Improving Newton's Method

MA 348

Kurt Bryan

Introduction

Recall Newton's method for minimizing $f(\mathbf{x})$:

1. Make an initial guess $\mathbf{x}_0$, set $k = 0$.

2. Solve the linear system
$$\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$$
for the step $\mathbf{h}_k$. (This comes from approximating f as a quadratic function, $f(\mathbf{x}) \approx f(\mathbf{x}_k) + \mathbf{h}^T\nabla f(\mathbf{x}_k) + \frac{1}{2}\mathbf{h}^T H(\mathbf{x}_k)\mathbf{h}$ where $\mathbf{h} = \mathbf{x} - \mathbf{x}_k$, and finding the critical point $\mathbf{h} = \mathbf{h}_k$ of the approximating quadratic.)

3. Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}_k$; if $\mathbf{h}_k$ is small (or $|\nabla f(\mathbf{x}_k)|$ is small) then terminate with minimum $\mathbf{x}_{k+1}$; otherwise increment $k = k + 1$, and repeat step 2.

Newton's method is *exact* on quadratic functions: it will find the minimum (or critical point) in one iteration.

More generally, Newton's method can be shown to converge quadratically on "nice" functions. Specifically, if f has continuous third derivatives, and if $\mathbf{H}(\mathbf{x}^*)$ is positive definite at the minimum $\mathbf{x}^*$, and if the initial guess $\mathbf{x}_0$ is close enough to $\mathbf{x}^*$, then

$$|\mathbf{x}_{k+1} - \mathbf{x}^*| \leq c|\mathbf{x}_k - \mathbf{x}^*|^2$$

for some constant c. This is extremely fast convergence.

However, Newton's method as presented above will just as happily find ANY critical point of f, whether a max, min, or saddle. And unfortunately, "most" critical points are not minima. Thus it would be nice to modify Newton's method to bias it toward finding critical points which are actually minima.

We'll make a small modification to Newton's method to encourage it to head for a minimum. First, recall that a matrix M is said to be *positive definite* if $\mathbf{v}^T\mathbf{M}\mathbf{v} > 0$ for all nonzero vectors $\mathbf{v}$. Second, recall the method of steepest descent: at a given iteration at some base point $\mathbf{x}_k$ we compute a search direction $\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$. This search direction is guaranteed to be a *descent direction*, in the sense that if we leave $\mathbf{x}_k$ along the line $\mathbf{L}(t) = \mathbf{x}_k + t\mathbf{h}_k$ then as t increases the function value decreases (at least for awhile). This is a simple consequence of the chain rule, which says that

$$\frac{d}{dt}(f(\mathbf{L}(t)) = \nabla f(\mathbf{L}(t)) \cdot \mathbf{L}'(t) = -\nabla f(\mathbf{L}(t)) \cdot \nabla f(\mathbf{x}_k).$$

When $t = 0$ ($\mathbf{L}(0) = \mathbf{x}_k$, so we're just leaving the base point) the above formula gives $\frac{d}{dt}(f(\mathbf{L}(t)) = -|\nabla f(\mathbf{x}_k)|^2 < 0$. In other words, the search direction $-\nabla f(\mathbf{x}_k)$ in steepest descent points downhill. We then follow this direction and perform a 1D line search.

Something similar can be made to happen in Newton's method; with a small modification the vector $\mathbf{h}_k$ chosen in step (2) of Newton's method above will be a descent direction.

Lemma 1: *Suppose that at a given iteration of Newton's method we have that the Hessian matrix $\mathbf{H}(\mathbf{x}_k)$ is positive definite, and that $\nabla f(\mathbf{x}_k) \neq 0$. Then the direction $\mathbf{h}_k$ computed in Newton's method is a descent direction, i.e.,*

$$\frac{d}{dt}(f(\mathbf{x}_k + t\mathbf{h}_k)) < 0$$

at $t = 0$.

To prove Lemma 1 just note that $\frac{d}{dt}(f(\mathbf{x}_k + t\mathbf{h}_k)) = \nabla f(\mathbf{x}_k + t\mathbf{h}_k)^T\mathbf{h}_k$. At $t = 0$ this becomes

$$\frac{d}{dt}(f(\mathbf{x}_k + t\mathbf{h}_k))|_{t=0} = \nabla f(\mathbf{x}_k)^T\mathbf{h}_k.$$

But from Newton's method we have $\nabla f(\mathbf{x}_k) = -\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k$, so we have

$$\begin{aligned}
\frac{d}{dt}(f(\mathbf{x}_k + t\mathbf{h}_k))|_{t=0} &= \nabla f(\mathbf{x}_k)^T\mathbf{h}_k \\
&= -\mathbf{h}_k^T\mathbf{H}(\mathbf{x}_k)^T\mathbf{h}_k \\
&= -\mathbf{h}_k^T\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k \\
&< 0
\end{aligned}$$

where I've used the matrix algebra fact that $(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$ for any matrices or vectors $\mathbf{A}$ and $\mathbf{B}$, and that $\mathbf{H} = \mathbf{H}^T$ (remember $\mathbf{H}$ is symmetric). And of course in the last step I used that $\mathbf{H}(\mathbf{x}_k)$ is assumed to be positive definite. This proves Lemma 1.

What this means is that if we go in the *direction* $\mathbf{h}_k$ computed in step (2) of Newton's method, we're guaranteed to go downhill, IF $\mathbf{H}(\mathbf{x}_k)$ is positive definite. It doesn't mean that $f(\mathbf{x}_k + \mathbf{h}_k) < f(\mathbf{x}_k)$, but merely that if we leave $\mathbf{x}_k$ and head in the direction $\mathbf{h}_k$ we'll be going downhill initially. So in our modified Newton's method we don't take $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}_k$ as in straight Newton's method—instead we take $\mathbf{h}_k$ as a search direction and perform a line search (more on this later), minimizing the one variable function

$$g(t) = f(\mathbf{x}_k + t\mathbf{h}_k)$$

in t. If we find a minimum at $t = t^*$ then we take $\mathbf{x}_{k+1} = \mathbf{x}_k + t^*\mathbf{h}_k$. Straight Newton's method corresponds to always making the choice $t^* = 1$.

Cholesky Factorization

The fact that $\mathbf{h}_k$ is a descent direction relies on $\mathbf{H}(\mathbf{x}_k)$ being positive definite. What if that's not true? In fact, if $\mathbf{H}(\mathbf{x}_k)$ is negative definite (i.e., $-\mathbf{H}$ is positive definite) then $\mathbf{h}_k$ is guaranteed to point UPHILL! How can we tell if $\mathbf{H}$ is positive definite, and if it isn't, what do we do about it?

These questions all tie in nicely with step (2) of Newton's method, in which we must solve the linear system $\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$. There is a very effective method for solving linear systems of the form

$$\mathbf{M}\mathbf{x} = \mathbf{b} \tag{1}$$

where $\mathbf{M}$ is a symmetric positive definite matrix. It's called *Cholesky Factorization*. It's faster than Gaussian elimination (or LU decomposition, if you've heard of that), and it can tell when a matrix is or is not positive definite.

We will use $\mathbf{L}$ to represent a unit lower triangular matrix, that is, a matrix of the form

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ L_{21} & 1 & 0 & \cdots & 0 \\ L_{31} & L_{32} & 1 & \cdots & 0 \\ & & & \vdots & \\ L_{n1} & L_{n2} & L_{n3} & \cdots & 1 \end{bmatrix},$$

and $\mathbf{D}$ to represent a diagonal matrix, i.e., a matrix with zeros everywhere except on the diagonal.

Theorem: *Any symmetric positive definite matrix* $\mathbf{M}$ *can be factored as*

$$\mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^T$$

for some unit lower triangular matrix $\mathbf{L}$ *and diagonal matrix* $\mathbf{D}$ *which has entirely positive entries on the diagonal.*

The factorization above is called the *Cholesky* factorization of $\mathbf{M}$. One thing is easy to check immediately: Any matrix $\mathbf{M}$ which can be expressed as $\mathbf{M} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ is symmetric and positive definite.

Proof of Theorem: How about just an example? You can consult a linear algebra book (or section 2.4 of Jeff Leader's Numerical book) for the general proof. The matrix

$$\mathbf{M} = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 5 \end{bmatrix}$$

is positive definite (which I'll prove here in a moment.) Consider trying to express $\mathbf{M}$ via the factorization above. We would need

$$\mathbf{M} = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{bmatrix} \begin{bmatrix} D_1 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & D_3 \end{bmatrix} \begin{bmatrix} 1 & L_{21} & L_{31} \\ 0 & 1 & L_{32} \\ 0 & 0 & 1 \end{bmatrix}.$$

Now we can easily multiply out the $\mathbf{D}$ and $\mathbf{L}^T$ matrices on the right to get

$$\mathbf{M} = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ L_{21} & 1 & 0 \\ L_{31} & L_{32} & 1 \end{bmatrix} \begin{bmatrix} D_1 & D_1L_{21} & D_1L_{31} \\ 0 & D_2 & D_2L_{32} \\ 0 & 0 & D_3 \end{bmatrix}.$$

Now multiplying the first row in $\mathbf{L}$ times the first column in the second matrix shows that $D_1 = 2$. Multiplying the first row in $\mathbf{L}$ times the second column gives $D_1L_{21} = -1$, so $L_{21} = -1/2$. The first row in $\mathbf{L}$ times the third column gives $D_1L_{31} = 1$, so $L_{31} = 1/2$.

If we fill in what's known, here's how things stand at the moment:

$$\mathbf{M} = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/2 & L_{32} & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 1 \\ 0 & D_2 & D_2L_{32} \\ 0 & 0 & D_3 \end{bmatrix}.$$

You can check that the second row in $\mathbf{L}$ times the first column in the second matrix on the right gives -1, the correct entry in $\mathbf{M}$—but then that HAD to work, given that $\mathbf{LDL}^T$ must be symmetric. Now multiply second row times second column to get $1/2 + D_2 = 3$, so $D_2 = 5/2$. Second row by third column gives $-1/2 + D_2L_{32} = 0$, so $L_{32} = (1/2)/D_2 = 1/5$. We currently have

$$\mathbf{M} = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/2 & 1/5 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 1 \\ 0 & 5/2 & 1/2 \\ 0 & 0 & D_3 \end{bmatrix}.$$

Finally, take the third row times third column to find that $D_3 = 22/5$. The entire decomposition looks like

$$\mathbf{M} = \begin{bmatrix} 2 & -1 & 1 \\ -1 & 3 & 0 \\ 1 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/2 & 1/5 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 5/2 & 0 \\ 0 & 0 & 22/5 \end{bmatrix} \begin{bmatrix} 1 & -1/2 & 1/2 \\ 0 & 1 & 1/5 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

Notice how at each stage (some row times some column) we always had from a previous stage the information needed to solve for exactly one more unknown. This holds in general if the original matrix is positive definite. It's not hard to see how to do the Cholesky decomposition for a general n by n matrix—everything just falls into place. Moreover, the

procedure as detailed above is numerically stable for positive definite matrices. You can do an operation count to see that the procedure requires about $n^3/6$ operations for an n by n matrix. This is about half as many operations as needed to do a standard LU decomposition on $\mathbf{M}$.

This also gives an efficient way to solve the equation $\mathbf{Mx} = \mathbf{b}$. If we have the Cholesky decomposition of $\mathbf{M}$ this becomes $\mathbf{LDL}^T\mathbf{x} = \mathbf{b}$. We can solve the system by first solving $\mathbf{Lx}' = \mathbf{b}$ (so $\mathbf{DL}^T\mathbf{x} = \mathbf{x}'$), then $\mathbf{Dx}'' = \mathbf{x}'$ (so $\mathbf{L}^T\mathbf{x} = \mathbf{x}''$), then finally $\mathbf{L}^T\mathbf{x} = \mathbf{x}''$. All of these systems are easy to solve by "backsubstitution." For example, suppose $\mathbf{b} = (1, -2, 3)^T$ for the example $\mathbf{M}$ above. The equation $\mathbf{Lx}' = \mathbf{b}$ is just

$$\begin{bmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/2 & 1/5 & 1 \end{bmatrix} \begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 3 \end{bmatrix}.$$

We immediately obtain $x_1' = 1$. The second equation is just $(-1/2)x_1' + x_2' = 2$, or $x_2' = -3/2$. The last equation is $(1/2)x_1' + (1/5)x_2' + x_3' = 3$, so $x_3' = 14/5$. In short, $\mathbf{Lx}' = \mathbf{b}$ is easy to solve because $\mathbf{L}$ is lower triangular—we can pick off the x_k' one at a time.

Solving $\mathbf{Dx}'' = \mathbf{x}'$ is even more trivial (think about it). You get $\mathbf{x}_1'' = 1/2$, $\mathbf{x}_2'' = -3/5$, $\mathbf{x}_3'' = 7/11$. Finally, solving $\mathbf{L}^T\mathbf{x} = (1/2, -3/5, 7/11)^T$ is also easy—backsubstitute as for $\mathbf{L}$, but start from the bottom up. You get $x_3 = 7/11$, $x_2 = -8/11$, $x_1 = -2/11$.

Cholesky factorization is a very standard method for solving linear systems involving positive definite matrices, and would be the logical choice for solving $\mathbf{Hh}_k = -\nabla f$ in Newton's method if we know that $\mathbf{H}$ is positive definite. What happens if we feed a symmetric matrix $\mathbf{M}$ which isn't positive definite to Cholesky? Then at some stage we will find that $D_k \leq 0$ for some k. In fact, the computation may break down entirely—try doing a Cholesky decomposition on the matrix

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

We need to make a small modification.

Improved Newton's Method

By using Cholesky decomposition in step (2) of Newton's method, we can implement a strategy to guarantee that the direction $\mathbf{h}_k$ computed is always a descent direction. Let's suppose we are currently faced with the task of solving $\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$. We try to perform a Cholesky decomposition. Suppose that in computing some D_k we find that $D_k \leq 0$—then we replace D_k by some fixed number $\delta > 0$, and continue with the factorization. When we're done we have matrices $\mathbf{L}$ and $\mathbf{D}$, and we can form a positive definite matrix $\tilde{\mathbf{H}} = \mathbf{LDL}^T$, but $\tilde{\mathbf{H}} \neq \mathbf{H}$. However, $\tilde{\mathbf{H}}$ is "closely related" to $\mathbf{H}$. Moreover, if we solve $\tilde{\mathbf{H}}\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$ then $\mathbf{h}_k$ is guaranteed to be a descent direction. This can be shown

exactly as in Lemma 1 above—just replace $\mathbf{H}(\mathbf{x}_k)$ by $\tilde{\mathbf{H}}$ and the entire computation proceeds unchanged. But note that if $\mathbf{H}$ is positive definite (with all diagonal entries of $\mathbf{D}$ larger than the chosen δ) then $\tilde{\mathbf{H}} = \mathbf{H}$ and we obtain the usual Newton step.

The general philosophy is this: Far from a minimum, where the quadratic approximation on which Newton is based is a poor model for f, we expect to find $\mathbf{H}$ is not necessarily positive definite, but this modified Newton's Method still gives a descent direction and we can make progress downhill. When we get closer to a minimum (and so $\mathbf{H}$ should become positive definite) then $\tilde{\mathbf{H}} = \mathbf{H}$, and so the search direction is chosen as in the usual Newton's method, although we don't take the full step dictated by Newton's method, but rather do a line search in that direction.

A simple outline of a modified Newton's method would look like

1. Set an initial guess $\mathbf{x}_0$, set $k = 0$.

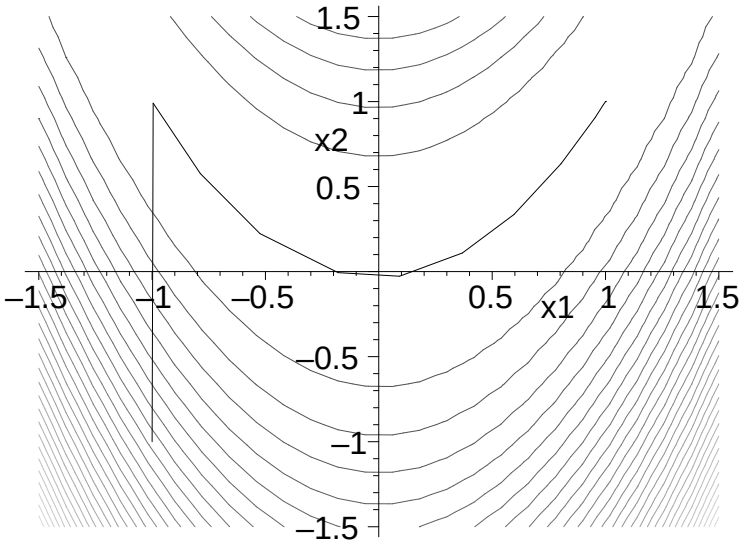
2. Solve the linear system
$$\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$$
for the search direction $\mathbf{h}_k$, using the modified Cholesky decomposition (if $D_k < \delta$ at some stage of the decomposition for some fixed positive δ, replace D_k with δ.)

3. Perform a line search from $\mathbf{x}_k$ in the direction $\mathbf{h}_k$ (guaranteed to be a descent direction.)

4. Take $\mathbf{x}_{k+1}$ as the minimizing point in the line search; if $\mathbf{h}_k$ is small (or $|\nabla f(\mathbf{x}_k)|$ is small) then terminate with minimum $\mathbf{x}_{k+1}$; otherwise increment $k = k + 1$, and repeat step 2.

We could also make the simple modification that when we get close to a minimum (perhaps as measured by $|\mathbf{x}_{k+1} - \mathbf{x}_k|$) then we dispense with the line search and take the unmodified Newton step.

Here's an example using Rosenbrock's function, $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ with starting point $(-1, -1)$. I chose $\delta = 0.1$ in the Cholesky decomposition.The algorithm required 14 iterations to locate the minimum to within 10^{-6}.



Line Search Algorithms

Both steepest descent and the modified Newton's method fall into a class of optimization algorithms call "line search algorithms." They all fit into the following mold:

1. Set an initial guess $\mathbf{x}_0$, set $k = 0$.

2. Compute a search direction $\mathbf{h}_k$. Make sure it is a descent direction.

3. Perform a line search from $\mathbf{x}_k$ in the direction $\mathbf{h}_k$. Take $\mathbf{x}_{k+1}$ as the minimizing point in the line search, or less stringently, require that $\mathbf{x}_{k+1}$ be chosen so that $f(\mathbf{x}_{k+1})$ is "sufficiently" less than $f(\mathbf{x}_k)$.

4. If $|\nabla f(\mathbf{x}_k)|$ is small or $\mathbf{x}_{k+1}$ is close to $\mathbf{x}_k$ then terminate with minimum $\mathbf{x}_{k+1}$; otherwise increment $k = k + 1$, and repeat step 2.

It may appear that one should perform an *exact* line search in step (3), that is, locate the minimum along the line $\mathbf{x}_k + t\mathbf{h}_k$ to high accuracy. In fact, this is not always necessary or desirable. The effort expended in doing this (the excess function and possibly derivative evaluations) may not compensate for the resulting decrease in functional value; it's often better to do an "inexact" line search and expend additional computational effort in computing a new search direction. Let's look at one strategy for doing a less rigorous line search.

The Armijo Condition

Let the current base point be $\mathbf{x}_k$ and the search direction be $\mathbf{h}_k$; we will do a line search in the direction $\mathbf{h}_k$, i.e., try to minimize $g(t) = f(\mathbf{x}_k + t\mathbf{h}_k)$. We'll look at a technique which doesn't precisely minimize $g(t)$—just decreasing its value "sufficiently" from $g(0) = f(\mathbf{x}_k)$ is often good enough.

One common method for quantifying "sufficient decrease" is as follows. You can compute that

$$g'(t) = \nabla f(\mathbf{x}_k + t\mathbf{h}_k) \cdot \mathbf{h}_k.$$

Given that near $t = 0$ we have $g(t) \approx g(0) + tg'(0)$ this leads to $f(\mathbf{x}_k + t\mathbf{h}_k) \approx f(\mathbf{x}_k) + t\nabla f(\mathbf{x}_k) \cdot \mathbf{h}_k$ or

$$f(\mathbf{x}_k + t\mathbf{h}_k) - f(\mathbf{x}_k) \approx t\nabla f(\mathbf{x}_k) \cdot \mathbf{h}_k. \tag{3}$$

This quantifies how much we can expect to decrease f by moving small distance along the line $\mathbf{x}_k + t\mathbf{h}_k$ starting at $\mathbf{x}_k$ ($t = 0$); note that if $\mathbf{h}_k$ is a descent direction then $f(\mathbf{x}_k + t\mathbf{h}_k) - f(\mathbf{x}_k) < 0$. Let $t = t_k$ be the value of t that we eventually accept in our line search. The Armijo condition for sufficient decrease is that

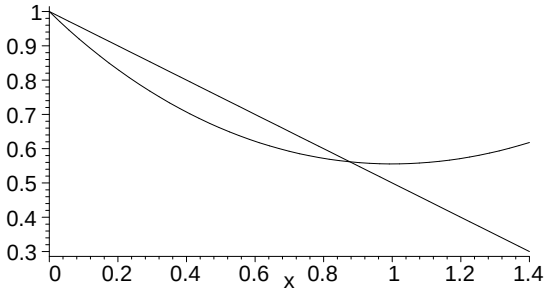
$$f(\mathbf{x}_k + t_k\mathbf{h}_k) - f(\mathbf{x}_k) < \mu t_k \nabla f(\mathbf{x}_k) \cdot \mathbf{h}_k \tag{4}$$

where μ is some constant with $0 < \mu < 1$. In other words, the resulting step from $t = 0$ to $t = t_k$ must not merely decrease the function value, but must decrease it by at least some specified fraction of the decrease predicted by the approximation in equation (3).

If μ is close to zero then condition (4) is easier to satisfy, but as a result the line search could be quite inefficient—almost any step t is likely to work. Taking μ close to 1 may seem desirable, but can result in very small step sizes t_k (and again, inefficiency in the line search). But for $\mu < 1$ there's always some choice for t_k which will work. To see this, suppose, to the contrary, that for all t sufficiently close to 0 we have $f(\mathbf{x}_k + t\mathbf{h}_k) - f(\mathbf{x}_k) \geq \mu\nabla f(\mathbf{x}_k) \cdot \mathbf{h}_k$. Divide by t and take the limit as $t \to 0$, then divide by $\nabla f(\mathbf{x}_k) \cdot \mathbf{h}_k$ (which is negative) to obtain $\mu \geq 1$, a contradiction.

Thus taking $\mu \approx 1$ may only force any line search to take $t \approx 0$ and hence small, inefficient steps. Typically one chooses μ in the range 0.1 to 0.5, although this is hardly set in stone.

Here's a picture illustrating the Armijo condition. It is a graph of $g(t) = f(\mathbf{x}_k + t\mathbf{h}_k)$ for some f. The slope $g'(0) = -1$. With $\mu = 1/2$ the Armijo condition is that t be chosen so that $g(t) - g(0) < \frac{1}{2}tg'(0)$, which is equivalent to requiring that $g(t)$ lies below the line $L(t) = g(0) + \frac{1}{2}g'(0)t$.



Here is a concrete strategy for doing an inexact line search using the Armijo condition. Given the descent direction $\mathbf{h}_k$ computed from modified Newton's method we set $t = 1$ and check to see if $f(\mathbf{x}_k + t\mathbf{h}_k)$ satisfies the Armijo condition (4). If it does we take $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{h}_k$. If not we set (for example) $t = 1/2$ and try again—if the Armijo condition is satisfied we take $\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{1}{2}\mathbf{h}_k$. If that still doesn't satisfy the Armijo condition we continue trying $t = 2^{-j}$; eventually some value of t must succeed (after t gets small enough this is guaranteed.) When this happens we take $\mathbf{x}_{k+1} = \mathbf{x}_k + t\mathbf{h}_k$.

Here it is written out:

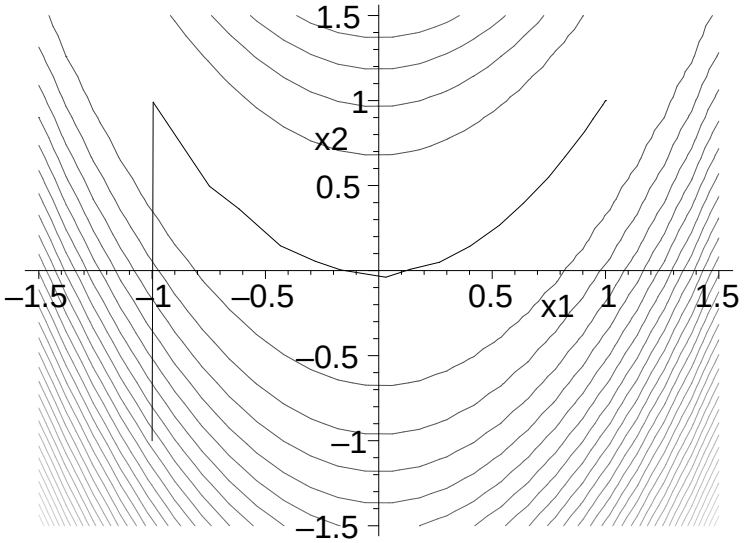
1. Set an initial guess $\mathbf{x}_0$, set $k = 0$.

2. Solve the linear system
$$\mathbf{H}(\mathbf{x}_k)\mathbf{h}_k = -\nabla f(\mathbf{x}_k)$$
for the search direction $\mathbf{h}_k$, using the modified Cholesky decomposition (if $D_k < \delta$ at some stage of the decomposition for some fixed positive δ, replace D_k with δ.)

3. Try $t = 2^{-j}$ for $j = 0, 1, 2, \ldots$ and accept the first value for t which satisfies
$$f(\mathbf{x}_k + t\mathbf{h}_k) - f(\mathbf{x}_k) < \mu t \nabla f(\mathbf{x}_k) \cdot \mathbf{h}_k.$$

4. Take $\mathbf{x}_{k+1} = \mathbf{x}_k + t\mathbf{h}_k$; if $|\nabla f(\mathbf{x}_k)|$ is small (or $|\mathbf{x}_{k+1} - \mathbf{x}_k|$ is small) then terminate with minimum $\mathbf{x}_{k+1}$; otherwise increment $k = k + 1$, and repeat step 2.

Here again is an example using Rosenbrock's function, $f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ with starting point $(-1, -1)$. I chose $\delta = 0.1$ in the Cholesky decomposition and $\mu = 0.5$. The algorithm required 22 iterations to locate the minimum to within 10^{-6}.


1.5
1
x2
0.5
–1.5
–1
–0.5
0.5
x1
1
1.5
–0.5
–1
–1.5