Newton's Method [*]

Juan C. Meza[**]

March 15, 2010

Abstract

Newton's method is one of the most powerful techniques for solving systems of nonlinear equations and minimizing functions. It is easy to implement and has a provably fast rate of convergence under fairly mild assumptions. Because of these and other nice properties, Newton's method is at the heart of many solution techniques used to solve real-world problems. This article, gives a short introduction to this method with a brief discussion of some of the main issues in applying this algorithm for the solution of practical problems.

1 Introduction

Newton's method is perhaps the best known method for finding the root of a nonlinear equation or for minimizing a general nonlinear function. This method, also known as the Newton-Raphson method, can be traced back to Isaac Newton(c. 1669) and Joseph Raphson (1690). Both Newton and Raphson appear to have derived essentially the same method independently. Newton based his derivation on a linearization of a higher-order polynomial and he showed how the method could be used to solve a particular cubic equation. Raphson's scheme on the other hand more closely resembles what we now know as Newton's method.

In its basic form, Newton's method is easy to implement and requires only the ability to compute a function and its first and second derivatives. One of the main advantages of Newton's method is the fast rate of convergence that it possesses and a well-studied convergence theory that provides the underpinnings for many other methods. In practice, however, Newton's method needs to be modified to make it more robust and computationally efficient. With these modifications, Newton's method (or one of its many variations) is arguably the method of choice for a wide variety of problems in science and engineering.

*This work was supported by the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

**Lawrence Berkeley National Laboratory, Berkeley, CA 94720

2 General Description

Newton's method is most commonly introduced as a technique for finding roots of a general nonlinear function. Since finding the minima of a nonlinear function amount to finding the roots of the gradient of a function, we can also study Newton's method from that perspective. Suppose then that we would like to find the minimum of a general nonlinear function $f(x)$, starting from some known initial guess $x_0 \in R^N$. Newton's method can be derived in several ways, the two most popular being to use a Taylor's theorem approach and the second being to use a quadratic model of the function that one wishes to minimize. Here we will take the quadratic model approach as it can provide a framework from which we can make several generalizations. We will denote the first derivative of $f(x)$ by $g(x) = \nabla f(x)$, and the second derivative matrix, H by $H(x) = \nabla^2 f(x)$. The first derivative is often referred to as the gradient or Jacobian and the second derivative matrix as the Hessian. We can now write a quadratic approximation to the nonlinear function $f(x)$ by

$$q(x) = x_0 + g^T x + \frac{1}{2} x^T H x. \tag{1}$$

The next step is to use the minimizer of $q(x)$ as an approximation to the minimizer of $f(x)$. It is easy to see that the unique minimizer to equation (1) can be obtained by taking the derivative with respect to x and finding the zero of the resulting equation, i.e.,

$$\nabla q(x) = g + Hx = 0, \tag{2}$$

or

$$Hx = -g. \tag{3}$$

We can then define the step one would take as

$$s = x_{k+1} - x_k. \tag{4}$$

Equation (3) is commonly called the Newton equation and the step generated by equation (4) is the Newton step. This then leads to the following algorithm for Newton's method.

3 Convergence Theory

Newton's method has a well-studied convergence theory that guarantees the convergence to a solution under a standard set of assumptions. For a detailed description

Algorithm 1 Newton's Method

```
Given an initial $x_0$ and a convergence tolerance $tol$
for $k = 0$ to $maxiter$ do
    Compute the step, $s_k = -H^{-1}(x_k)g(x_k)$
    $x_{k+1} = x_k + s_k$
    if $||g(x_{k+1})|| \leq tol$ then
        converged
    end if
end for
```

of the convergence theory, the interested reader should consult one of the many excellent books on this subject [1, 2, 5]. Here we only point out the main ideas. If we assume that a solution exists, then the main assumptions are that the gradient or Jacobian of the function is Lipschitz continuous near the solution, and that it is in fact nonsingular at the solution. As a reminder, a function is said to be Lipschitz continuous near a point x^* if there exists a constant $\gamma > 0$, such that

$$||g(x) - g(y)|| \leq \gamma ||x - y||,$$

for all x, y near x^*. Under these assumptions, it is easy to show that the Newton algorithm as described above will converge to the solution for any starting point x_0 that is sufficiently close to the solution x^* and that there exists a constant C such that the error, $e_{k+1} = x_{k+1} - x^*$ at the $k+1$ iteration satisfies

$$||e_{k+1}|| \leq C||e_k||^2,$$

for k large enough. In other words the error decreases quadratically at each iteration. A common way of interpreting this result is that close enough to the solution the number of digits of accuracy in the computed solution will double at each iteration.

While this is satisfying from a theoretical point, the question of what "sufficiently close" means can be hard to quantify for any given problem, and the fast quadratic rate of converge may not apply until after many iterations. Still, for many problems, Newton's method is the method of choice, and it is certainly worth applying if the necessary derivative information is available.

4 Practicalities

Although Newton's method is quite powerful and simple to implement, there are many obstacles to using it on real world applications. When someone first implements a Newton method scheme, the most common observations are that Newton's method is converging too slowly or not at all, or that some derivative information is not available. Once the method is working, then attention is usually turned to the computational cost, which can be prohibitive for medium to large-scale problems. Let us now consider how one might address some of these issues.

4.1 Robustness

Most people who write their own implementation fail to consider the effect of the line search parameter. As scientists become more familiar with computational methods this has become rarer, although it is not uncommon to still see an implementation without step length restrictions. To be more explicit, define the Newton direction by the formula:

$$s_k = -H^{-1}(x_k)g(x_k),$$

and the new iterate by:

$$x_{k+1} = x_k + \alpha s_k,$$

where the Newton step is given by setting $\alpha = 1$. The key observation is that taking a full Newton step, i.e., setting $\alpha = 1$, is not necessarily the best strategy when one is far away from the solution. In fact, it is not difficult to construct simple examples where Newton's method will diverge (for example, try $f(x) = \arctan(x)$).

Two schools of thought have formed on how to deal with this problem. The first is that the Newton direction is good, but that the *Newton step* is not necessarily a good one to take when far from the solution. The second school of thought is that the *Newton direction* generated by the quadratic model should only be trusted within a certain region near the current iterate. This can be argued by considering that the quadratic model that one builds by using the current information may not be representative of the function one is minimizing when one is far away from a solution. In the first case, one simply backtracks and chooses a step that is shorter than the Newton step. As such, these are called line search methods, because one simply searches along the line defined by the Newton direction until some criteria for decrease is satisfied. In the second case, one adds a constraint to the solution of the Newton equation (3) by adding a step-length restriction that says in essence that one only trusts the quadratic model within a certain region of the current approximation. These methods are known as trust region methods. Both methods work well in practice, and the choice of one method over another is usually more a question of personal preference. One small caveat is that if the function being minimized (or its derivatives) have some amount of noise in their calculation, then trust region methods appear to work slightly better.

Fortunately, there is a large amount of theory that can tell us what conditions must be met to ensure local convergence. Perhaps the easiest is to ensure that whatever step is taken, the function value at the next iterate must be at least some fraction of the decrease that would have been attained by taking a steepest descent direction (ref: steepest descent article). For more details, please check [4, 6, 7].

Another difficulty is that the Hessian of the function may become singular far away from the solution. For example, if the function is linear (or nearly linear) in some areas, then it is easy to see that the Hessian can become singular, which will lead to numerical problems in computing the step direction. In other cases, the Hessian can become indefinite, which creates its own set of problems, because the Newton direction could then fail to be a descent direction.

4.2 Derivatives

One of the main impediments in using Newton's method on real world problems is the need for not only first derivatives, but second derivatives in order to compute a step. In a typical science and engineering problem, second derivatives are usually either not available or expensive to compute. Even first derivatives are often missing in many practical problems. Since this is such a common occurrence, there are many alternatives that a user can consider. In the case of second derivatives, there is a vast literature on methods for approximating the Hessian that have proven to be quite successful in practice. There is also the option of using finite-differences to numerically approximate both first and second derivatives. Due to the large computational cost of doing this, however, it is rare to see this approach for approximating second derivatives.

Another approach is to approximate the Hessian matrix by another matrix that uses information gathered throughout the iteration process. The methods use various formulas that update the Hessian approximation, or perhaps its inverse, at each iteration. In fact, these methods can be thought of as generalizations of the familiar secant method in one dimension. A popular misconception is that these secant methods, also called quasi-Newton methods, followed the development of Newton's method. In fact, the secant methods have been shown to predate Newton's method by more than 3000 years and can be traced back to the Babylonians in the 18th-century B.C. For a fascinating historical overview of this subject see [8].

An interesting development in the last 20 years is the use of computer codes to automatically generate derivatives. These technologies, which go under the name of *automatic differentiation*, have proven to be quite useful for computing accurate and cost efficient derivatives, especially in the case where the objective function is itself the output of a computer code such as a simulation.

4.3 Computational Cost

Finally, there is the question of computational cost. While Newton's method is provably fast when close to the solution, it may take many steps before it finds itself in a region where this quadratic rate of convergence is evident. The main computational cost at each iteration is the solution of the system of linear equations necessary to compute the Newton direction, which for large dimensional problems, can be prohibitively expensive. Several approaches have been proposed, including iterative methods for the solution of the linear system of equations and truncated Newton methods, both of which attempt to solve for the Newton direction approximately in an attempt to decrease the computational cost. In addition, the use of the quasi-Newton approximations to the Hessian (or its inverse) have contributed greatly to the success of Newton methods by decreasing the overall cost of the computation of the Newton direction. Of course, all of these approaches have the disadvantages of increasing the number of iterations, but since each iteration is substantially cheaper, the overall result is a decrease in the total computational cost.

One of the more exciting areas of research today is the topic of parallel optimization. The advent of parallel computers and the trend towards multi-core and many-core computer architectures, even in desktop and laptop computers, has led to a growing need to develop new methods that take advantage of this computational power. While it would be natural to try to parallelize Newton's method, it is surprisingly difficult because there are few places where parallelization can be used effectively. One approach combines Newton's method with a parallel search direction algorithm for evaluating multiple trial points at each iteration [3]. The resulting method retains the strong convergence properties of Newton's method, while taking advantage of any computer parallelism that is available.

5 Conclusion

Newton's method is arguably the best known method for finding roots of equations or for searching for local minima. It is simple to understand, easy to program and has a fast rate of convergence. Despite these nice properties, Newton's method will often not work well on practical problems. Difficulties with ill-conditioning, computational cost, and lack of derivatives generally lead to difficulties in its use. However, many advances in all of these areas have led to robust and efficient algorithms and codes that can be easily used. In recent years, many variations on Newton's method have been proposed and used to solve large and difficult science and engineering problems. It is clear that this field is still a vibrant and exciting area for research and will remain so for many years to come.

References

[1] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1999.

[2] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[3] P. D. Hough and J. C. Meza. A class of trust-region methods for parallel optimization. *SIAM Journal on Optimization*, 13(1):264–282, 2002.

[4] C.T. Kelley. *Solving Nonlinear Equations with Newton's Method.* SIAM, Philadelphia, PA, 2003.

[5] D. G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming.* Springer, 2008.

[6] S. G. Nash and A. Sofer. *Linear and Nonlinear Programming.* McGraw-Hill, 1996.

[7] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer-Verlag, New York, 1999.

[8] J. Papakonstantinou. *Historical Development of the BFGS Secant Method and Its Characterization Properties.* PhD thesis, Rice University, 2009.