Bisection

If \( f \) is continuous on \([a, b]\), and \( f(a)f(b) < 0 \), then the intermediate value theorem guarantees that there is at least one \( x^* \in (a, b) \) for which \( f(x^*) = 0 \). If you had to approximate such an \( x^* \), what point would you use? That is, knowing only that \( x^* \in [a, b] \), what value \( x \) minimizes the maximum possible error \( |x^* - x| \)? It’s the midpoint \( p = a + (b - a)/2 \) which satisfies this minimax property.

If \( f(a)f(p) < 0 \), then there must be a root of \( f \) in \((a, p)\), otherwise there must be one in \((p, b)\). This observation is the basis for the method of bisection. Let \( a_0 = a \) and \( b_0 = b \) be such that \( f(a_0)f(b_0) < 0 \). For a given \( a_i \) and \( b_i \), define

\[
p = a_i + (b_i - a_i)/2.
\]

If \( f(p) = 0 \) we are done; if not, define \( a_{i+1} \) and \( b_{i+1} \) by

\[
\begin{align*}
\text{if } f(a)f(p) < 0 & \quad a_{i+1} = a_i, \quad b_{i+1} = p \\
\text{else} & \quad a_{i+1} = p, \quad b_{i+1} = b_i
\end{align*}
\]

The new interval is half as big as the previous, and it contains a root of \( f \). This process defines a sequence of intervals \([a_i, b_i]\) of length \( b_i - a_i = (b - a)/2^i \), each of which contains a root of \( f \). If \( x^* \) is a root of \( f \) in \([a_i, b_i]\), then \( p = a_i + (b_i - a_i)/2 \) satisfies

\[
|x^* - p| \leq (b - a)/2^{i+1}.
\]

It is rare for an algorithm to provide a bound on the error (as bisection does), and this is one of its most appealing properties. The error bound is strictly monotone decreasing, guaranteeing that for any tolerance \( \tau > 0 \), the absolute error will satisfy \( |x^* - p| \leq \tau \) after at most

\[
N = \lceil \log_2 \left( \frac{b-a}{\tau} \right) \rceil
\]

steps. This certainty comes at the price of (i) requiring, a priori, an interval \([a, b]\) for which \( f(a)f(b) < 0 \), and (ii) slowness. Since we haven’t seen other methods yet, it may not be clear that this method is slow, but you can see that the algorithm is terribly near-sighted: the only information about \( f(x) \) that is used is its sign.

The floating point implementation of bisection is relatively simple. The accuracy of the computed value of \( f(p) \) is rather less important here than in other methods, because here the quantity \( \text{sign}(f(p)) \) is all that is needed, which is always well conditioned away from \( f(p) = 0 \). When it is difficult to determine \( \text{sign}(f(p)) \) it is safe to say (up to rounding errors in the evaluation of \( f \)) that \( p \) is a root. We do find that we risk underflow in the evaluation of \( f(a)f(p) \), and thus a careful implementation would use \( \text{sign}(f(a))\text{sign}(f(p)) \) instead. Of course we should remember to save our latest value of \( \text{sign}(f(a)) \), so we do not need to recompute it in the next iteration. Another consideration is the evaluation of the midpoint of \([a, b]\), which, in floating point should be evaluated as \( a + (b - a)/2 \) rather than \((a + b)/2 \) (Why?).