Adaptations of Newton's Method

UBC Math 604 Lecture Notes by Philip D. Loewen

Preamble. The quadratic convergence of Newton's method for minimization is guaranteed only for initial points sufficiently near a critical point satisfying certain hypotheses. For for starting points far from the minimizer, or in the absence of these conditions, Newton's method may stall (as on $f(x) = |x|^{3/2}$) or even diverge (as on $f(x) = x\tan^{-1}x - \frac{1}{2}\ln(1+x^2)$). (Note that both these functions of one variable are strictly convex, with unique global minimizers at the origin). So we work toward a hybrid method that combines

- guaranteed progress [perhaps slow] from arbitrary points;
- rapid, Newton-like convergence from points sufficiently near a well-conditioned minimizer.

The iteration framework runs like this, from x_k:

1. Decide if x_k is an acceptable approximation to the desired minimizer. If so, quit; otherwise, continue with Step 2.
2. Use current information to evaluate H_k, a *positive-definite* symmetric matrix to use in the "model"
$$m_k(x) = f(x_k) + \nabla f(x_k)(x - x_k) + \tfrac{1}{2}(x - x_k)^T H_k (x - x_k).$$
(Classic Newton's Method just uses $H_k = D^2 f(x_k)$, but $H_k > 0$ must be *assumed*. Taking $H_k = I$ for all k gives the famous Steepest Descent Method, discussed below.)
3. Minimize $m_k(x_k + v)$ over all v analyticcally: the solution v_k is characterized by
$$H_k v_k = -\nabla f(x_k)^T. \tag{1}$$
Call v_k the "search direction".
4. Step to the new point
$$x_{k+1} = x_k + t_k v_k$$
for some well-chosen $t_k > 0$ (the "step length").

A. Descent Directions

Descent Directions. "Progress" in minimizing f from a starting point x_k might be defined as a decrease in function value at the next step:
$$f(x_{k+1}) < f(x_k).$$
For a given vector v, the first-order approximation
$$f(x_k + tv) \approx f(x_k) + \nabla f(x_k)\,(tv)\,, \qquad t \to 0$$
indicates that $f(x_k + tv) < f(x_k)$ for all $t > 0$ sufficiently small will guaranteed if
$$f'(x; v) = \nabla f(x_k) v < 0.$$
Any v satisfying this condition is called *a (first-order) descent direction for f at x_k.*

Note. Insisting that the v_k determined by (1) should be a descent direction leaves plenty of choice for the matrix H_k we must build in Step 2. In fact, for any $H_k = H_k^T > 0$, the direction v_k defined in (1) will be a descent direction, since

$$\nabla f(x_k)v_k = \nabla f(x_k)\left[-H_k^{-1}\nabla f(x_k)^T\right] = -\nabla f(x_k)H_k^{-1}\nabla f(x_k)^T < 0. \qquad (*)$$

(Recall that $H_k > 0$ iff $H_k^{-1} > 0$.) Different schemes for producing H_k include ...

(i) Always choose $H_k = I$. This is of some theoretical interest, because it gives the famous "Method of Steepest Descent." This method, sketched below, will converge to a critical point of f under mild hypothesis, but the convergence can be painfully slow.

(ii) Decide beforehand on some constant $m > 0$. Compute the Hessian $D^2f(x_k)$ exactly. Then use $H_k = D^2f(x_k)+\mu_k I$, where $\mu_k \geq 0$ is chosen to guarantee that $H_k - mI$ is positive definite. Here $m > 0$ is a small offset that provides a uniform margin of nonsingularity for all the matrices H_k. For iterates x_k near a critical point $\widehat{x}$ for which $D^2f(\widehat{x}) > mI$, the choices $\mu_k = 0$ will eventually start to work, and the iterates will be constructed according to the classical Newton scheme.

(iii) Find a systematic way to *approximate* $D^2f(x_k)$ using only values of ∇f observed in earlier steps. We will study this prospect in detail in a later section.

B. Inexact Line Search

At current point x_k, suppose the vector $v_k \neq 0$ is known to be a descent direction. How far in this direction should we move before settling on x_{k+1}? To help decide, consider the scalar function

$$\phi_k(t) = f(x_k + tv_k) - f(x_k), \quad t \in \mathbb{R}.$$

Note that $\phi_k(0) = 0$, $\phi_k'(0) < 0$. In *exact line search,* we choose t_k to minimize ϕ over $(0, +\infty)$. This is useful in some contexts, but for Newton-based methods, it is more efficient to do something much less demanding.

Armijo's Rule. Fix a parameter c_1 in $(0, 1)$ [typically $c_1 = 10^{-4}$ for Newton-type methods] and consider this inequality:

$$\phi(t) \leq c_1 t\phi'(0). \qquad (D)$$

The t-values satisfying (D) are those whose function values lie below a straight line that slopes downward more gently than the function ϕ does at the origin. A descriptive name for (D) is the "sufficient decrease condition," also known as **Armijo's rule**. Notice that larger t-values in $[0, 1]$ are preferable, because they correspond to greater predicted decreases in ϕ.

Picture Here

There are two common implementations of Armijo's rule, both involving parameters λ in $(0,1)$ and k^* in $\mathbb{Z}$. These conspire to produce the set in which we will look for t:

$$\Lambda = \left\{\lambda^k \,:\, k \geq k^*\right\}.$$

Notice that the numbers λ^k decrease to 0 as $k \to \infty$, so

(i) smaller k-values in Λ correspond to preferable choices of t in (D);

(ii) Inequality (D) is true for $t = \lambda^k$ for all k sufficiently large, by definition of the derivative.

One simple way to get a good t-value, then, is just to test $t = \lambda^k$ in (D) for successive values of $k = k^*, k^*+1, \ldots$ until you find an exponent that works. This is easy to program and guaranteed to terminate with the largest eligible t in Λ after a finite number of steps. In practice, though, it may require too many evaluations of ϕ.

A less computationally demanding modification of the basic approach is to settle for any point t in Λ where (D) holds, but for which the next larger choice violates (D). That is, we seek a point $t = \lambda^k$ such that both

(A) $\phi\left(\lambda^k\right) \leq c_1\lambda^k\phi'(0)$,

(B) Either $k = k^*$ or $\phi\left(\lambda^{k-1}\right) > c_1\lambda^{k-1}\phi'(0)$.

With this stipulation, it is not necessary to search the set Λ starting with the largest element. (If you do, you will end up with the same k-value generated in the previous paragraph, produced for exactly the same amount of computing effort.) Instead, Polak suggests the following procedure.* Start with some trial exponent $k_{\text{trial}} \geq k^*$: set $k = k_{\text{trial}}$, and then

```
Test (A) and (B):
  If both are true, then select t = λ^k.
  If (A) is false, move to the left,
     i.e., replace k with k + 1 and repeat the test.
  If (A) is true but (B) is false, move to the right,
     i.e., replace k with k − 1 and repeat the test.
```

The origins of k_{trial} are somewhat ad-hoc: in a situation where repeated applications of Armijo's step-size rule take place, Polak suggests using $k_{\text{trial}} = k^*$ on the very first try, and subsequently setting k_{trial} to the k-value used successfully on the previous step. (He cautions that if this produces a run of uncomfortably large k-values, it may be wise to restart with $k_{\text{trial}} = k^*$ at some intermediate time.)

In original variables, the sufficient decrease condition central to Armijo's rule

* This is not quite what Polak (Subprocedure 1.2.23a) says: his tests do not prescribe an action when both (A) and (B) are false. The modification above evidently makes sense, but is there something better one could do?

takes the form

$$f(x_k + tv_k) - f(x_k) \leq c_1 t \nabla f(x_k) v_k. \qquad (**)$$

Lemma. *Let $c_1 \in (0, 1/2)$ be fixed. If $f: \mathbb{R}^n \to \mathbb{R}$ has a critical point at $\widehat{x}$ and satisfies the conditions of the convergence theorem for Newton's method, then there is an open ball centred at $\widehat{x}$ in which the Newton Direction v_k satisfies $(**)$ with $t = 1$.*

Proof. (Sketch.) Pick any x_k in the ball identified in the cited convergence theorem. Use H_k as shorthand for $D^2 f(x_k)$, and recall the Newton step

$$v_k = -H_k^{-1} \nabla f(x_k)^T.$$

Then

$$\begin{aligned} f(x_k + v_k) &\approx f(x_k) + \nabla f(x_k) v_k + \tfrac{1}{2} v_k^T H_k v_k \\ &\approx f(x_k) + c_1 \nabla f(x_k) v_k + (1 - c_1) \nabla f(x_k) \left[-H_k^{-1} \nabla f(x_k)^T \right] \\ &\quad + \tfrac{1}{2} \nabla f(x_k) H_k^{-1} H_k H_k^{-1} \nabla f(x_k)^T \\ &\approx f(x_k) + c_1 \nabla f(x_k) v_k + (c_1 - \tfrac{1}{2}) v_k^T H_k v_k. \end{aligned}$$

Note that $c_1 - \frac{1}{2} < 0$ by hypothesis and $H_k > 0$, so the last term on the right side is negative. This proves the desired inequality. ////

The Wolfe Conditions. Given a function $\phi: [0, \infty) \to \mathbb{R}$ with $\phi(0) = 0$, $\phi'(0) < 0$, we want to choose a good step length $t_k > 0$. The Wolfe conditions help. They are set up using two user-specified constants, $0 < c_1 < c_2 < 1$.

- Armijo Rule [enforces Sufficient Decrease]:

$$\phi(t_k) \leq \phi(0) + c_1 \phi'(0) t_k. \qquad (D)$$

- Curvature Condition: Insist that $\phi'(t_k)$ has increased somewhat from its initial value $\phi'(0)$, via

$$\phi'(t_k) \geq c_2 \phi'(0). \qquad (C)$$

  Idea: If $\phi'(t)$ is still close to $\phi'(0)$, the rate of decrease at t is still quite good, so we should increase t. Note that if $\phi'(t_k) \leq 0$, this condition is equivalent to

$$|\phi'(t_k)| \leq c_2 |\phi'(0)|. \qquad (C^+)$$

  Using (C^+) even when $\phi'(t_k) \geq 0$ makes some sense: a strongly positive value of $\phi'(t_k)$ indicates that t_k falls on a steep uphill slope, and is likely too large.

Together, the sufficient decrease and curvature conditions are called the Wolfe Conditions:

Wolfe Conditions:	(D)+(C)
Strong Wolfe Conditions:	(D)+(C^+)

C. Descent Methods and Their Convergence

Steepest Descent. Using $H_k = I$ at every step of the general framework outlined earlier produces the search direction $v_k = -\nabla f(x_k)$. As shown above, this v_k points in the direction where the best linear approximation to f based at x_k predicts the most rapid decrease. Thus the classical "steepest descent method" updates the current point x_k as follows:

`Introduce` $v_k = -\nabla\ f(x_k)^T$ `, the steepest descent direction from` x_k
`Choose` $t_k > 0$ `to minimize (exactly)` $t \mapsto\ f(x_k + t\ v_k)$.
`Define` $x_{k+1} = x_k + t_k\ v_k$.

It can be very slow even on purely quadratic problems when the relative scaling of the variables is poor. (See zig-zag picture in Numerical Recipes, with associated explanation; contrast the fact that Newton's method gets the right answer in a single step on all quadratic problems.) Choosing t_k using inexact line search as described above just makes it slower.

Still, the steepest descent direction is an important benchmark for assessing the rate of decrease associated with other directions. Suppose v_k in $\mathbb{R}^n$ is a proposed search direction at the current point x_k. Let $\theta_k \in [0, \pi]$ be the (smaller) angle between v_k and $-\nabla f(x_k)$, defined by

$$-\nabla f(x_k)v_k = \|\nabla f(x_k)\|\|v_k\| \cos\theta_k, \quad \text{i.e.,} \quad \cos\theta_k = -\frac{\nabla f(x_k)v_k}{\|\nabla f(x_k)\|\|v_k\|}.$$

Here is a theoretical basis for showing some respect for the steepest descent method:

Theorem. *Given $x_0 \in \mathbb{R}^n$ and a function $f\colon \mathbb{R}^n \to \mathbb{R}$, suppose that there exist a constant $L > 0$ and an open set Ω containing $\{x \in \mathbb{R}^n : f(x) \le f(x_0)\}$, such that*

$$\|\nabla f(y) - \nabla f(x)\| \le L\|y - x\| \qquad \forall x, y \in \Omega.$$

Then for any sequences of trial points $x_k \in \mathbb{R}^n$, step lengths $t_k > 0$, and search directions $v_k \in \mathbb{R}^n$, related by

(i) *$f'(x_k; v_k) < 0$ always (each v_k is a descent direction at x_k),*

(ii) *$x_{k+1} = x_k + t_k v_k$ (each update is a step in direction v_k), and*

(iii) *each t_k obeys the Wolfe Conditions,*

one has $\sum_{k=0}^{\infty} (\|\nabla f(x_k)\| \cos\theta_k)^2 < +\infty$. In particular, $\nabla f(x_k) \cos\theta_k \to 0$ as $k \to \infty$.

Note. In the Steepest Descent method, we choose $v_k = -\nabla f(x_k)$. This obeys (i)–(ii) automatically, and $\theta_k = 0$ for all k. The conclusion reduces to $\sum_{k=0}^{\infty} (\|\nabla f(x_k)\|)^2 <$

$+\infty$, which implies $\nabla f(x_k) \to 0$ as $k \to \infty$. So although Steepest Descent is slow in practice, it is sure to produce plenty of nearly-critical points eventually. Inexact line search is just fine for this theoretical conclusion.

The desirable outcome $\|\nabla f(x_k)\| \to 0$ turns up even if all we know is that there is some constant $\varepsilon > 0$ such that $\cos\theta_k \geq \varepsilon$ for all k. In this case, the squeeze theorem gives $\varepsilon\|\nabla f(x_k)\| \to 0$. So as long as our search directions lie in some kind of rather blunt cone directed along the steepest descent direction, we can expect convergence.

Proof. Rewrite the Wolfe conditions in terms of the original function:

$$f(x_k + t_k v_k) \leq f(x_k) + c_1 \nabla f(x_k) t_k v_k \tag{D}$$
$$\nabla f(x_k + t_k v_k) v_k \geq c_2 \nabla f(x_k) v_k \tag{C}$$

By the Lipschitz condition and (C),

$$\begin{aligned} t_k L \|v_k\|^2 &\geq \|\nabla f(x_k + t_k v_k) - \nabla f(x_k)\| \|v_k\| \\ &\geq \left(\nabla f(x_{k+1}) - \nabla f(x_k)\right) v_k \\ &\geq \left(\nabla f(x_{k+1}) v_k - c_2 \nabla f(x_k) v_k\right) + (c_2 - 1)\nabla f(x_k) v_k \\ &\geq (c_2 - 1)\nabla f(x_k) v_k. \end{aligned}$$

This gives

$$t_k \geq \frac{c_2 - 1}{L} \frac{\nabla f(x_k) v_k}{\|v_k\|^2}.$$

Use this in (D): for a suitable definition of M,

$$f(x_{k+1}) \leq f(x_k) - c_1 \left(\frac{1 - c_2}{L}\right) \frac{\left(\nabla f(x_k) v_k\right)^2}{\|v_k\|^2} = f(x_k) - M \left(\nabla f(x_k) \cos\theta_k\right)^2.$$

Sum both sides from $k = 0$ to $k = N$, then cancel:

$$f(x_{N+1}) \leq f(x_0) - M \sum_{k=0}^{N} \left(\nabla f(x_k) \cos\theta_k\right)^2.$$

This gives

$$\sum_{k=0}^{N} \left(\nabla f(x_k) \cos\theta_k\right)^2 \leq \frac{f(x_0) - f(x_{N+1})}{M} \leq \frac{f(x_0) - \inf(f)}{M} \quad \text{(assumed finite)}.$$

Since RHS is indep of N, taking limit as $N \to \infty$ gives the result. ////

Globalization. The good feature of the theorem above is that it makes no requirement that x_0 should be anywhere near a true (local) minimum point $\widehat{x}$. Hence a method obeying the hypotheses outlined above will converge from any starting point. This is called "global convergence," and it's obviously a good thing. [Caution: It's not the same as finding a "global minimum."]

D. Secant Methods

Derivation. In the model Newton algorithm above, the choice of Hessian approximation H_k in Step 3 makes a big difference. In the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, we choose H_{k+1} to make sure that the quadratic model

$$m_{k+1}(x) = f(x_{k+1}) + \nabla f(x_{k+1})(x - x_{k+1}) + \tfrac{1}{2}(x - x_{k+1})^T H_{k+1}(x - x_{k+1})$$

matches the gradient of the true objective f at both x_k and x_{k+1}. Now

$$\nabla m_{k+1}(x) = \nabla f(x_{k+1}) + (x - x_{k+1})^T H_{k+1},$$

so $\nabla m_{k+1}(x_{k+1}) = \nabla f(x_{k+1})$ is easy. But forcing $\nabla m_{k+1}(x_k) = \nabla f(x_k)$ leads to

$$\nabla f(x_{k+1})^T - \nabla f(x_k)^T = H_{k+1}\,(x_{k+1} - x_k)\,. \tag{S}$$

This is the *secant condition.*

(Another derivation of (S) uses linear approximation: knowing x_k and H_k gives us a point x_{k+1} where we can calculate $\nabla f(x_{k+1})$. Linear approximation suggests

$$\nabla f(x_k) - \nabla f(x_{k+1}) \approx D^2 f(x_{k+1})\,(x_k - x_{k+1})\,.$$

Condition (S) suggests that we choose H_{k+1} to give equality here.)

Here we have n equations to determine the n^2 unknown components of H_{k+1}. Other considerations restrict our freedom somewhat:

(i) H_{k+1} should be symmetric (so only $n(n+1)/2$ elements are open for choice) and positive definite (imposing n additional inequalities, on the eigenvalues);

(ii) H_{k+1} should not be too far from H_k.

Curvature Conditions. If (S) has a positive-definite solution for H_{k+1}, multiplying on the left by $(x_{k+1} - x_k)^T$ will give

$$(x_{k+1} - x_k)^T(\nabla f_{k+1} - \nabla f_k)^T = (x_{k+1} - x_k)^T H_{k+1}(x_{k+1} - x_k) > 0. \tag{2}$$

So positivity of the left side is an absolute prerequisite for (S) to have any solution at all. Fortunately, (2) is automatic if the sequence $\ldots, x_k, x_{k+1}, \ldots$ satisfies the Wolfe Conditions. Recall that these involve $0 < c_1 < c_2 < 1$, and require both

$$\nabla f(x_{k+1})(x_{k+1} - x_k) \geq c_2 \nabla f(x_k)(x_{k+1} - x_k), \tag{C}$$

i.e.,

$$(\nabla f(x_{k+1}) - \nabla f(x_k))(x_{k+1} - x_k) \geq (c_2 - 1)\nabla f(x_k)(x_{k+1} - x_k),$$

and that $x_{k+1} - x_k$ be a descent direction for f at x_k. This makes the right side positive.

Derivation of the BFGS Update. To smooth the derivation, let's simplify the notation:

H_0: the current approximate Hessian, obeying $H_0 = H_0^T > 0$. This could come from anywhere.

s: $x_{k+1} - x_k$, the most recent step in the method.

y: $\nabla f(x_{k+1})^T - \nabla f(x_k)^T$, the change in gradient (as a column).

H: H_{k+1}, the new approximate Hessian we are to construct.

We assume $y^T s > 0$.

Eventually we will get

$$H = H_0 + \frac{1}{y^T s}\left[yy^T\right] - \frac{1}{s^T H_0 s}\left[H_0 s s^T H_0\right].$$

The assumption that $y^T s > 0$ makes H well-defined. Clearly $H = H^T$ (each summand is symmetric), and direct calculation confirms that $Hs = y$.

Here are the details. A matrix H is symmetric and positive-definite if and only if it can be decomposed as $H = J^2$ for some matrix $J = J^T > 0$. (This decomposition clearly implies that $H = H^T > 0$; given $H = H^T > 0$, orthogonal diagonalization of H is possible, and leads easily to a suitable proposal for J.) So one way to solve $Hs = y$ is to split it into a pair of linear systems, with a new variable v: we seek an invertible matrix J for which $J^2 s = y$, i.e.,

$$(3) \quad Jv = y \qquad \text{where} \qquad (4) \quad v = Js \neq 0.$$

If v were known, a simple matrix J satisfying (3) would be $\frac{1}{v^T v}(yv^T)$. More generally, one could start with any invertible matrix J_0 and use

$$\begin{aligned} J &= \frac{1}{v^T v}\left(yv^T\right) + J_0 - \frac{1}{v^T v}\left(J_0 v\right)v^T \\ &= J_0 + \frac{1}{v^T v}\left(y - J_0 v\right)v^T. \end{aligned}$$

But v is not known—it's a choice parameter that yields J according to the recipe just proposed. For a J of this form to fulfill our needs, we must have

$$\begin{aligned} v = J^T s &= \left(J_0^T + \frac{1}{v^T v} v\left(y - J_0 v\right)^T\right) s \\ &= J_0^T s + \frac{\left(y - J_0 v\right)^T s}{v^T v} v \end{aligned}$$

This can only happen if v and $J_0^T s$ are parallel, i.e., if $v = \alpha J_0^T s$ for some real α. Substitution provides a vector equation in which each term is a scalar multiple of $J_0^T s$—a vector that is nonzero because J_0 is invertible and $s \neq 0$:

$$\alpha J_0^T s = J_0^T s + \frac{\left(y - \alpha J_0 J_0^T s\right)^T s}{\alpha^2\left(s^T J_0 J_0^T s\right)}\left(\alpha J_0^T s\right).$$

Now a good choice for J_0 would be to make $H_0 = J_0 J_0^T$. Then equating coefficients of $J_0^T s$ above produces the scalar equation

$$\alpha = 1 + \frac{y^T s}{\alpha\,(s^T H_0 s)} - 1, \quad \text{i.e.,} \quad \alpha^2 = \frac{y^T s}{s^T H_0 s}.$$

The denominator here is positive because J_0 is invertible, and the numerator is positive by hypothesis. Thus this equation has two solutions for α, for either one, we can trace our steps backwards to build up J: with $v = \alpha J_0^T s$, we have

$$\begin{aligned} v^T v &= \alpha^2 \left(s^T H_0 s\right) = y^T s \\ J_0 v &= \alpha H_0 s, \\ J &= J_0 + \frac{\alpha}{y^T s}\left(y - \alpha H_0 s\right) s^T J_0 = J_0 + \frac{\alpha}{y^T s} y s^T J_0 - \frac{1}{s^T H_0 s} H_0 s s^T J_0. \end{aligned}$$

[At this point, Dennis and Schnabel choose the positive root for α, and assert that the invertibility of J is evident.] To recover an explicit formula for H, we expand $H = JJ^T$: this generates three "square terms" and three pairs of "mixed terms" as follows:

$$\begin{aligned} JJ^T = J_0 J_0^T &+ \left(\frac{\alpha}{y^T s}\right)^2 y s^T J_0 J_0^T s y^T + \left(\frac{1}{s^T H_0 s}\right)^2 H_0 s s^T J_0 J_0^T s s^T H_0 \\ &+ \left(\frac{\alpha}{y^T s}\right)\left(J_0 J_0^T s y^T + y s^T J_0 J_0^T\right) \\ &- \left(\frac{1}{s^T H_0 s}\right)\left(J_0 J_0^T s s^T H_0 + H_0 s s^T J_0 J_0^T\right) \\ &- \left(\frac{\alpha}{(y^T s)\,(s^T H_0 s)}\right)\left(H_0 s s^T J_0 J_0^T s y^T + y s^T J_0 J_0^T s s^T H_0\right) \end{aligned}$$

Recalling $H_0 = J_0 J_0^T$ and recognizing the corresponding product $s^T J_0 J_0^T s = s^T H_0 s$ as a (positive) scalar, we find that the second and fourth lines in this equation cancel, while the third line can be combined with the last term of the first. The result is

$$H = H_0 + \frac{1}{y^T s}\left[y y^T\right] - \frac{1}{s^T H_0 s}\left[H_0 s s^T H_0\right].$$

As part of the development below we will provide an explicit formula for H^{-1} in terms of H_0^{-1}; this will confirm the invertibility of H and complete the proof. ////

BFGS (Basics). The proof above contains several key results. We look first at the formula

$$H = H_0 + \frac{1}{y^T s}\left[y y^T\right] - \frac{1}{s^T H_0 s}\left[H_0 s s^T H_0\right].$$

Given any vectors s and y with $y^T s > 0$, and any symmetric, positive-definite matrix H_0, this formula generates a symmetric, positive-definite matrix H for which $Hs = y$.

In the context of a quasi-Newton iteration, using the current approximation H_k for the Hessian as the seed value produces the updated approximate Hessian

$$H_{k+1} = H_k + \frac{1}{y^T s}\left[yy^T\right] - \frac{1}{s^T H_k s}\left[H_k s s^T H_k\right],$$
$$\text{where } s = x_{k+1} - x_k, \quad y = \left(\nabla f(x_{k+1}) - \nabla f(x_k)\right)^T.$$

This is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update rule, and it is the most effective general-purpose scheme in practice.

Additional theoretical effort, based on the next lemma, can streamline the process of solving a linear system in which H_{k+1} is the coefficient matrix.

Sherman-Morrison Formula. *Given an invertible matrix A in $\mathbb{R}^{n\times n}$ and two vectors $a, b \in \mathbb{R}^n$ the sum $P = A + ab^T$ is invertible if and only if $b^T A^{-1} a \neq -1$; in this case,*

$$P^{-1} = A^{-1} - \frac{1}{1 + b^T A^{-1} a}\left[A^{-1} a b^T A^{-1}\right].$$

Proof. ($\Leftarrow$) If $b^T A^{-1} a \neq -1$, then the expression on the right side of the displayed equation is well-defined. Just multiplying this expression by P produces the identity matrix, so P must be invertible and the given expression must be its inverse. (The key to success here is recognizing that $b^T A^{-1} a$ is a scalar, even when it shows up in the middle of a long matrix product in the expansion of the product mentioned above.)

($\Rightarrow$) If $b^T A^{-1} a = -1$, then certainly $a \neq 0$; consequently $w \stackrel{def}{=} A^{-1} a$ is nonzero. But

$$Pw = \left[A + ab^T\right] A^{-1} a = a + a\left(b^T A^{-1} a\right) = a - a = 0.$$

Hence P cannot be invertible. ////

Remark. In general $\left(A^{-1}\right)^T = \left(A^T\right)^{-1} = A^{-T}$. If A is symmetric, it follows that $A^{-T} = A^{-1}$, so the Sherman-Morrison formula can be reorganized into a sum of A^{-1} with a transformed dyad:

$$\left[A + ab^T\right]^{-1} = A^{-1} - \frac{1}{1 + b^T A^{-1} a}\left(A^{-1} a\right)\left(A^{-1} b\right)^T.$$

BFGS (Improved). The BFGS update rule for approximate hessians builds H_{k+1} from H_k by adding two dyads. Hence two applications of the Sherman-Morrison formula will allow us to express $W_{k+1} \stackrel{def}{=} H_{k+1}^{-1}$ in terms of $W_k \stackrel{def}{=} H_k^{-1}$ and the vectors in these dyads. Carrying this calculation forward takes some care (exercise!), and produces

$$W_{k+1} = W_k + \frac{s^T y + y^T W_k y}{\left(s^T y\right)^2}\left[ss^T\right] - \frac{1}{s^T y}\left[W_k y s^T + s y^T W_k\right],$$
$$\text{where } s = x_{k+1} - x_k, \quad y = \left(\nabla f(x_{k+1}) - \nabla f(x_k)\right)^T.$$

This observation will speed up the implementation of a BFGS Quasi-Newton method. Recall that in Step 3 of the general framework, we must find the search direction v_k from point x_k by solving the linear system

$$H_k v_k = -\nabla f(x_k)^T.$$

The equivalent expression $v_k = -H_k^{-1}\nabla f(x_k)^T$ is dangerous if it deceives the unwary into computing H_k^{-1} numerically, but it's just fine when—like now—a tidy formula for H_k^{-1} is available. So the search directions can be found easily using

$$v_k = -W_k \nabla f(x_k)^T$$

for the sequence of approximate inverse Hessians W_k generated above. The only matrix inversion required is the computation of $W_0 = H_0^{-1}$, and in many implementations the matrix H_0 is so simple that W_0 can be found quickly and easily.

DFP. A second reasonable update rule arising from similar principles is the Davidon-Fletcher-Powell (DFP) scheme. One way to produce this is to apply the construction in the Lemma above directly to the problem of producing H^{-1}. Given vectors y and s with $y^T s > 0$, the equation $Hs = y$ is equivalent to $s = H^{-1}y$. Write this as $\widetilde{H}\widetilde{s} = \widetilde{y}$ for $\widetilde{H} = H^{-1}$, $\widetilde{s} = y$, and $\widetilde{y} = s$, and the lemma ensures that any nonsingular $\widetilde{H}_0$ will produce a positive-definite, symmetric $\widetilde{H}$ via

$$\widetilde{H} = \widetilde{H}_0 + \frac{1}{\widetilde{y}^T\widetilde{s}}\left[\widetilde{y}\widetilde{y}^T\right] - \frac{1}{\widetilde{s}^T\widetilde{H}_0\widetilde{s}}\left[\widetilde{H}_0\widetilde{s}\widetilde{s}^T\widetilde{H}_0\right].$$

Writing $W_{k+1} = \widetilde{H}$, $W_k = \widetilde{H}_0$, and simplifying the notation for $\widetilde{s}$ and $\widetilde{y}$, we have

$$W_{k+1} = W_k + \frac{1}{s^T y}\left[ss^T\right] - \frac{1}{y^T W_k y}\left[W_k y y^T W_k\right].$$

With the same choices for s and y as before, this is the DFP updating formula for the inverse approximate Hessian; two applications of the Sherman-Morrison formula give the corresponding formula for updating the Hessian itself:

$$H_{k+1} = H_k + \frac{y^T s + s^T H_k s}{\left(y^T s\right)^2}\left[yy^T\right] - \frac{1}{y^T s}\left[H_k s y^T + y s^T H_k\right].$$

Notice the symmetry between the DFP formula for updating H and the BFGS formula for updating $W = H^{-1}$.

Self-Correcting. BFGS tends to compensate for bad initial approximations for the Hessian. Experimental evidence shows that it does a better job at this than DFP, and the experts think that this explains why it seems to work better in practice.

Practical Matters—Updating. There are some cases where changing the approximate Hessian is a waste of time because the current approximation is good enough. Test for this by recalling the user's convergence tolerance $\eta_1 > 0$ for the x-values. If each component of the residual $y - H_k s$ is smaller than $\eta_1 \left(|\nabla f(x_k)| + |\nabla f(x_{k+1})|\right)$ [the estimated noise in the corresponding component], just update $H_{k+1} = H_k$.

The BFGS update is sure to work if the Wolfe Conditions are satisfied. Typical choices for the parameters are $c_1 = 10^{-4}$, $c_2 = 0.9$. Our backtracking Armijo rule confirms only the descent condition (D), not the curvature condition (C). Best remedy: a better approximate line search that enforces both.

Ad-hoc approaches—use one or both of these for HW03. Assume the usual situation: We're at stage k. We know x_k and H_k, and we have used them to find x_{k+1} using some kind of line search. We're ready to start stage $k+1$. Our notation is

$$y = \left(\nabla f(x_{k+1}) - \nabla f(x_k)\right)^T, \qquad s = x_{k+1} - x_k.$$

1. **Skipping (Not Great).**
   - If $y^T s \leq \sqrt{\varepsilon_{\text{machine}}} \|y\|_2 \, \|s\|_2$, update $H_{k+1} = H_k$.
   - Otherwise, get H_{k+1} (or, better, W_{k+1}) from the BFGS or DFP updating formula given above.

   [Problem: This might set $H_{k+1} = H_k$ over and over again many times, so the curvature information being collected never gets used.]

2. **Damping (better).** Define $q_k = (0.2) s^T H_k s$. Note that $q_k > 0$. Build

$$\theta_k = \begin{cases} 1, & \text{if } s^T y \geq q_k, \\ 0.8 \dfrac{s^T H_k s}{s^T H_k s - s^T y}, & \text{if } s^T y < q_k. \end{cases}$$

   Then let

$$r_k = \theta_k y_k + (1 - \theta_k) H_k s_k$$

   and update

$$H_{k+1} = H_k + \frac{1}{r^T s} \left[r r^T\right] - \frac{1}{s^T H_k s} \left[H_k s s^T H_k\right].$$

   This is the same update formula as for BFGS, only we have changed y_k to r_k everywhere. [Advantage: This always uses the curvature info at least a little.]

   This strategy interpolates between the unmodified BFGS update (which it selects whenever $\theta_k = 1$, i.e., $s^T y$ is positive and reasonably big), and the do-nothing choice $H_{k+1} = H_k$ (which would correspond to $\theta_k = 0$). Note that $s^T r > 0$ in all cases (exercise!), so the update formula above does produce a positive-definite matrix at every step.

Convergence. Superlinear convergence is expected under reasonable hypotheses. The proofs in the literature do assume Lipschitz continuity of $D^2 f$ to establish this, though, in spite of the fact that the algorithm uses only first derivatives. The motivation for the algorithm makes this not completely surprising.

E. Starting

To get the algorithm moving we need some matrix to use for $H_0 = H_0^T > 0$. One could use an exact Hessian, a numerically-approximated Hessian, or do something much simpler: just pick

$$H_0 = \beta I \qquad (W_0 = H_0^{-1} = \beta^{-1} I).$$

Ideally β should be related somehow to the typical size of f. HW03 says choose $\beta = |f(x_0)|$, which is not so smart. It's better to ask the user to guess how long the first Newton step ought to be, call this value δ, then choose

$$\beta = \frac{\delta}{\|\nabla f(x_0)\|}.$$

For an even better start, Nocedal and Wright (pp. 200-201) suggest a modified BFGS update on the first step only. (This is a "heuristic"—i.e., a good idea that often works, but does not have a rock-solid theoretical justification applicable in all cases.) After using $H_0 = \beta I$ to compute x_1, and define y_1 and s_1 as above, they replace H_0 with $\widetilde{b}I$, for

$$\widetilde{\beta} = \left(\frac{y_1^T s_1}{y_1^T y_1} \right),$$

before applying the update rule (BFGS) to generate H_1.

F. Stopping and Scaling

Reasonable stopping tests should be independent of the scale of both the variables x and the function values f. Hints were provided on Assignment 2: these need only to be modified by safeguarding them against division by zero.

Iteration should stop after the computation of a new point x_{k+1} either if

(i) This point is very close to the previous point x_k, i.e., for some (user-specified) tolerance η_1,

$$\max_{i=1,\ldots,n} \frac{|(x_{k+1})_i - (x_k)_i|}{\max\{|(x_{\text{typ}})_i|, |(x_k)_i|\}} \leq \eta_1.$$

On the left is the largest relative error among all components.

(ii) The gradient is nearly zero, i.e., for some (user-specified) tolerance η_2,

$$\max_{i=1,\ldots,n} \frac{|\nabla f(x_{k+1})_i (x_{k+1})_i|}{\max\{|f_{\text{typ}}|, |f(x_{k+1})|\}} \leq \eta_2,$$

The left side measures the largest relative slope among all components.

(iii) Time and money have run out, i.e., for some (user-specified) maximum number of iterations $K_{\max}$,

$$k + 1 \geq K_{\max}.$$