Key Concepts: Economic Computation, Part III

Brent Hickman*

Summer, 2008

1 Using Newton's Method to Find Roots of Real-Valued Functions

The intuition behind Newton's method is that finding zeros of non-linear functions is hard, but finding zeros of linear functions is trivial. Thus, for a function $f(x)$ and an initial guess x_0, we linearize the function around x_0 with a first-order Taylor approximation:

$$f(x) \approx f(x_0) + f\prime(x_0)(x - x_0).$$

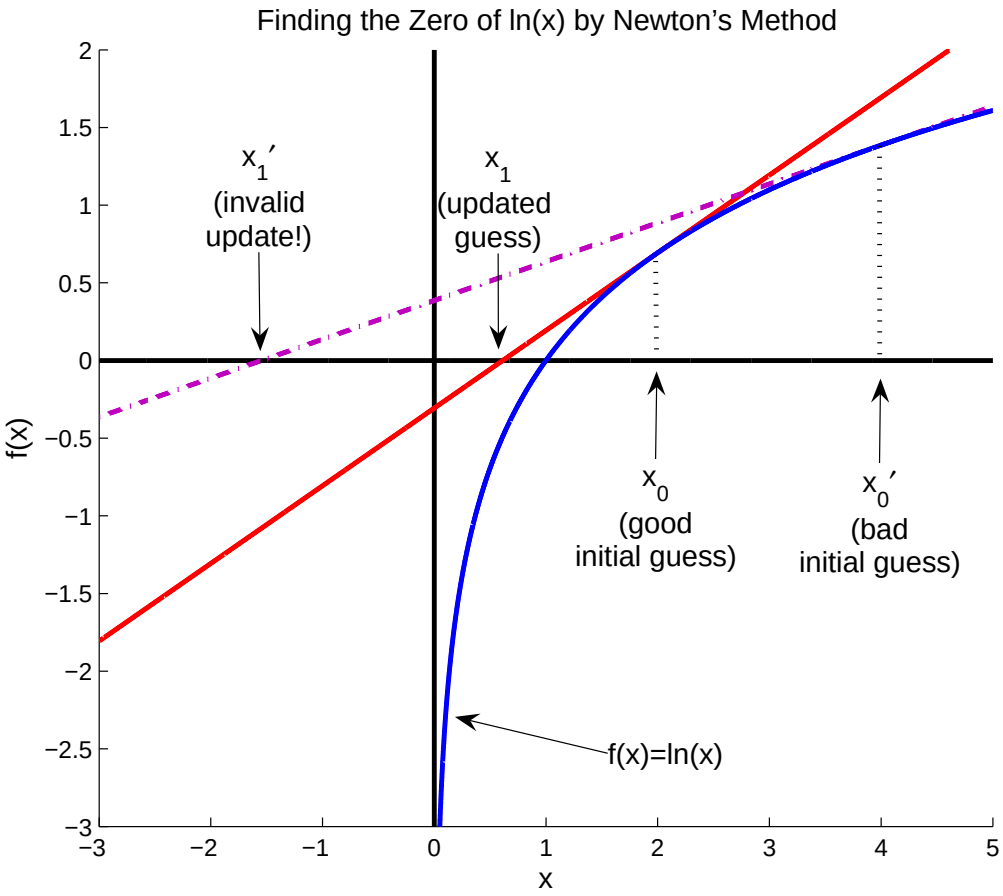
Now, setting the RHS to zero and solving for x gives the following:

$$x = x_0 - \frac{f(x_0)}{f\prime(x_0)}.$$

Taking another step, we adopt the solution to this equation as our updated guess and repeat the process. We then continue on until our successive guesses render functional values within a given tolerance of zero, at which point we adopt the final guess as our approximate zero. Figure 1 provides a visualization of Newton updating for $f(x) = \ln(x)$.

*University of Iowa Brent-Hickman@UIowa.Edu

Figure 1:


Finding the Zero of ln(x) by Newton's Method
x1′
(invalid
update!)
x1
(updated
guess)
x0
(good
initial guess)
x0′
(bad
initial guess)
f(x)=ln(x)
f(x)
x
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2.5
−3
−3
−2
−1
0
1
2
3
4
5

Newton's method is known for its rapid rate of convergence, but it does have some drawbacks. Notice that when we derived the process, we implicitly assumed that f was well behaved enough to make it work. Obviously, the function must be differentiable near its root, but if you look at Figure 1, you will see that this is not enough. As the graph shows, Newton's method may be sensitive to the quality of the initial guess. Even if you code the algorithm flawlessly, it may not work if the initial guess is not close enough to the root. The problem In Figure 1 stems from the fact that $f(x)$ is undefined for $x \leq 0$, but even if this were not the case we could still run into problems. Consider the function $f(x) = -x^2 + 4$. It has zeros at $x = -2$ and $x = -2$. Most initial guesses will cause Newton's method to converge with lightning speed, but an initial guess of $x_0 = 0$ will cause it to fall apart (why?).

One remedy to the problem of initial-guess sensitivity is to augment Newton's method with some sort of adaptive step-size control subroutine. Miranda and Fackler (2002, pp. 40-42) recommend a procedure that they call "back-stepping," which prevents Newton's method from taking a large step in the wrong direction. At each Newton iteration, the back-stepping subroutine compares the Euclidean norm of $f(x_k)$ and $f(x_{k+1})$, to see whether the updated guess produced a functional value that is closer to zero (*i*.e. one that constitutes an improvement). If no improvement is found, then the step-size from x_k to x_{k+1} is cut in half and the new updated guess becomes

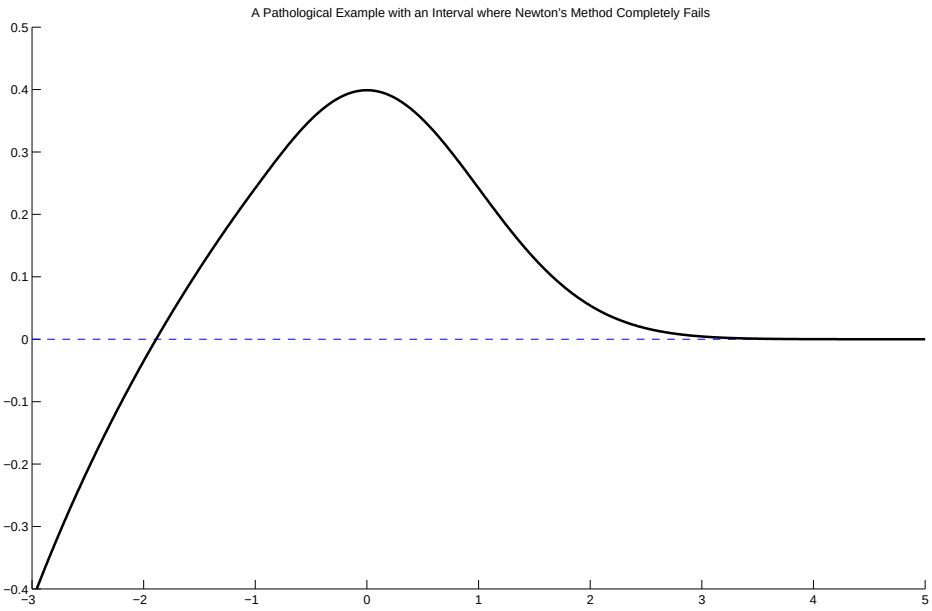
$$x_{k+1}\prime = x_k + \left(\frac{x_{k+1} - x_k}{2}\right) = x_k - \frac{f(x_k)}{f\prime(x_k)} \cdot \frac{1}{2}.$$

This new guess is also tested and revised in similar fashion before moving on to the next Newton iteration.

Exercise 1 Write a MATLAB program to find the unique root of the function $f(x) = \log(x)$ by Newton's method. Include a back-stepping routine to ensure that the program will terminate correctly for any initial guess.

It is easy to see that in the logarithmic example, back-stepping will make Newton's method perfectly robust to choice of initial guess. A solution for Exercise 1 is contained

Figure 2:



in the accompanying file `Part_III_solutions.m`. The power of Newton's method can readily be seen by experimenting with different tolerance levels and different initial guesses for this function. By incorporating adaptive step-size control, Newton's method converges to the correct root from an initial guess of 700,000,000 within 18 steps! ■

On the other hand, back-stepping also introduces additional complications for non-monotone functions, as it makes it possible for the algorithm to get stuck at a local non-zero minimum of the function $g(x) = \|f(x)\|$, where $\|\cdot\|$ denotes the Euclidean norm on the real line. Accordingly, a useful Newton routine will include two subroutines for step-size control: one for back-stepping and another for getting unstuck from local minima of $g(x)$. With a slight abuse of the notation, Miranda and Fackler (2002, pp. 41) propose a simple unsticking subroutine where, at each back-step iteration, we check whether $\|f(x_{k+1})\| > \|f(x_{k+1}\prime)\|$ and if not, then x_{k+1} is still adopted as the updated guess.

Okay, so what if we know that f and its derivative are well-defined and well-behaved everywhere; can we sit back and relax? Unfortunately the answer is no. If f is not a

monotone function, then we also have additional concerns to worry about. Indeed, it is possible to construct a non-monotone $\mathcal{C}^1$ function on the entire real line having only one zero, but where there is a continuum of initial guesses for which Newton's method will never converge, as in Figure 2 (can you tell what the interval is?). It is interesting to note that this pathological example would not even be helped by adding in a back-stepping routine as described above (why not?).

Even when dealing with a nicely behaved function for which back-stepping ensures convergence almost everywhere, non-monotone functions give rise to the possibility of multiple roots. For simple functions like polynomials, we can easily bound the number of roots, but for many functions of economic interest (*e*.g.; the first derivative of a log-likelihood function in maximum likelihood estimation) it is not always easy to determine the number of zeros beforehand. If there are multiple roots and one of them corresponds to a global maximum (as with solutions to FOCs), then one cannot be sure of the final solution until all roots have been discovered.

2 Newton's Method in Higher Dimensions

Now that we have a good understanding of the virtues and limitations of Newton's method in one dimension, we are ready to think about it in multiple dimensions, keeping in mind that similar concerns apply here too. I will just discuss it in 2-D, as the extension to higher dimensions is conceptually identical. Suppose now that we have a vector-valued function

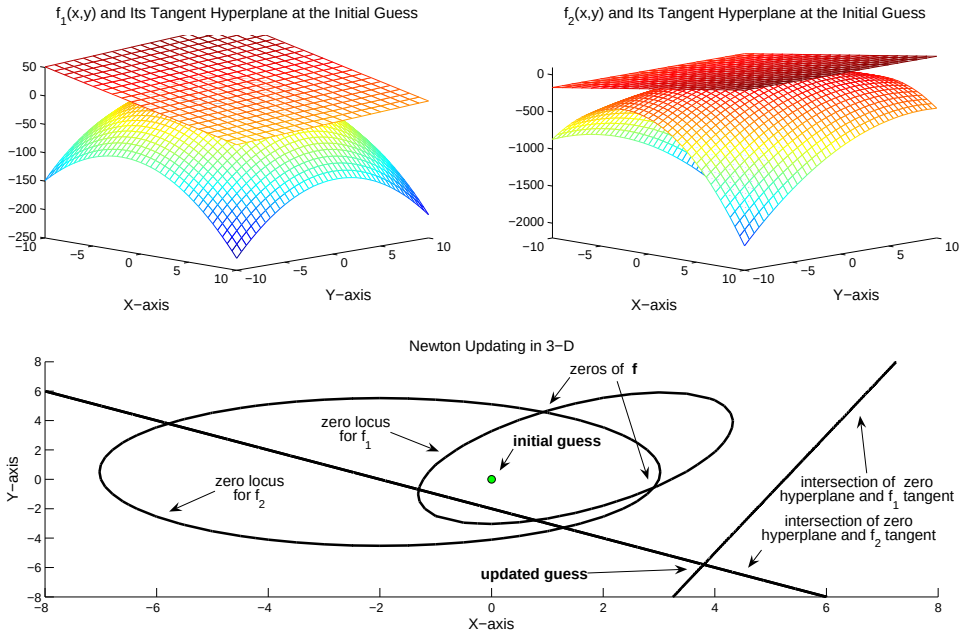
$$\mathbf{f}(x, y) = \begin{pmatrix} f_1(x, y) \\ f_2(x, y) \end{pmatrix},$$

and we would like to know which (x, y) pair will give us

$$\mathbf{f}(x, y) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Similarly as before, for initial guess (x_0, y_0), Newton's method again finds the root

Figure 3:



by using linear approximations to f_1 and f_2 in the following manner:

$$\mathbf{f}(x, y) \approx \mathbf{f}(x_0, y_0) + \mathbf{J}(x_0, y_0) \left[(x, y) - (x_0, y_0)\right]^\top,$$

where $\mathbf{J}(x_0, y_0)$ denotes the Jacobian matrix evaluated at the initial guess. Geometrically, the linear approximations to f_1 and f_2 are now tangent *p*lanes. Each of these tangent planes intersects with the zero-plane along a line, and the intersections of these two lines provides an updated guess for the zero of $\mathbf{f}$. Figure 3 provides a graphic illustration of this process. Mathematically, the guess is updated by setting the RHS of the above equation equal to $(0, 0)^\top$ and solving for (x, y). This gives us the following recursion:

$$(x_{t+1}, y_{t+1})^\top = (x_t, y_t)^\top - [\mathbf{J}(x_t, y_t)]^{-1} \mathbf{f}(x_t, y_t).$$

Again, our stopping rule judges convergence by the Euclidean norm of $\mathbf{f}(x_{k+1}, y_{k+1})$.

If Newton's method was sensitive to starting points before, it is doubly so now, as we must supply an initial guess along two dimensions. Thus, adaptive step-size control

may become even more important. Back-stepping takes similar form here and at each iteration we check whether $\|\mathbf{f}(x_{k+1}, y_{k+1})\| < \|\mathbf{f}(x_k, y_k)\|$. If not, then the adjustment from (x_k, y_k) to (x_{k+1}, y_{k+1}) is cut in half and the revised update becomes

$$(x_{k+1\prime}, y_{k+1\prime})^\top = (x_k, y_k)^\top - [\mathbf{J}(x_k, y_k)]^{-1}\mathbf{f}(x_k, y_k) \cdot \frac{1}{2}.$$

Similarly as before, one should also include an additional subroutine to avoid getting stuck at local minima of $\mathbf{g}((x, y) = \|\mathbf{f}(x, y)\|$.

3 Applications of Newton's Method in Economics

At this point, you may be wondering why an economist would ever wish to find a root of a high-dimensional vector-valued function anyway. As it turns out, this idea appears in many aspects of economics, and it generally takes two different forms: 1) solving systems of non-linear equations, and 2) optimization. There are many economic models where the equilibrium arises as the solution to a system of non-linear equations. This is common in general equilibrium models where various market-clearing conditions and individual optimality conditions come together to determine prices and allocations. Optimization is very common in econometrics, where the researcher is often faced with the task of computing a parameter estimate as the optimum of some nonlinear criterion function. Prominent examples are maximum likelihood, nonlinear least squares and GMM.

Exercise 2 Consider an economy with two price-taking firms. Firm 1 has a production technology given by

$$f(k_1, n_1) = k_1^{1/3} n_1^{2/3} + 2k_1$$

and Firm 2 has a production technology given by

$$g(k_2, n_2) = k_2^{2/3} n_2^{1/3} + n_2.$$

Suppose that the total supply of capital is 5 and the total supply of labor is 3. The

equilibrium allocation of labor and capital must equalize the marginal productivities of the inputs. Therefore, we have four equations that pin down allocations:

$$\begin{aligned}
MPK_1 = MPK_2: &\quad \frac{1}{3}\left(\frac{k_1}{n_1}\right)^{-2/3} + 2 = \frac{2}{3}\left(\frac{k_2}{n_2}\right)^{-1/3} \\
MPL_1 = MPL_2: &\quad \frac{2}{3}\left(\frac{k_1}{n_1}\right)^{1/3} = \frac{1}{3}\left(\frac{k_2}{n_2}\right)^{2/3} + 1 \\
\text{Capital market clearing:} &\quad k_1 + k_2 = 5 \\
\text{Labor market clearing:} &\quad n_1 + n_2 = 3.
\end{aligned}$$

Use Newton's method to compute the solution to the system of equations that defines the equilibrium. *HINT:* to make things easier, define $x_1 = \left(\frac{k_1}{n_1}\right)^{1/3}$ and $x_2 = \left(\frac{k_2}{n_2}\right)^{1/3}$ and solve the first two equations by Newton's method. This will then result in a simple set of 4 linear equations in 4 unknowns. Also, multiply both sides of the first re-written equation by $x_1^2 x_2$ to get rid of the negative exponents; this will improve numerical stability (can you see why?).

A solution is contained in `Part_III_solutions.m`. I first code Newton's method by hand, but without adaptive step-size control. Luckily though, the code seems to converge for most initial guesses. I then code the solution using Miranda and Fackler's `newton` function, which uses uses a safeguarded back-stepping subroutine. See the references section for a URL where Miranda and Fackler's CompEcon Toolbox can be downloaded for free. ■

Exercise 3 Write a MATLAB program to generate a random sample from the Weibull distribution. Pick some parameter values and then construct a log-likelihood function to estimate those parameters from the sample. Optimize the log-likelihood function using Newton's method and compute standard errors for the parameter estimates. *HINT:* the maximum likelihood estimator is defined as the root of the first-order condition of the log-likelihood function. The FOC is sometimes called the *score vector*, and it is a two-dimensional vector-valued function, where each element of the vector is a real-valued function of two variables. The first derivative of the score vector is a 2×2 matrix called the *Hessian*, and it is needed for Newton's method (see above). Aside from

computational speed, another advantage to Newton's method in the context of maximum likelihood is that the inverse of the Hessian (evaluated at the optimized parameter values) provides an estimator for the variance-covariance matrix for parameter estimates.

4 References:

Miranda, Mario J. and Paul L. Fackler. *A*pplied Computational Economics and Finance. MIT Press, 2002.

Miranda, Mario J. and Paul L. Fackler. CompEcon Toolbox for MATLAB to accompany *A*pplied Computational Economics and Finance. Downloaded from http://www4.ncsu.edu/∼pfackler/compecon/ on 3/8/2008.