $\|\nabla f_k\|$ converge to zero, when provided that the search directions are never too close to orthogonality with the gradient.
Since the p_k of steepest descent is parallel to the negative gradient, the method produces a gradieent sequencee that converges to zero, when the Wolfe or Goldstein conditions is satisfied.

If an algorithm satisfies Zoutendijk condition, we say the algorithm is **globally convergent**, however, *globally convergent* is sometimes used in otheer contexts to mean different things.

For line search methods, the Zoutendijk condition is strongest global convergence result that can be obtained: we can not guarantee that the method converges to a minimizer, but only that it is attracted by stationary points.
Only by making additional requirements on p_k--by introducing negative curvature information from the Hessian $\nabla^2 f_k$,, for example--can we strengthen these results to include convergence to a local minimum.

Consider now the *Newton-like* method and assume that the matrices B_k are positive definite with au niformly bounded condition number. That is,there is a constant M s.t.

$$\|B_k\|\|B_k^{-1}\| \leq M, \quad \forall k$$

It is easy with the result ($\|Bx\| \geq \|x\|/\|B^{-1}\|$) to show that

$$\cos\theta_k \geq 1/M$$

Then, we have

$$\lim_{k\to\infty} \|\nabla f_k\| = 0$$

Therefore, we have shown that Newton and quasi-Newton methods are globally convergent if the matrices B_k have a bounded condition number and are positive definite (which is needed to ensure that p_k is a descent direction), and if the step lengths satisfy the Wolfe conditions.

For some algorithms, such as *conjugate gradient methods*, we will be able to prove the converge result, but only the weaker result

$$\liminf_{k\to\infty} ||\nabla f|| = 0$$

The detail will be talked later.

Consider *any* algorithm for which

1. every iteration produces a decrease in the objective function
2. every mth iteration is a steepest descent step, with step length chosen to satisfy the Wolfe or Goldstein conditions.

Then, since $\cos\theta_k = 1$ for the steepest descent steps, the $\liminf$ result holds. Of course, we would design the algorithm so that it does something “better" than steepest descent at the other $m-1$ iterates. The occasional steepest descent steps may not make much progress, but they at least guarantee overall global convergence.

Rate of Convergence

It would seem that designing optimization algorithms with good convergence properties is easy.
We could simply compute $\cos\theta_k$ at every iteration and turn p_k toward the steepest descent direction if $\cos\theta_k$ is smaller than some preselected constant $\delta > 0$.
Angle tests of this type ensure global convergence, but they are undesirable for two reasons.

1. They may impede a fast rate of convergence, because for problems with an ill-conditioned Hessian, it may be necessary to produce search directions that are almost orthogonal to the gradient, and an inappropriate choice of the parameter δ may cause such steps to be rejected.
2. Angle tests destroy the invariance properties of quasi-Newton methods.

Trade of between rapid convergence and global convergence

Algorithmic strategies that achieve rapid convergence can sometimes conflict with the requirements of global convergence, and vice versa. For example, the *steepest descent* method is the quintessential globally convergent algorithm, but it is quite slow in practice. On the other hand, the *pure Newton iteration* converges rapidly when started close enough to a solution, but its steps may not even be descent directions away from the solution. The challenge is to design algorithms that incorporate both properties: good global convergence guarantees and a rapid rate of convergence.

Convergence Rate of Steepest Descent

We can learn much about the steepest descent method by considering the ideal case, in which the objective function is quadratic and the line searches are exact. Let us suppose that:

$$f(x) = \frac{1}{2}x^T Q x - bx$$

where Q is symmetric and positive definite. The gradient is given by $\nabla f(x) = Qx - b$, and the minimizer x^* is the unique solution of the linear system $Qx = b$. Then

$$\begin{aligned}\phi(\alpha) &= f(x_k - \alpha \nabla f_k) \\ &= \frac{1}{2}(x_k - \alpha \nabla f_k)^T Q(x_k - \alpha \nabla f_k) - b^T(x_k - \alpha \nabla f_k) \\ \phi'(\alpha) &= -\nabla f_k^T Q(x_k - \alpha \nabla f_k) + b^T \nabla f_k \\ \Rightarrow \alpha_k &= \frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T Q \nabla f_k}\end{aligned}$$

The steepest descent iteration is given by

$$x_{k+1} = x_k - \left(\frac{\nabla f_k^T \nabla f_k}{\nabla f_k^T Q \nabla f_k} \right) \nabla f_k$$

Since $\nabla f_k = Q x_k - b$, this equation yields a closed-form expression for x_{k+1} in terms of x_k.
To quantify the rate of convergence we introduce the weighted norm $\|x\|_Q^2 = x^T Q x$. By using the relation $Q x^* = b$, we can show that

$$\frac{1}{2}\|x - x^*\|_Q^2 = f(x) - f(x^*)$$

so this norm measures the difference between the current objective value and the optimal value. By using the iteration equality and noting that $\nabla f = Q(x_k - x^*)$, we can derive

$$\|x_{k+1} - x^*\|_Q^2 = \left\{ 1 - \frac{(\nabla f_k^T \nabla f_k)^2}{(\nabla f_k^T Q \nabla f_k)(\nabla f_k^T Q^{-1} \nabla f_k)} \right\} \|x_k - x^*\|_Q^2$$

(see Exercise 3.7). This expression describes the exact decrease in f at each iteration, but since the term inside the brackets is difficult to interpret, it is more useful to bound it in terms of the condition number of the problem.

Theorem 3.3

When the steepest descent method with exact line searches is applied to the strongly convex quadratic function, the error norm satisfies

$$\|x_{k+1} - x^*\|_Q^2 \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1} \right) \|x_k - x^*\|_Q^2$$

where $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the e-value of Q.
The proof of this result is given by *Luenberger*. The inequalities show that the function values f_k converge to the minimum f^* at a linear rate. Note there is a special case when $\lambda_1 = \lambda_n$, convergence is achieved in one iteration.

The rate-of-convergence behavior of the steepest descent method is essentially the same on general nonlinear objective functions. In the following result we assume that the step length is the global minimizer along the search direction.

Theorem 3.4

Suppose that $f : R^n \to R$ is twice continuously differentiable, and that the iterates generated by the steepest-descent method with exact line searches converge to a point x^* at which the Hessian matrix $\nabla^2 f(x^*)$ is positive definite. Let r be any scalar satisfying

$$r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}, 1\right)$$

where $0 \leq \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the e-value of $\nabla^2 f(x^*)$. Then for all k sufficiently large, we have

$$f(x_{k+1}) - f(x^*) \leq r^2[f_k - f^*]$$

In general, we cannot expect the rate of convergence to improve if an inexact line search is used. Therefore, Theorem 3.4 shows that the steepest descent method can have an unacceptably slow rate of convergence, even when the Hessian is reasonably well conditioned. Specifically, if $\kappa(Q) = 800, f(x_0) = 1, f(x^*) = 0$, Thm.3.4. suggests that the function value will still be about 0.08 after 500 iterations and 0.006 after 1000 iterations of the steepest descent method with exact line search.

Newron's Method

We now consider the Newton iteration, for which the search is given by

$$p_k^N = -\nabla^2 f_k^{-1} \nabla f_k$$

Since the Hessian matrix $\nabla^2 f_k$ may not always be positive definite, p_k^N may not always be a descent direction, and many of the ideas discussed so far in this chapter no longer apply. Later we will describe two approaches for obtaining a globally convergent iteration based on the Newton step.

Here we discuss just the *local* rate-of-convergence properties of Newton's method. We know that for all x in the vicinity of a solution point x^* such that $\nabla^2 f^*$ is positive definite, the Hessian $\nabla^2 f_k$ will also be positive definite. Newton's method will be well defined in this region and will converge quadratically, provided that the step lengths α_k are eventually always 1.

Theorme 3.5

Suppose that f is twice differentiable and that the Hessian $\nabla^2 f$ is Lipschitz continuous in a neighborhood of a solution x^* at which the Second-Order sufficient conditions are satisfied. Consider the Newton iteration $x_{k+1} = x_k + p_k$, where $p_k = -\nabla^2 f_k^{-1} \nabla f_k$. Then

1. If the starting point x_0 is sufficiently close to x^*, the sequence of iterates converges to x^*;
2. The rate of convergence of $\{x_k\}$ is quadratic;
3. The sequence of gradient norms $\{\|\nabla f_k\|\}$ converges quadratically to zero.

As the iterates generated by Newton's method approach the solution, the Wolfe (or Goldstein) conditions will accept the step length $\alpha = 1$ for all large k. This observation follows from Theorem 3.6 below.
Implementations of Newton's method using these line search conditions, and in which the line search always tries the unit step length first, will set $\alpha = 1$ for all large k and attain a local quadratic rate of convergence.

Quasi-Newton Methods

Suppose now that the search direction has the form

$$p_k = -B_k^{-1}\nabla f_k$$

where the symmetric and positive definite matrix B_k is updated at every iteration by a quasi-Newton updating formula. We assume here that the step length α_k is computed by an inexact line search that satisfies the Wolfe or strong Wolfe conditions, with the same proviso mentioned above for Newton's method: The line search algorithm will always try the step length $\alpha = 1$ first,and will accept this value if it satisfies the Wolfe conditions.
The following result shows that if the search direction of a quasi-Newton method approximates the Newton direction well enough, then the unit step length will satisfy the Wolfe conditions as the iterates converge to the solution.
It also specifies a condition that the search direction must satisfy in order to give rise to a superlinearly convergent iteration. To bring out the full generality of this result, we state it first in terms of a general descent iteration, and then examine its consequences for quasi-Newton and Newton methods.

Theorem 3.6

Suppose that f is twice differentiable. Consider the iteration $x_{k+1} = x_k + \alpha_k p_k$, where p_k is descent direction and α_k satisfies the Wolfe conditions with $c_1 < 1/2$. If the sequence $\{x_k\}$ converges at a point x^* s.t. $\nabla f^* = 0$ and $\nabla^2 f^*$ is p.d. and if the search direction satisfies

$$\lim_{k\to\infty} \frac{\|\nabla f_k + \nabla^2 f_k p_k\|}{\|p_k\|} = 0$$

then

1. the step length $\alpha = 1$ is admissible for all k greater than a centain index k_0
2. if $\alpha_k = 1$ for all $k > k_0$, $\{x_k\}$ converges to x^* superlinearly.

It is easy to see that if $c_1 > 1/2$, then the line search would exclude the minimizer of a quadratic, and unit step lengths may not be admissible.

If p_k is aquasi-Newton search direction($p_k = -B_k^{-1}\nabla f_k$), then the former inequality is equivalent to the below condition denoted by (3.36)

$$\lim_{k\to\infty} \frac{\|(B_k - \nabla^2 f(x^*))p_k\|}{\|p_k\|} = 0$$

Hence, we have the surprising (and delightful) result that a superlinear convergence rate can be attained even if the sequence of quasi-Newton matrices B_k does not converge to $\nabla^2 f(x^*)$; it suffices that the B_k become increasingly accurate approximations to $\nabla^2 f(x^*)$ *along the search directions* p_k. Importantly, this condition is **both necessary and sufficient** for the superlinear convergence of quasi-Newton methods.

Theorem 3.7

Suppose that f is twice differentiable. Consider the iteration $x_{k+1} = x_k + \alpha_k p_k$, where $p_k = -B_k^{-1}\nabla f_k$. Let us assume also that $\{x_k\}$ converges to a point x^* s.t. $\nabla^2 f(x^*) = 0$ and $\nabla^2 f(x^*)$ is p.d.. Then $\{x^*\}$ converges superlinearly if and only if the (3.36) holds.

Proof

(3.36) is equivalent to

$$p_k - p_k^N = o(\|p_k\|)$$

Then we have

$$\|x_k + p_k - x^*\| \le O(\|x_k - x^*\|^2) + o(\|p_k\|)$$

A simple manipulation of this inequality reveals that $\|p_k\| = O(\|x_k - x^*\|)$, so we obtain

$$\|x_k + p_k - x^*\| \le o(\|x_k - x^*\|)$$

giving the superlinear convergence result.

We will see in Ch.6 that quasi-Newton methods normally satisfy condition (3.36) and are therefore superlinearly convergent.

Newton's Method with Hessian Modification

Away from the solution, the Hessian matrix may not be positive definite, so the Newton direction p_k^N defined by (3.38):

$$\nabla^2 f(x_k) p_k^N = -\nabla f(x_k)$$

may not be a descent direction. We now describe an approach to overcome this difficulty when a direct linear algebra technique, such as Gaussian elimination, is used to solve the Newton equations (3.38).

This approach obtains the step p_k from a linear system identical to (3.38), except that the coefficient matrix is replaced with a positive definite approximation, formed before or during the solution process. The modified Hessian is obtained by adding either a positive diagonal matrix or a full matrix to the true Hessian. A general description of this method follows.

Algorithm3.2: Line Search Newton with Modification

```
Given initial point x0;
for k = 0,1,2,...
    Factorize the matrix Bk = H + Ek, where
        if H is p.d. Ek = 0
        else Ek is sufficiently large to ensure Bk is p.d.
    Solve Bk pk = -df(xk)
    set x(k+1) <- xk + alphak*pk, where alphak satisfies the Wolfe,
        Goldstein, or Armijo backtracking conditions;
end
```

Some approaches do not compute E_k explicitly, but rather introduce extra steps and tests into standard factorization procedures, modifying these procedures "on the fly" so that the computed factors are the factors of a positive definite matrix. Strategies based on modifying a Cholesky factorization and on modifying a symmetric indefinite factorization of the Hessian are described in this section.

Algorithm 3.2 is a practical Newton method that can be applied from any starting point. We can establish fairly satisfactory global convergence results for it, provided that the strategy for choosing E_k (and hence B_k) satisfies the **bounded modified factorization**. This property is that the matrices in the sequence B_k have bounded condition number whenever the sequence of Hessians $\nabla^2 f(x_k)$ is bounded; that is (3.39)

$$\kappa(B_k) = \|B_k\|\|B_k^{-1}\| \leq C, \quad \text{some } C > 0, \forall k = 0, 1, 2, ...$$

If this property holds, global convergence of the modified line search Newton method follows from the former result.

Theorem 3.8

Suppose that f is twice differentiable on an open set $\mathcal{D}$, and assume that the starting point x_0 of Algorithm 3.2 is such that the level set $\mathcal{L} = \{x \in \mathcal{D} : f(x) < f(x_0)\}$ is compact. Then if the bounded modified factorization property holds, we have that

$$\lim_{k \to \infty} \nabla f(x_k) = 0$$

We now consider the convergence rate of Algorithm 3.2. Suppose that the sequence $\{x_k\}$ converges to a point x^* where $\nabla^2 f(x^*)$ is sufficiently positive definite in the sense that the modification strategies described in the next section return the modification $E_k = 0$ for all sufficiently large k. By Theorem 3.6, we have that $\alpha_k = 1$ for all sufficiently large k, so that Algorithm 3.2 reduces to a pure Newton method and the rate of convergence is quadratic.

For problems in which $\nabla^2 f^*$ is close to singular, there is no guarantee that the modification E_k will eventually vanish, and the convergence rate may be only linear. Besides requiring the modified matrix B_k to be well conditioned (so that Theorem 3.8 holds), we would like the modification to be as small as possible, so that the second-order information in the Hessian is preserved as far as possible. Naturally, we would also like the modified factorization to be computable at moderate cost.

To set the stage for the matrix factorization techniques that will be used in Algorithm 3.2, we will begin by assuming that the eigenvalue decomposition of $\nabla^2 f_k$ is available.This is **not realistic** for large-scale problems because this decomposition is generally too expensive to compute, but it will *motivate* several practical modification strategies.

Eigenvalue Modification

To improve the indefinite Hessian matrix, there is a great deal of freedom in devising modification strategies, and there is currently no agreement on which strategy is best.

Actually different criterion can lead to different strategy. If A is a symmetric matrix with spectral decomposition $A = Q\Lambda Q^T$ then the correction matrix ΔA of minimum *Frobenius norm* that ensures that $\lambda_{\min}(A + \Delta A) > \delta$ is given by

$$\Delta A = Q\mathrm{diag}(\tau_{\mathrm{i}})\mathrm{Q}^{\mathrm{T}}, \quad \text{with } \tau_{\mathrm{i}} = \begin{cases} 0, & \lambda_{\mathrm{i}} \geq \delta \\ \delta - \lambda_{\mathrm{i}}, & \lambda_{\mathrm{i}} < \delta \end{cases}$$

where $\|A\|_F^2 = \sum_{i,j=1}^{n} a_{ij}^2$

Suppose again that A is a symmetric matrix with spectral decomposition $A = Q\Lambda Q^T$ then the correction matrix ΔA of minimum *Euclidean norm* that ensures that $\lambda_{\min}(A + \Delta A) > \delta$ is given by (3.44)

$$\Delta A = \tau I, \quad \text{with}\tau = \max(0, \delta - \lambda_{\min}(A))$$

These results suggest that both diagonal and nondiagonal modifications can be considered.
Various practical diagonal and nondiagonal modifications have been proposed and implemented in software.They *do not* make use of the spectral decomposition of the Hessian, since it is generally too expensive to compute. Instead, they use Gaussian elimination, choosing the modifications indirectly and hoping that somehow they will produce good steps.
Numerical experience indicates that the strategies described next often (but not always) produce good search directions.

Adding a Multiple of the Identity

Perhaps the simplest idea is to find a scalar $\tau > 0$ s.t. $H + \tau I$ is p.d. From the previous discussion we know that τ must satisfy (3.44), but a good estimate of the smallest eigenvalue of the Hessian is normally not available. The following algorithm describes a method that tries successively larger values of τ.

Algorithm 3.3 Cholesky with Added Multiple of the Identity

```
choose beta>0;
if min(a(i,i))>0
    set tau(0) <- 0;
else
    tau0 = -min(a(i,i)) + beta;
end
for k = 0,1,2,...
    try do Cholesky algorithm to obtain LL' = A + tau(k)I
    if succeessful
        return L;
    else
        tau(k+1) <- max(2tau(k),beta)
    end
end
```

The choice of β is heuristic; a typical value is 10^{-3}. We could choose the first nonzero shift τ_0 to be proportional to be the final value of τ used in the latest Hessian. The strategy implemented in Algorithm 3.3 is quite simple and may be preferable to the modified factorization techniques described next, but it suffers from one drawback. Every value of τ_k requires a new factorization of $A + \tau_k I$, and the algorithm can be quite expensive if several trial values are generated. Therefore it may be advantageous to increase τ more rapidly, say by a factor of 10 instead of 2 in the last *else* clause.

Modified Cholesky Factorization

Another approach for modifying a Hessian matrix that is not positive definite is to perform a Cholesky factorization of $\nabla^2 f(x_k)$, but to increase the diagonal elements
encountered during the factorization (where necessary) to ensure that they are sufficiently positive. This **modified Cholesky approach** is designed to accomplish two goals:

1. It guarantees that the modified Cholesky factors exist and are bounded relative to the norm of the actual Hessian
2. It does not modify the Hessian if it is sufficiently positive definite.

We Firstly briefly review the Cholesky factorrization:
Every symmetric positive definite matrix A can be written as (3.46)

$$A = LDL^T$$

where L is a lower triangular matrix with unit diagonal elements and D is a diagonal matrix with positive elements on the diagonal. By equating the elements in (3.46), *column by column*, it is easy to derive formulas for computing L and D.

Algorithm 3.4 Cholesky Factorization LDL^T Form

```
for j = 1,2,...,n
    c(j,j) <- a(j,j) - sum(@s,1,j-1,d(s)*l(j,s)^2);
    d(j) <- c(j,j);
    for i = j+1,...,n
        c(i,j) <- a(i,j) - sum(@s,1,j-1,d(s)*l(i,s)*l(j,s));
        l(i,j) <- c(i,j)/d(j);
    end
end
```

One can show that the diagonal elements are positive whenever A is p.d.. The scalars c_{ij} have been introduced only to facilitate the description of the modified factorization discussed below. We should note that Algorithm 3.4 differs a little from the standard form of the Cholesky factorization, which produces a lower triangular matrix M such that (3.47)

$$A = MM^T$$

In fact, $M = LD^{1/2}$.

If A is indefinite, the factorization $A = LDL^T$ may not exist. Even if it does exist, Algorithm 3.4 is numerically unstable when applied to such matrices, in the sense that the elements of L and D can become arbitrarily large. It follows that a strategy of computing the LDL^T factorization and then modifying the diagonal after the fact to force its elements to be positive may break down, or may result in a matrix that is drastically different from A.

Instead, we can modify the matrix A during the course of the factorization in such a way that all elements in D are sufficiently positive, and so that the elements of D and L are not too large. To control the quality of the modification, we choose two positive parameters δ and β, and require that during the computation of the jth columns of L and D in Algorithm 3.4 (that is, for each j in the outer loop of the algorithm) the following bounds be satisfied:(3.48)

$$d_j \geq \delta, \quad |m_{ij}| \leq \beta, \quad i = j+1, ..., n,$$

where $m_{ij} = l_{ij}\sqrt{d_j}$. To satisfy these bounds we only need to change one step in Algorithm 3.4: The formula for computing the diagonal element d_j in Algorithm 3.4 is replaced by

$$d_j = \max\left(|c_{jj}|, \left(\frac{\theta_j}{\beta}\right)^2, \delta\right), \quad \text{with } \theta_j = \max_{j<i\leq n} |c_{ij}|.$$

And the (3.48) is automatically satisfied.
Note that θ_j can be computed prior to d_j, and that is the reason we introduce c_{ij} into the algorithm.
If P denotes the permutation matrix associated with the row and column interchanges, the algorithm produces the Cholesky factorization of the permuted, modified matrix $PAP^T + E$, that is,(3.50)

$$PAP^T + E = LDL^T = MM^T$$

One can show that the matrices B_k obtained by this modified Cholesky algorithm to the exact Hessians have bounded condition numbers, that is, the bound (3.39) holds for some value of C.

Modified Symmetric Indefinite Factorization

Another strategy for modifying an indefinite Hessian is to use a procedure based on a symmetric indefinite factorization. Any symmetric matrix A, whether positive definite or not, can be written as(3.51)

$$PAP^T = LBL^T$$

where L is unit lower triangular, B is a block diagonal matrix with blocks of dimension 1 or 2, and P is a permutation matrix.
By using the block diagonal matrix B, we can guarantee that the factorization (3.51) always exists and can be computed by a numerically stable process.

As for the Cholesky factorization, an indefinite symmetric factorization algorithm can be modified to ensure that the modified factors are the factors of a positive definite matrix. The strategy is first to compute the factorization (3.51), as well as the spectral decomposition $B = Q\Lambda Q^T$, which is inexpnsive to compute because B is block diagonal. We then construct a modification matrix F s.t.

$$L(B + F)L^T$$

is sufficiently positive definite. Motivated by the modified spectral decomposition (3.43), we choose a parameter $\delta > 0$ and define F to be (3.53)

$$F = Q\mathrm{diag}(\tau_i)Q^T, \quad \tau_i = \begin{cases} 0, & \lambda_i \geq \delta \\ \delta - \lambda_i, & \lambda_i < \delta \end{cases}$$

where λ_i are the eigenvalues of B. The matrix F is thus the modification of minimum Frobenius norm that ensures that all eigenvalues of the modified matrix $B + F$ are no less than δ. This strategy therefore modifies the factorization (3.51) as follows:

$$P(A + E)P^T = L(B + F)L^T, \quad \text{where } E = P^T LFL^T P$$

Note that E will not be diagonal here, in general. Hence, in contrast to the modified Cholesky approach, this modification strategy changes the entire matrix A, not just its diagonal.
The aim of (3.53) is that the modified matrix satisfies $\lambda_{\min}(A+E) \approx \delta$ whenever the orign A has $\lambda_{\min} < \delta$. It is not clear, however, whether it always comes close to attaining this goal.

Step-Length Selection Algorithms

We now consider techniques for finding a minimum of the one-dimensional function(3.54)

$$\phi(\alpha) = f(x_k + \alpha p_k)$$

or for simply finding a step length α_k satisfying one of the termination conditions described before. We assume that p_k is a descent direction, that is $\phi'(0) < 0$, so that our search can be confined to positive values of α.
If f is a convex quadratic, $f(x) = \frac{1}{2} x Q x^T - b^T x$, we have

$$\alpha_k = -\frac{\nabla f_k^T p_k}{p_k^T Q p_k}$$

For general nonlinear functions, it is necessary to use an iterative procedure. The line search procedure deserves particular attention because it has a major impact on the robustness and efficiency of all nonlinear optimization methods.
This section, we discuss only algorithms that make use of derivative information.

All line search procedures require an initial estimate α_0 and generate a sequence $\{\alpha_k\}$ that either terminates with a step length satisfying the conditions specified by the user or determines that such a step length does not exist.
Typical procedures consist of two phases:

1. A bracketing phase that finds an interval $[\bar{a}, \bar{b}]$ containing acceptable step lengths
2. A selection phase that zooms in to locate the final step length.

The selection phase usually reduces the bracketing interval during its search for the desired step length and interpolates some of the function and derivative information gathered on earlier steps to guess the location of the minimizer. We first discuss how to perform this interpolation.

In the following discussion we let α_k and α_{k-1} denote the step lengths used at iterations k and $k-1$ of the optimization algorithm, respectively. On the other hand, we denote the trial step lengths generated during the line search by α_i and α_{i-1} and also α_j. We use α_0 to denote the initial guess.

Interpolation

We begin by describing a line search procedure based on interpolation of known function and derivative values of the function ϕ. This procedure can be viewed as an enhancement of Algorithm

3.1. The aim is to find a value of α that satisfies the sufficient condition, without being "too small".

Note that we can write the sufficient decrease condition in the notation of (3.54) as (3.56):

$$\phi(\alpha_k) \leq \phi(0) + c_1 \alpha_k \phi'(0)$$

and taht since the constant c_1 is usually chosen to be small in practice($c_1 = 10^{-4}$, say), this condition asks for little more than descent in f. We design the procedure to be “efficient” in the sense that it computes the derivative ∇f as few times as possible.

Suppose that the initial guess α_0 is given.
If this step length satisfies the condition, and we terminate the search.
Otherwise, we know that the interval $[0, \alpha_0]$ contains acceptable step lengths. We form a *quadratic* $\phi_q(\alpha)$ to ϕ by interpolating the three pieces of information available—$\phi(0), \phi'(0), \phi(\alpha_0)$, and take the minimizer as α_1 (3.58):

$$\alpha_1 = -\frac{\phi'(0)\alpha_0^2}{2[\phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0]}$$

If α_1 satisfied the condition, we terminate the search.
Otherwise, we construct a *cubic* function ϕ_c that interpolates the four pieces of information $\phi(0), \phi'(0), \phi(\alpha_0), \phi(\alpha_1)$, finding that the minimizer α_2 of ϕ_c lies in $[0, \alpha_1]$, and is given by

$$\alpha_2 = \frac{-b + \sqrt{b^2 - 3a\phi'(0)}}{3a}$$

If necessary, the process is repeated, using $\phi(0), \phi'(0)$ and two recent value, until an α satisfied is located. If any α_i is too close to α_{i-1}, we will take $\alpha_i = \alpha_{i-1}/2$. This safeguard procedure ensures that we make reasonable progress on each iteration and that the final α is not too small.
The strategy just described assumes that derivative values are significantly more expensive to compute than function values. Accordingly, we can design an alternative strategy based on cubic interpolation of the values of ϕ and ϕ' at the two most recent values of α. The decision on which of α_{i-1} and α_i should be kept and which discarded depends on the specific conditions used to terminate the line search.

Initial Step Length

For Newton and quasi-Newton methods, the step $\alpha_0 = 1$ should always be used as the initial trial step length. This choice ensures that unit step lengths are taken whenever they satisfy the termination conditions and allows the rapid rate-of-convergence properties of these methods to take effect.

For methods that do not produce well scaled search directions,such as the steepestdescent and conjugate gradient methods, it is important to use current information about the problem and the algorithm to make the initial guess.

A popular strategy is to assume that the first-order change in the function at iterate x_k will be the same as that obtained at the previous step. In other words, we choose the initial guess α_0 s.t. $\alpha_0 \nabla f_k^T p_k = \alpha_{k-1} \nabla f_{k-1}^T p_{k-1}$, that is

$$\alpha_0 = \alpha_{k-1} \frac{\nabla f_{k-1}^T p_{k-1}}{\nabla f_k^T p_k}$$

Another useful strategy is to interpolate a quadratic to the data $f(x_{k-1}), f(x_k), \nabla f_{k-1}^T p_{k-1}$ and to define α_0 to be its minimizer. This strategy yields (3.60)

$$\alpha_0 = \frac{2(f_k - f_{k-1})}{\phi'(0)}$$

It can be shown that if $x_k \to x^*$ superlinearly, then the ratio in this expression converges to 1. If we adjust the choice (3.60) by setting

$$\alpha_0 \leftarrow \min(1, 1.01\alpha_0)$$

we find that the unit step length $\alpha_0 = 1$ 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 procedure that is guaranteed to find a step length satisfying the strong Wolfe conditions(3.7) for any parameters c_1, c_2 satisfying $0 < c_1 < c_2$. As before, we assume that p is a descent direction and that f is bounded below along the direction p.

The algorithm has two stages. This first stage begins with a trial estimate α_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.

Algorithm 3.5 (Line Search Algorithm).

```
Set alpha0 <- 0, choose alpha_m>0 and alpha(1) in (0,alpha_m);
i <- 1;
repeat
    Evaluate phi(alpha(i));
    if phi(alpha(i)) > phi(0) + c1*alpha(i)*phi'(0)
        or [phi(alpha(i)) >= phi(alphi(i-1)) and i>1]
        alpha* <- zoom(alpha(i-1),alpha(i)) and stop;
    Evaluate phi'(alpha(i));
    if |phi'(alpha(i))| <= -c2*phi'(0)
        alpha* <- alphi(i) and stop;
    if phi'(alphi(i)) >= 0
        alpha* <- zoom(alpha(i-1),alpha(i)) and stop;
    Choose alpha(i+1) in (alphi(i),alphi_m);
    i <- i+1;
end
```

The procedure uses the knowledge that the interval (α_{i-1}, α_i) contains step lengths satisfying the strong Wolfe conditions if one of the following three conditions is satisfied:

1. α_i violates the sufficient decrease condition;
2. $\phi(\alpha_i) \geq \phi(\alpha_{i-1})$;
3. $\phi'(\alpha_i) \geq 0$.

The last step of the algorithm performs extrapolation to find the next trial value α_i. To implement this step we can use approaches like the interpolation procedures above or other method. Whichever strategy we use, it is important that the successive steps increase quickly enough to reach the upper limit $\alpha_{\max}$ in a finite number of iterations.

We now specify the function zoom, which requires a little explanation. The order of its input arguments is such that each call has the form **zoom(α_l, α_h)**, where

1. the interval bounded by α_l and α_h contains step lengths that satisfy the strong Wolfe conditions;
2. α_l is, among all step lengths generated so far and satisfying the sufficient decrease condition, the one giving the smallest function value;
3. α_h is chosen so that $\phi'(\alpha_l)(\alpha_h - \alpha_l) < 0$

Algorithm 3.6 (zoom).

```
repeat
    Interpolate (using quadratic, cubic, or bisection) to find
        a trial step length alpha(j) between alpha_l and alpha_h;
    Evaluate phi(alpha(j));
    if phi(alpha(j)) > phi(0) + c1*alpha(j)*phi'(0) or phi(alpha(j)) >= phi(alpha_l)
        alpha_h <- alpha(j);
    else
        Evaluate phi'(alpha(j));
        if |phi'(alpha(j))| <= -c2*phi(0)
            alpha* <- alpha(j) and stop
        if phi'(alpha(j))(alpha_h - alpha_l) >= 0
            alpha_h <- alpha_l;
        alpha_l <- alpha(j)
end
```

If the new estimate α_j happens to satisfy the strong Wolfe conditions, then zoom has served its purpose of identifying such a point, so it terminates with $\alpha_* = \alpha_j$. Otherwise, if α_j satisfies the sufficient decrease condition and has a lower function value than x_l, then we set $\alpha_l < -\alpha_j$ to maintain condition (b). If this setting results in a violation of condition (c), we remedy the situation by setting α_h to the old value of α_l.

The strong Wolfe conditions have the advantage that by decreasing c_2 we can directly control the quality of the search, by forcing the accepted value of α to lie closer to a local minimum.