Lecture 39: Root Finding via Newton's Method

We have studied two bracketing methods for finding zeros of a function, bisection and *regula falsi*. These methods have certain virtues (most importantly, they always converge), but it may be difficult to find an initial interval that brackets a root. Though they exhibit steady linear convergence, rather many evaluations of f may be required to attain sufficient accuracy. In this lecture, we will swap these reliable methods for a famous algorithm that often converges with amazing speed, but is more temperamental. Versions of this algorithm spring up everywhere.[†]

7.2. Newton's Method.

The idea behind the method is similar to *regula falsi*: model f with a line, and estimate the root of f by the root of that line. In *regula falsi*, this line interpolated the function values at either end of the root bracket. Newton's method is based purely on local information at the current solution estimate, x_k. Whereas the bracketing methods only required that f be continuous, we will now require that $f \in C^2(\mathbb{R})$, that is, f and its first two derivatives should be continuous. This will allow us to expand f in a Taylor series around some approximate root x_k,

$$f(x_*) = f(x_k) + f'(x_k)(x_* - x_k) + \tfrac{1}{2}f''(\xi)(x_* - x_k)^2, \tag{39.1}$$

where x_* is the exact solution, $f(x_*) = 0$, and ξ is between x_k and x_*. Ignore the error term in this series, and you have a linear model for f; i.e., $f'(x_k)$ is the slope of the line secant to f at the point x_k. Specifically,

$$0 = f(x_*) \approx f(x_k) + f'(x_k)(x_* - x_k), \quad \text{which implies} \quad x_* \approx x_k - \frac{f(x_k)}{f'(x_k)},$$

so we get an iterative method by replacing x_* in the above formulas with x_{k+1},

$$x_{k+1} := x_k - \frac{f(x_k)}{f'(x_k)}. \tag{39.2}$$

This celebrated iteration is Newton's method, implemented in the MATLAB code below.

```
 function xstar = newton(f,fprime,x0)
% Compute a root of the function f using Newton's method
% f:      a function name
% fprime: a derivative function name
% x0:     the starting guess
% Example: newton('sin','cos',3), or newton('my_f','my_fprime',1)
 maxit = 60;
 fx = feval(f,x0); x=x0; k=0;       % initialize
 fprintf(' %3d  %20.14f  %10.7e\n', k, x, fx);
 while (abs(fx) > 1e-15) & (k < maxit)
    x = x - fx/feval(fprime,x);     % Newton's method
    k = k+1;
    fx = feval(f,x);
    fprintf(' %3d  %20.14f  %10.7e\n', k, x, fx);
 end
 xstar = x;
```

[†]Richard Tapia gives a lecture titled 'If It Is Fast and Effective, It Must be Newton's Method.'

What distinguishes this iteration? For a bad starting guess x_0, it can *diverge* entirely. When it converges, the root it finds can, in some circumstances, depend sensitively on the initial guess: this is a famous source of beautiful fractal illustrations. However, for a good x_0, the convergence is usually lightning quick. Let $e_k = x_k - x_*$ be the error at the kth step. Subtract x_* from both sides of the iteration (39.2) to obtain a recurrence for the error,

$$e_{k+1} = e_k - \frac{f(x_k)}{f'(x_k)}.$$

The Taylor expansion of $f(x_*)$ about the point x_k given in (39.1) gives

$$0 = f(x_k) - f'(x_k)e_k + \tfrac{1}{2}f''(\xi)e_k^2.$$

Solving this equation for $f(x_k)$ and substituting that formula into the expression for e_{k+1} we just derived, we obtain

$$e_{k+1} = e_k - \frac{f'(x_k)e_k + \frac{1}{2}f''(\xi)e_k^2}{f'(x_k)} = -\frac{f''(\xi)e_k^2}{2f'(x_k)}.$$

Supposing that x_* is a simple root, so that $f'(x_*) \neq 0$, the above analysis suggests that when x_k is near x_*,

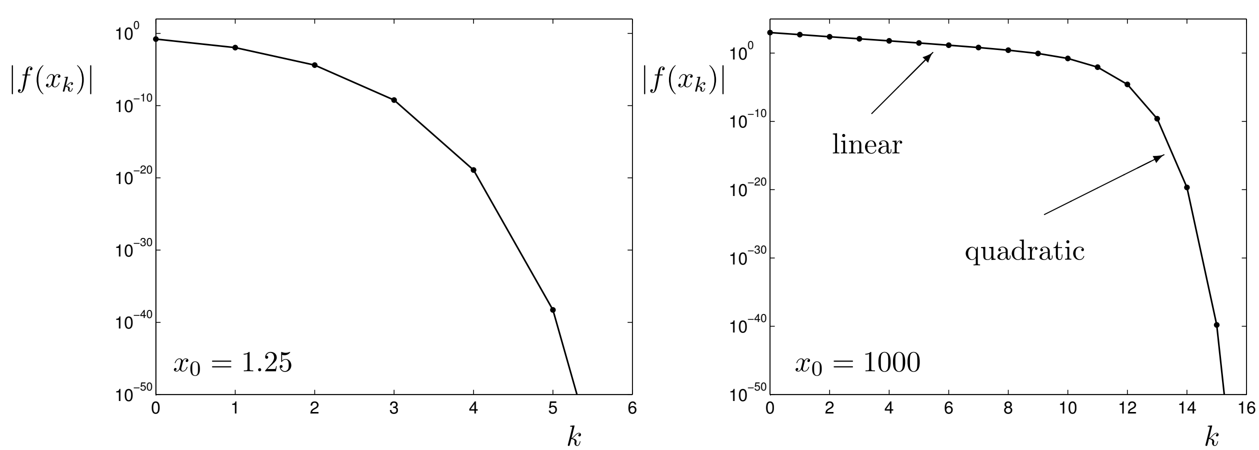
$$|e_{k+1}| \leq C|e_k|^2$$

for some constant C independent of k. This is *quadratic convergence*, and it roughly means that you *double the number of correct digits* at each iteration. Compare this to bisection, where

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

meaning that the error was halved at each step. Significantly, Newton's method will often exhibit a transient period of linear convergence while it gets sufficiently close to the answer, but once in a region of quadratic convergence, full machine precision is attained in just a couple more iterations.

The following example approximates the zero of $f(x) = x^2 - 2$, i.e., $x_* = \sqrt{2}$. As initial guesses, we choose $x_0 = 1.25$ (left), which gives us very rapid convergence, and $x_0 = 1000$ (right), which is a ridiculous estimate of $\sqrt{2}$, but illustrates the linear phase of convergence that can precede superlinear convergence when x_0 is far from x_*.



The table below shows the iterates for $x_0 = 1000$, computed exact arithmetic in Mathematica, and displayed here to more than eighty digits. This is a bit excessive: in the floating point arithmetic we have used all semester, we can only expect to get 15 or 16 digits of accuracy in the best case. It is worth looking at all these digits to get a better appreciation of the quadratic convergence. Once we are in the quadratic regime, notice the characteristic doubling of the number of correct digits (underlined) at each iteration.

```
k                                              x_k
-----------------------------------------------------------------------------------------------
0     1000.00000000000000000000000000000000000000000000000000000000000000000000000000000000000
1      500.00100000000000000000000000000000000000000000000000000000000000000000000000000000000
2      250.00249999600000799998400003199993600012799974400051199897600204799590400819198361603
3      125.00524995800046799458406305526512856598014823595622393441695800477446685799463896484
4       62.51062464301703314888691358403320464529759944325744566631164600631017391478309761341
5       31.27130960206219455596422358771700548374565801842332086536365236578278080406153827364
6       15.66763299486836640030755527100281652065100159710324459452581543767403479921834012248
7        7.89764234785635806719051360934236238116968365174167025116461034160777628217364960111
8        4.07544124051949892088798573387067133352991149961309267159333980191548308075360961862
9        2.28309282439255383986306690358177946144339233634377781606055538481637200759555376236
10       1.57954875240601536527547001727498935127463981776389016188975791363939586265860323251
11       1.42286657957866825091209683856309818309310929428763928162890934673847036238184992693
12       1.41423987359153062319364616441120035182529489347860126716395746896392690040774558375
13       1.41421356261784851265589000359174396632207628548968908242398944391615436335625360056
14       1.41421356237309504882286807775717118221418114729423116637254804377031332440406155716
15       1.41421356237309504880168872420969807856983046705949994860439640079460765093858305190
16       1.41421356237309504880168872420969807856967187537694807317667973799073247846210704774

exact:   1.41421356237309504880168872420969807856967187537694807317667973799073247846210703885038753...
```

7.2.1. Convergence analysis.

We have already performed a simple analysis of Newton's method to gain an appreciation for the quadratic convergence rate. For a broader perspective, we shall now put Newton's method into a more general framework, so that the accompanying analysis will allow us to understand simpler iterations like the 'constant slope method:'

$$x_{k+1} = x_k - \alpha f(x_k)$$

for some constant α (which could approximate $1/f'(x_*)$, for example). We begin by formalizing our notion of the rate of convergence.

Definition. A root-finding algorithm is *pth-order convergent* if

$$|e_{k+1}| \le C|e_k|^p$$

for some $p \ge 1$ and positive constant C. If $p = 1$, then $C < 1$ is necessary for convergence, and C is called the *linear convergence rate.*

Newton's method is second-order convergent (i.e., it converges quadratically) for $f \in C^2(\mathbb{R})$ when $f'(x_*) \ne 0$ and x_0 is sufficiently close to x_*. Bisection is linearly convergent for $f \in C[a_0, b_0]$ with rate $C = 1/2$.

Functional iteration. One can analyze Newton's method and its variants through the following general framework.[‡] Consider iterations of the form

$$x_{k+1} = \Phi(x_k),$$

for some iteration function Φ. For example, for Newton's method

$$\Phi(x) = x - \frac{f(x)}{f'(x)}.$$

If the starting guess is an exact root, $x_0 = x_*$, the method should be smart enough to return $x_1 = x_*$. Thus the root x_* is a *fixed point* of Φ, i.e.,

$$x_* = \Phi(x_*).$$

We seek an expression for the error $e_{k+1} = x_{k+1} - x_*$ in terms of e_k and properties of Φ. Assume, for example, that $\Phi(x) \in C^2(\mathbb{R})$, so that we can write the Taylor series for Φ expanded about x_*:

$$\begin{aligned} x_{k+1} = \Phi(x_k) &= \Phi(x_*) + (x_k - x_*)\Phi'(x_*) + \tfrac{1}{2}(x_k - x_*)^2\Phi''(\xi) \\ &= x_* + (x_k - x_*)\Phi'(x_*) + \tfrac{1}{2}(x_k - x_*)^2\Phi''(\xi) \end{aligned}$$

for some ξ between x_k and x_*. From this we obtain an expression for the errors:

$$e_{k+1} = e_k\Phi'(x_*) + \tfrac{1}{2}e_k^2\Phi''(\xi).$$

Convergence analysis is reduced to the study of $\Phi'(x_*)$, $\Phi''(x_*)$, etc.

Example: Newton's method. For Newton's method

$$\Phi(x) = x - \frac{f(x)}{f'(x)},$$

so the quotient rule gives

$$\Phi'(x) = 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} = \frac{f(x)f''(x)}{f'(x)^2}.$$

Provided x_* is a simple root so that $f'(x_*) \neq 0$ (and supposing $f \in C^2(\mathbb{R})$), we have $\Phi'(x_*) = 0$, and thus

$$e_{k+1} = \tfrac{1}{2}e_k^2\Phi''(\xi),$$

and hence we again see quadratic convergence provided x_k is sufficiently close to x_*.

What happens when $f'(x_*) = 0$? If x_* is a multiple root, we might worry that Newton's method might have trouble converging, since we are dividing $f(x_k)$ by $f'(x_k)$, and both quantities are nearing zero as $x_k \to x_*$. This general convergence framework allows us to investigate this situation more precisely. We wish to understand

$$\lim_{x\to x_*} \Phi'(x) = \lim_{x\to x_*} \frac{f(x)f''(x)}{f'(x)^2}.$$

[‡]For further details on this standard approach, see G. W. Stewart, *Afternotes on Numerical Analysis*, §§2–4; J. Stoer & R. Bulirsch, *Introduction to Numerical Analysis*, 2nd ed., §5.2; L. W. Johnson and R. D. Riess, *Numerical Analysis*, second ed., §4.3.

This limit has the indeterminate form 0/0. Assuming sufficient differentiability, we can invoke l'Hôpital's rule:

$$\lim_{x\to x_*} \frac{f(x)f''(x)}{f'(x)^2} = \lim_{x\to x_*} \frac{f'(x)f''(x)+f(x)f'''(x)}{2f'(x)f''(x)},$$

but this is also of the indeterminate form 0/0 when $f'(x_*)=0$. Again using l'Hôpital's rule and now assuming $f''(x_*)\neq 0$,

$$\lim_{x\to x_*} \frac{f(x)f''(x)}{f'(x)^2} = \lim_{x\to x_*} \frac{f''(x)^2+2f'(x)f'''(x)+f(x)f^{(iv)}(x)}{2(f'(x)f'''(x)+f''(x)^2)} = \lim_{x\to x_*} \frac{f''(x)^2}{2f''(x)^2} = \frac{1}{2}.$$

Thus, Newton's method converges locally to a double root according to

$$e_{k+1} = \tfrac{1}{2}e_k + O(e_k^2).$$

Note that this is linear convergence *at the same rate as bisection*! If x_* has multiplicity exceeding two, then $f''(x_*)=0$ and further analysis is required. One would find that the rate remains linear, and gets even slower. The slow convergence of Newton's method for multiple roots is exacerbated by the chronic ill-conditioning of such roots. Let us summarize what might seem to be a paradoxical situation: the more 'copies' of root there are present, the more difficult that root is to find!