4
Nonlinear Equations

LECTURE 27: *Introduction to Nonlinear Equations*
LECTURE 28: *Bracketing Algorithms*

THE SOLUTION OF NONLINEAR EQUATIONS has been a motivating challenge throughout the history of numerical analysis. We opened these notes with mention of Kepler's equation, $M = E - e \sin E$, where M and e are known, and E is sought. One can view this E as the solution of the equation $f(E) = 0$, where

$$f(x) = M - x + e \sin(x).$$

This is a simple equation in one variable, $x \in \mathbb{R}$, and it turns out that it is not particularly difficult to solve. Other examples are complicated by nastier nonlinearities, multiple solutions (in which case one might like to find them all), ill-conditioned zeros (where $f(x) \approx 0$ for x far from the true zeros of f), solutions in the complex plane, and expensive f evaluations.

Optimization is closely allied to the solution of nonlinear equations, since one finds extrema of $F : \mathbb{R} \to \mathbb{R}$ by solving

$$F'(x) = 0.$$

When $F : \mathbb{R}^n \to \mathbb{R}$, one seeks $\mathbf{x} \in \mathbb{R}^n$ that solves

$$\nabla F(\mathbf{x}) = \mathbf{0},$$

a nonlinear system of n equations in n variables. Handling $n > 1$ variables leads to more sophisticated theory and algorithms. Optimization problems become more difficult when additional constraints are imposed upon $\mathbf{x} \in \mathbb{R}^n$,

$$\min_{\mathbf{x} \in S} F(\mathbf{x}),$$

where $\mathcal{S}$ is the set of feasible solutions (e.g., vectors with nonnegative entries). When F is linear in $\mathbf{x}$ and $\mathbf{x}$ is constrained by simple inequalities, one has the important field of *linear programming*. When the set $\mathcal{S}$ constrains $\mathbf{x}$ to take discrete values (e.g., integers), the optimization problem becomes exceptionally difficult. Such *combinatorial optimization* or *integer programming* problems connect closely to the study of NP-hard problems in computer science.

In this course, we only have time to look at a few of the simplest algorithms for solving nonlinear equations. These will give some brief introduction to some of the over-arching themes in this important field. Further study is highly recommended!

4.1 Bracketing Algorithms for Root Finding

Given a function $f : \mathbb{R} \to \mathbb{R}$, we seek a point $x_* \in \mathbb{R}$ such that $f(x_*) = 0$. This x_* is called a *root* of the equation $f(x) = 0$, or simply a *zero* of f. At first, we only require that f be continuous a interval $[a, b]$ of the real line, $f \in C[a, b]$, and that this interval contains the root of interest. The function f could have many different roots; we will only look for one. In practice, f could be quite complicated (e.g., evaluation of a parameter-dependent integral or differential equation) that is expensive to evaluate (e.g., requiring minutes, hours, ...), so we seek algorithms that will produce a solution that is accurate to high precision while keeping evaluations of f to a minimum.

In some applications, like polynomial root-finding, we know f has multiple zeros, and we might want to find all of them. This particular case is complicated by the fact that the zeros could be located in the complex plane, $x_* \in \mathbb{C}$.

The first algorithms we study require the user to specify a finite interval $[a_0, b_0]$, called a *bracket*, such that $f(a_0)$ and $f(b_0)$ differ in sign, $f(a_0)f(b_0) < 0$. Since f is continuous, the intermediate value theorem guarantees that f has at least one root x_* in the bracket, $x_* \in (a_0, b_0)$.

4.1.1 Bisection

Given a bracket $[a, b]$ for which f takes opposite sign at a and b, the simplest technique for finding x_* is the *bisection* algorithm:

For $k = 0, 1, 2, \ldots$

1. Compute $f(c_k)$ for $c_k = \frac{1}{2}(a_k + b_k)$.
2. If $f(c_k) = 0$, exit; otherwise, repeat with
$$[a_{k+1}, b_{k+1}] := \begin{cases} [a_k, c_k], & \text{if } f(a_k)f(c_k) < 0; \\ [c_k, b_k], & \text{if } f(c_k)f(b_k) < 0. \end{cases}$$
3. Stop when the interval $b_{k+1} - a_{k+1}$ is sufficiently small, or if $f(c_k) = 0$.

How does this method converge? Not bad for such a simple idea. At

the kth stage, there must be a root in the interval $[a_k, b_k]$. Take $c_k = \frac{1}{2}(a_k + b_k)$ as the next estimate to x_*, giving the error $e_k = c_k - x_*$. The *worst* possible error would occur if x_* was close to a_k or b_k, half the bracket's width away from c_k, and hence $|e_k| = |c_k - x_*| \leq \frac{1}{2}(b_k - a_k) = 2^{-k-1}(b_0 - a_0)$.

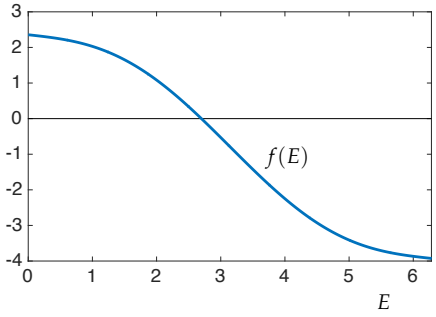
Theorem 4.9. Given a bracket $[a, b] \subset \mathbb{R}$ for which $f(a)f(b) < 0$, there exists $x_* \in [a, b]$ such that $f(x_*) = 0$ and the bisection point c_k satisfies

$$|c_k - x_*| \leq \frac{b-a}{2^{k+1}}.$$

We say this iteration *converges linearly* (the log of the error is bounded by a straight line when plotted against iteration count – see the next example) with rate $\rho = 1/2$. Practically, this means that the error is cut in half at each iteration, *independent of the behavior of* f. Reduction of the initial bracket width by ten orders of magnitude would require roughly $\log_2 10^{10} \approx 33$ iterations. If f is fast to evaluate, this convergence will be pretty quick; moreover, since the algorithm only relies on our ability to compute the sign of $f(x)$ accurately, the algorithm is robust to strange behavior in f (such as local minima).

Example 4.4. Kepler's equation is among the most historically important nonlinear equations, playing a pivotal role in two-body celestial mechanics. Given orbital parameters $e \in [0, 1)$ (the *eccentricity* of the elliptical orbit) and $M \in [0, 2\pi]$ (the *mean anomaly*), find $E \in [0, 2\pi]$ (the *eccentric anomaly*) such that

$$M = E - e \sin E.$$



Cast this as the root-finding problem $f(E) = 0$, where

$$f(E) = M - E + e \sin E.$$

In this example we set $e = 0.8$ and $M = 3\pi/4$, yielding the function shown in the margin. Judging from this plot, the desired root E_* falls in the interval $[2, 3]$. Using the initial bracket $[a, b] = [2, 3]$, the bisection method converges as steadily as expected, cutting the error in half at every step. Figure 4.2 shows the convergence to the exact root $E_* = 2.69889638445749738544\ldots$.

4.1.2 *Regula Falsi*

A simple adjustment to bisection can often yield much quicker convergence. The name of the resulting algorithm, *regula falsi* (literally 'false rule') hints at the technique. As with bisection, begin with an interval $[a_0, b_0] \subset \mathbb{R}$ such that $f(a_0)f(b_0) < 0$. The goal is to be

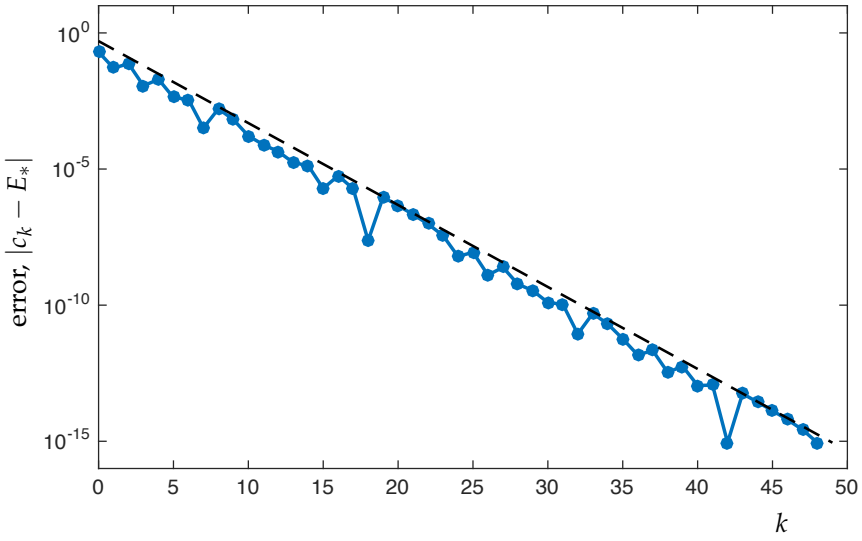


Figure 4.1: The convergence of the bisection method for solving Kepler's equation (Example 4.4) with initial bracket $[a, b] = [2, 3]$. Use the midpoint c_k of the kth bracket as an estimate of the root E_*. The dashed line shows the error bound $|c_k - E_*| \leq (b - a)/2^{k+1}$.

more sophisticated about the choice of the root estimate $c_k \in (a_k, b_k)$. Instead of simply choosing the middle point of the bracket as in bisection, we approximate f with the line $p_k \in \mathcal{P}_1$ that interpolates $(a_k, f(a_k))$ and $(b_k, f(b_k))$, so that $p_k(a_k) = f(a_k)$ and $p(b_k) = f(b_k)$. This unique polynomial is given (in the Newton form) by

$$p_k(x) = f(a_k) + \frac{f(b_k) - f(a_k)}{b_k - a_k}(x - a_k).$$

Now approximate the zero of f in $[a_k, b_k]$ by the zero of the linear model p_k:

$$c_k = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}.$$

The algorithm then takes the following form:

For $k = 0, 1, 2, \ldots$

1. Compute $f(c_k)$ for $c_k = \dfrac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}$.
2. If $f(c_k) = 0$, exit; otherwise, repeat with
$$[a_{k+1}, b_{k+1}] := \begin{cases} [a_k, c_k], & \text{if } f(a_k) f(c_k) < 0; \\ [c_k, b_k], & \text{if } f(c_k) f(b_k) < 0. \end{cases}$$
3. Stop when $f(c_k)$ is sufficiently small, or the maximum number of iterations is exceeded.

Note that only Step 3 differs significantly from the bisection method. The former algorithm forces the bracket width $b_k - a_k$ to zero as it homes in on the root. In contrast, there is no mechanism in the regula falsi algorithm to drive the bracket width to zero: it will still always converge (in exact arithmetic) even though the bracket length does

not typically decrease to zero. Analysis of *regula falsi* is more complicated than the trivial bisection analysis; we give a convergence proof only for a special case. We will use the opportunity to introduce the notion of *convex functions*, a fundamental idea in optimization theory, especially for problems in higher dimensions.

Definition 4.2. Let $[a, b]$ be a finite interval of $\mathbb{R}$. Then a function $f \in C[a, b]$ is *convex* provided that for all $x, y \in [a, b]$ and $\lambda \in [0, 1]$,

$$f\big(\lambda x + (1 - \lambda)y\big) \leq \lambda f(x) + (1 - \lambda) f(y).$$

At first sight convexity might seem to be an abstract concept. However, an easy sufficient condition will be more familiar.

Lemma 4.1. If $f \in C^2[a, b]$ and $f''(x) \geq 0$ for all $x \in [a, b]$, then f is convex.

Proof. (See Rockafeller, *Convex Analysis*, Theorem 4.4.)

First note that f' is monotonically increasing on $[a, b]$. To see this, use the Fundamental Theorem of Calculus to write, for any $\xi \in [a, b]$,

$$f'(\xi) - f'(a) = \int_a^\xi f''(s)\, \mathrm{d}s.$$

The integrand on the right is nonnegative, so as ξ increases, the value of the integral cannot decrease. Since $f'(a)$ is a constant,

$$f'(\xi) = f'(a) + \int_a^\xi f''(s)\, \mathrm{d}s$$

is a monotonically increasing function of $\xi \in [a, b]$.

Now fix $\lambda \in [0, 1]$ and $z := \lambda x + (1 - \lambda)y$. Again use the Fundamental Theorem of Calculus to write

Such a z is a *convex combination* of x and y.

$$f(z) - f(x) = \int_x^z f'(s)\, \mathrm{d}s.$$

Now use monotonicity of f' to get the upper bound

$$\begin{aligned} f(z) &= f(x) + \int_x^z f'(s)\, \mathrm{d}s \\ &\leq f(x) + f'(z)(z - x). \end{aligned} \tag{4.1}$$

Another upper bound follows similarly. Write

$$f(y) - f(z) = \int_y^z f'(s)\, \mathrm{d}s$$

and again use monotonicity of the derivative (with $z \leq y$) to obtain

$$\begin{aligned} f(z) &= f(y) - \int_z^y f'(s)\, \mathrm{d}s \\ &\leq f(y) - f'(z)(y - z). \end{aligned} \tag{4.2}$$

Now preumultiply (4.1) by λ and (4.2) by $1-\lambda$ and add to obtain:

$$f(z) = \lambda f(z) + (1-\lambda) f(z)$$

$$\leq \lambda f(x) + (1-\lambda) f(y) + \lambda f'(z)(z-x) - (1-\lambda) f'(z)(y-z). \tag{4.3}$$

Notice that

$$\lambda(z-x) - (1-\lambda)(y-z) = -\lambda x - (1-\lambda)y + z = 0,$$

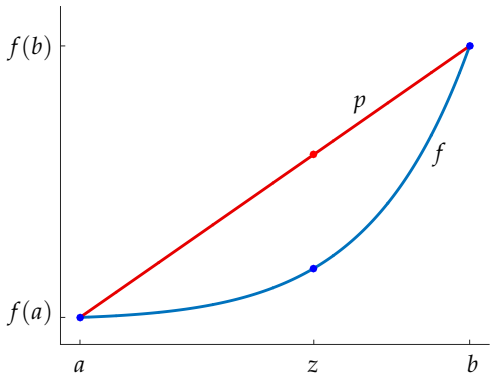
and hence (4.3) reduces to

$$f(z) \leq \lambda f(x) + (1-\lambda) f(y) + \lambda f'(z)(z-x).$$

Thus $f(x) \geq 0$ for all $x \in [a,b]$ proves that f is convex. ∎

Convexity implies that the linear interpolant will always be located above the function, as sketched on the right. The linear interpolant to f at $x = a$ and $x = b$ is

$$p(x) = f(a) + \frac{f(b)-f(a)}{b-a}(x-a).$$



Any $z \in [a,b]$ can be written as $z = \lambda a + (1-\lambda)b$, so

$$\begin{aligned} p(z) &= f(a) + \frac{f(b)-f(a)}{b-a}\big(\lambda a + (1-\lambda)b - a\big) \\ &= f(a) + \frac{f(b)-f(a)}{b-a}(1-\lambda)(b-a) \\ &= \lambda f(a) + (1-\lambda) f(b) \\ &\geq f\big(\lambda a + (1-\lambda)b\big) \\ &= f(z), \end{aligned}$$

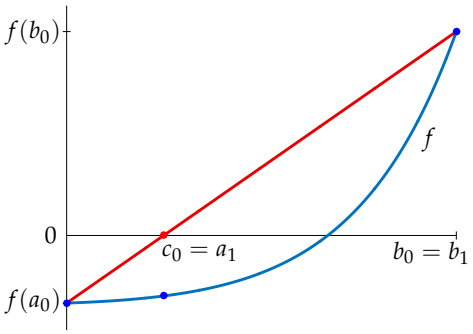
where the inequality follows from convexity of f.

This observation plays a key role in the next proof, which guarantees convergence of regula falsi for convex functions.

Theorem 4.10. Suppose f is a convex function on $[a_0, b_0]$ with $a_0 < b_0$ and $f(a_0) < 0 < f(b_0)$. Then *regula falsi* converges: either $f(c_k) = 0$ for some $k \geq 0$, or $c_k \to x_* \in [a_0, b_0]$ with $f(x_*) = 0$.

Proof. (See Stoer & Bulirsch, *Introduction to Numerical Analysis*, 2nd ed., §5.9.)

The argument preceding the proof ensures that the linear interpolant p to f at a_0 and b_0 satisfies $p(x) \geq f(x)$ for all $x \in [a_0, b_0]$. Since c_0 is selected so that $p_0(c_0) = 0$, it follows that $f(c_0) \leq 0$, so the new bracket will be $[a_1, b_1] = [c_0, b_0]$.



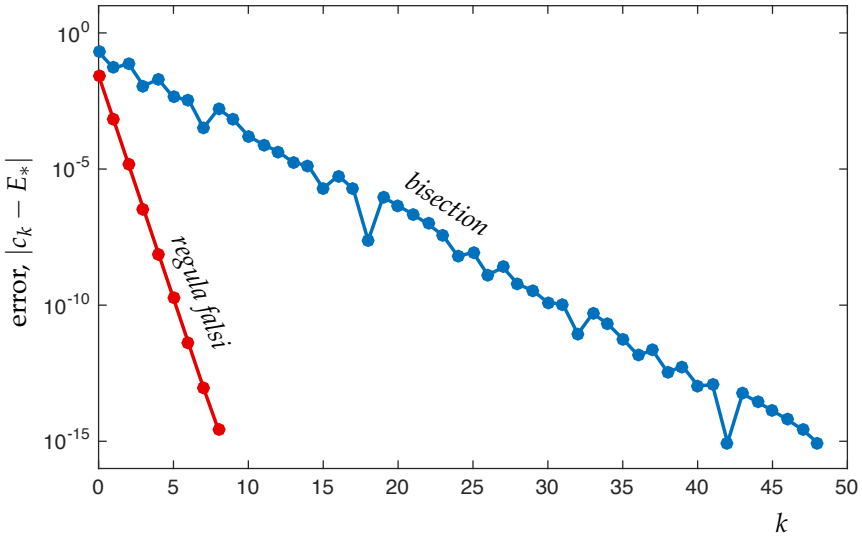


Figure 4.2: The convergence of the bisection and regula falsi methods for solving Kepler's equation (Example 4.4) with initial bracket $[a, b] = [2, 3]$. The root c_k in regula falsi converges to E_* much more rapidly than the midpoint c_k of the bisection brackets.

If $f(c_0) = 0$, the method has exactly found a root, and stops; otherwise,

$$f(a_1) = f(c_0) < 0 < f(b_0) = f(b_1).$$

Now use the convexity of f on the subinterval $[a_1, b_1]$ to iterate this argument, first showing that $[a_2, b_2] = [c_1, b_1]$, and then, in general,

$$[a_{k+1}, b_{k+1}] = [c_k, b_k].$$

Notice that $c_k > a_k = c_{k-1}$. If the algorithm never finds an exact root, it forms a sequence of estimates $\{c_k\}$ that is monotonically increasing and bounded above by b_0. The monotone convergence theorem in real analysis ensures that any bounded monotone sequence of real numbers must converge to a limit. Thus, $\lim_{k\to\infty} c_k = \gamma$ with $f(\gamma) \leq 0$, and this γ must be a fixed point of the regula falsi iteration, i.e.,

For a proof, see Rudin, *Principles of Mathematical Analysis*, Theorem 3.14.

$$\gamma = \frac{\gamma f(b_0) - b_0 f(\gamma)}{f(b_0) - f(\gamma)}.$$

Rearrange this expression to get $(\gamma - b_0)f(\gamma) = 0$. Since $f(b_0) > 0$, conclude that $\gamma \neq b_0$, and hence $f(\gamma) = 0$. Thus, *regula falsi* converges for convex functions. ∎

Example 4.5. Now apply the regula falsi method to the same version of Kepler's equation we solved with bisection in Example 4.4. Figure **??** compares the convergence of regula falsi to bisection, both with the same initial bracket $[2, 3]$. About 9 steps of regula falsi delivers the same accuracy obtained by more than 40 steps of bisection. For nice problems like this one, regula falsi is much more efficient.

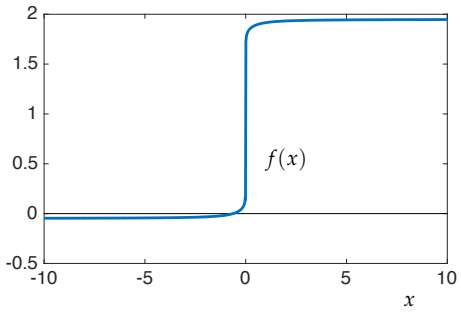
Is *regula falsi* always superior to bisection? If f has a zero, one can alway rig a bracket around it so that the zero x_* is exactly at its

midpoint, $x_* = \frac{1}{2}(a_0 + b_0)$, giving convergence of bisection in a single iteration. For most such functions, the first regula falsi iterate is different, and not a root of our function. Thus bisection can beat regula falsi, but such examples seem contrived, depending essentially on a lucky bracket. Can one construct more compelling examples?

Example 4.6. Lest Example 4.5 suggest that regula falsi is always superior to bisection, we now consider a function for which regula falsi converges very slowly. Sketch out a few sample functions. You will soon see how to design an f such that the root c_k of the linear approximation converges slowly toward x_* as k increases. The function should be relatively flat and small in magnitude in a large region near the root. One such example is

$$f(x) = \text{sign}(\tan^{-1}(x)) \left| \frac{2}{\pi} \tan^{-1}(x) \right|^{1/20} + \frac{19}{20},$$

which has a single root at $x_* \approx -0.6312881\ldots$. This function is illustrated in the margin. Figure 4.3 compares convergence of bisection and regula falsi for this f with the initial bracket $[-10, 10]$. The small value of f at the left end of the bracket ensures that $[a_1, b_1] = [c_0, b]$ will be almost as large as the initial bracket $[a, b]$.



4.1.3 *Accuracy*

Here we have assumed that we calculate $f(x)$ to perfect accuracy, an unrealistic expectation on a computer. If we attempt to compute x_* to very high accuracy, we will eventually experience errors due to inaccuracies in our function $f(x)$. For example, $f(x)$ may come from approximating the solution to a differential equation, were there

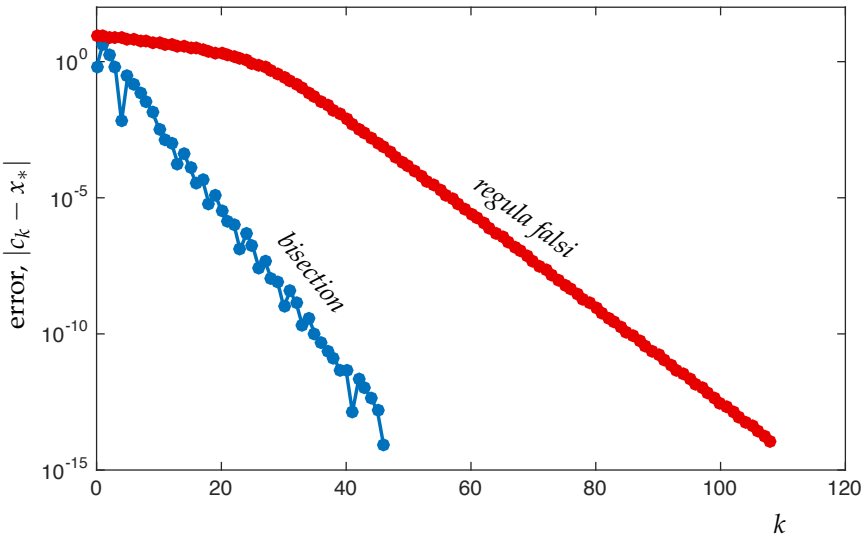
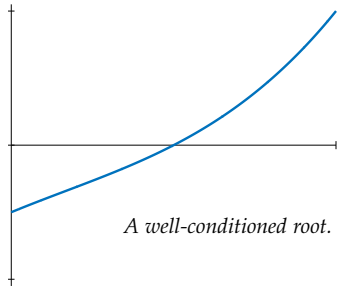


Figure 4.3: The convergence of the bisection and regula falsi methods for the function in Example 4.6 with initial bracket $[a, b] = [-10, 10]$. The root c_k in regula falsi converges very slowly to the right as k increases, due to the small value of f on the left end of the bracket. In contrast, bisection merrily cuts the error in half at each step, affected only by the sign of f, not its magnitude.

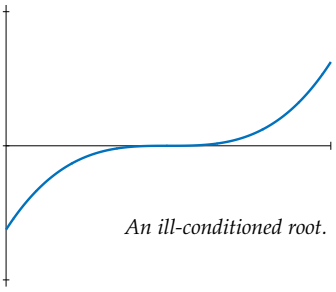
is some approximation error we must be concerned about; more generally, the accuracy of f will be limited by the computer's floating point arithmetic. One must also be cautious of subtracting one like quantity from another (as in construction of c_k in both algorithms), which can give rise to *catastrophic cancellation*.

4.1.4 *Conditioning*

When $|f'(x_0)| \gg 0$, the desired root is easy to pick out. In cases where $f'(x_0) \approx 0$, the root will be *ill-conditioned*, and it can be difficult to locate. This is the case, for example, when x_0 is a multiple root of f. (You may find it strange that the more copies of a root you have, the more difficult it can be to compute it!) In such cases bisection has the advantage that it only depends on the *sign* of f.



A well-conditioned root.



An ill-conditioned root.

4.1.5 *Deflation*

What is one to do if multiple distinct roots are required? One approach is to choose a new initial bracket that omits all known roots. Another technique, though numerically fragile, is to work with $\widehat{f}(x) := f(x)/(x - x_0)$, where x_0 is the previously computed root.