Root-finding: bisection and the fixed point iteration

1.1 Bisection method

Bisection method is the most straightforward and simple method for computing the root (zero) of a **continuous** function. If the interval [*a*,*b*] "brackets" our root, say $f(a) < 0$ and $f(b) > 0$ (or *vice versa*), then by the Intermediate Value Theorem this interval must contain our root *r*. To find it, start by checking the point in the middle, and assign it to *a* or *b* depending on the sign of $f(c_n)$. Then, keep repeating these steps. Algorithm can be summed up as follows:

- Pick as our next approximation the mid-point between *a* and *b* values: $c_n = (a + b) / 2$.
- If $f(c_n)$ has the same sign as $f(a)$, we set $a = c_n$,
- If $f(c_n)$ has the same sign as $f(b)$, we set $b = c_n$.
- If $f(c_n) = 0$, we can terminate the iteration (this situation is unlikely though).
- Repeat these steps until the bracketing interval length $|b - a|$ is below the desired absolute root tolerance (error)

The advantage of this method is its simplicity and robustness.

Note a trick: if the sign of $f(c_n)$ is the same as $f(a)$, then obviously $f(a)f(c_n) > 0$. Thus, we can use the latter inequality to check whether *a* or *b* should be replaced with c_n (this is done in our textbook). However, computing unneeded products is wasteful, and is best avoided in performance-critical applications: it's very easy for the computer to check the sign of $f(c_n)$ without computing any products.

Convergence speed:

Let's denote the absolute error

$$|\epsilon_n| = |c_n - r|$$

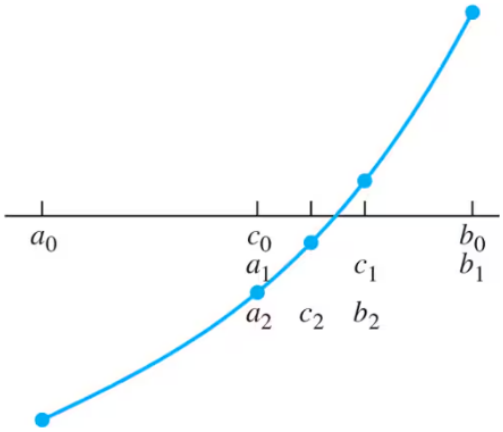
Since we are halving the bracketing interval at each step, it is easy to see that the absolute error after *n* steps satisfies:

$$|\epsilon_n| \le \frac{b-a}{2^n}$$

The textbook figure below illustrates the method, but note that the first iterate is counted as c_0 rather than c_1:

Figure 1.2

The Bisection Method.



On the first step, the sign of $f(c_0)$ is checked. Since $f(c_0)\,f(b_0) < 0$, set $a1 = c0$, $b1 = b_0$, and the interval is replaced by the right half $[a_1, b_1]$. On the second step, the subinterval is replaced by its left half $[a_2, b_2]$.

1.2 Direct iteration (fixed-point iteration)

Direct iteration is even simpler than the bisection method, but is the basis for even more advanced methods:

First, convert the equation for the root of a continuous function $f(x) = 0$ to the form $x = g(x)$.

This can always be done, for instance by adding x to both sides of the equation, and scaling $f(x)$ by any constant: $g(x) \equiv x + cf(x)$. At the root we have $f(r) = 0$, therefore $g(r) \equiv r + c \cdot 0 = r$, which is what we want.

Then, starting with an initial guess x_0, repeatedly apply the function $g(x)$ until the sequence converges sufficiently:

$$x_{n+1} = g(x_n).$$

It is easy to see that if this sequence does converge, $x_n \to r$, it can only converge to the solution we seek, $r = g(r)$.

The solution *r* is called a "fixed point" since the iterates x_n stop changing if x_n reaches *r*.

Convergence of iteration.

However, does this iteration actually converge? To answer this question, let's denote the error of the method at iteration step *n* as $\epsilon_n = x_n - r$, which implies $x_n = r + \epsilon_n$ (with this definition, a positive error ϵ_n implies that our approximant x_n is an over-estimate, and *vice versa*, which is convenient, but the ϵ_n sign is not as important as its absolute value $|\epsilon_n|$)

Plugging $x_n = r + \epsilon_n$ into our iteration $x_{n+1} = g(x_n)$, we obtain:

$$x_{n+1} = r + \epsilon_{n+1} = g(r + \epsilon_n).$$

If $g(x)$ is continuously differentiable in some neighborhood of the root, we can expand it around *r* using the Taylor theorem, to lowest order:

$$r + \epsilon_{n+1} = g(r + \epsilon_n) = g(r) + g'(\alpha)\epsilon_n \quad \text{where } \alpha \in (r, x_n)$$

Since by definition $r = g(r)$, we can cancel *r* on both sides of this equation, leading to a simple result:

$$\epsilon_{n+1} = g'(\alpha)\epsilon_n$$

Obviously, sufficiently close to the root $\alpha \approx r$, and we obtain an approximation:

$$\epsilon_{n+1} \approx g'(r)\epsilon_n$$

Therefore,

$$|\epsilon_{n+1}| \approx |g'(r)| \cdot |\epsilon_n| \equiv C\,|\epsilon_n|$$

Thus, the error **contracts** at each step by about a factor $C = |g'(r)|$. It is obvious that convergence requires $|g'(r)| < 1$.

Such convergence is called ***linear*** since the error at the next step is a linear function of the error at the previous step. In the near future we will learn faster methods: for instance, quadratic convergence corresponds to $\epsilon_{n+1} \sim \epsilon_n^2$

Applying this relationship over and over, i.e. $\epsilon_n \approx g'(r)\epsilon_{n-1} \approx [g'(r)]^2\epsilon_{n-2} \approx \ldots$, we obtain:

$$|\epsilon_n| \approx C^n|\epsilon_0|$$

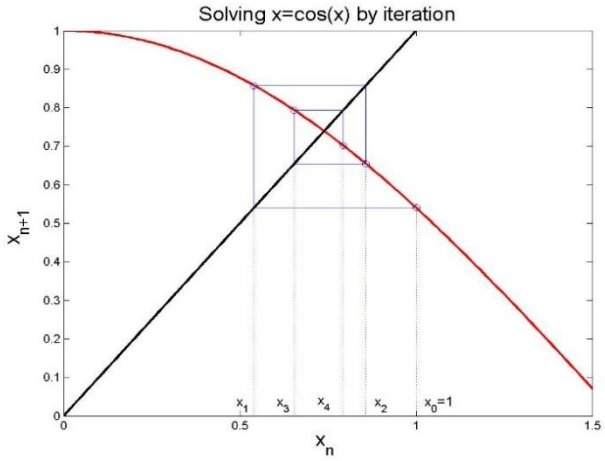
More precise upper bound on the error. The above approximation for the error is only accurate once we converge sufficiently close to the root. For a more precise relationship, find an interval $[a, b]$ containing both the root r and the initial guess x_0, such that $g(x)$ maps $[a, b]$ within itself. Then, denoting $C = max_{[a,b]}|g'(x)| < 1$, we obtain a more rigorous relationship, $|\epsilon_n| \le C^n|\epsilon_0|$.

These results can be viewed as a simple application of the **contraction mapping theorem,** or the **Banach fixed-point theorem,** one of the most important and powerful theorems in mathematics.

Example 1: Solve numerically the equation $\boldsymbol{x = cos(x)}$.

This is a transcendental equation that cannot be solved in closed form: it is easy to see that the root of this equation is some number between 0 and $\frac{\pi}{2}$ (see graph below)**.** Let's see if iteration $x_{n+1} = cos(x_n)$ works, with initial guess $x_0 = 1$:

$x_1 = \cos(x_0) = \cos(1.0) = 0.5403$
$x_2 = \cos(x_1) = \cos(0.5403) = 0.8576$
$x_3 = \cos(x_2) = \cos(0.8576) = 0.6543$
$x_4 = \cos(x_3) = \cos(0.6543) = 0.7935$



This sequence does converge to the exact solution $r \approx 0.739$. Let's check this: $g'(x) = -\sin(x)$, and since $r \in \left(0, \frac{\pi}{2}\right)$, we have $\boldsymbol{C = |g'(r)| = |\sin(r)| < 1}$**.** In this case we say that the root is a *stable fixed point* of this iteration.

We can illustrate the convergence of this sequence using a "**cobweb plot**": at each iteration we are substituting the new *y*-value (where $y = g(x)$) back into the argument of $g(x)$: geometrically, this is accomplished by projecting the y-value onto the diagonal line $y = x$ (this corresponds to the horizontal cobweb segments), and then plug that value back into $g(x)$ (see vertical cobweb segments)

APPENDIX: Other applications of iteration

The importance of iteration equation $x_{n+1} = g(x_n)$ goes far beyond root-solving. In fact, this equation is also known as a **first-order difference equation,** and can describe the **dynamics** of some systems where time is naturally divided in discrete units, like days or years. In this case, think of ***n*** as a point in time, $\boldsymbol{t_n}$.

Pharmacokinetics

Consider a medication with a half-life of $\boldsymbol{T_{1/2}}$ taken once a day, with equal time between doses $\boldsymbol{T}$, and each dose equal to $\boldsymbol{\Delta x}$. From day to day, how does the amount of medication in your system right **before** intake change? Assume full instantaneous absorption and a single pathway of extrusion, as described by the half-life $T_{1/2}$.

Denoting the amount of medication in the system right **before** intake on day *n* as x_n, we obtain:

$$\boldsymbol{x_{n+1} = g(x_n) = \gamma \cdot (x_n + \Delta x)}$$

where $\boldsymbol{\gamma = 2^{-T/T_{1/2}}}$ describes exponential decay of the previous day's amount, and the addition in brackets describes a new dose intake the day before. The iteration's initial value (first day of intake) corresponds to $\boldsymbol{x_0 = 0}$.

The fixed point of this iteration is easy to find:

$$\boldsymbol{r = g(r) = \gamma\,(r + \Delta x) \Rightarrow r(1-\gamma) = \gamma \Delta x \Rightarrow r = \Delta x \frac{\gamma}{1-\gamma}}$$

Example: suppose you are taking a small cup of coffee each day at 8am. What is the amount of caffeine in your system at steady state in the morning right before drinking your daily cup of coffee? Assume that a single cup contain $\Delta x = 120mg$ of caffeine, and that its half-life is $\boldsymbol{T_{1/2} = 6\ hrs}$

Answer: in this case $\boldsymbol{\gamma = 2^{-24/6} = 2^{-4} = \frac{1}{16}}$, and we obtain $\boldsymbol{r = 120\frac{1/16}{1-1/16} = \frac{120}{16-1} = \frac{120}{15} = 8\ mg}$. This means that the amount of caffeine in your system never falls below 8mg!

Note 1: the fixed point is always **stable**, since $|g'(r)| = \gamma = 2^{-T/T_{1/2}} < 1$. Therefore, no matter how long the medication half-life relative to the 24-hour period, eventually the medication amount has to asymptotically approach a steady state.

This is not quite obvious to everyone initially, since strictly speaking medication amount x_n is always bigger than x_{n-1}, but they both gradually approach the same steady state limit (fixed point), ***r***.

Note 2: we could use a natural exponent instead of base-2 exponent: $\gamma = 2^{-T/T_{1/2}} = e^{-T/\tau}$, where $\tau \equiv T_{1/2}/\ln 2$.

Chaos

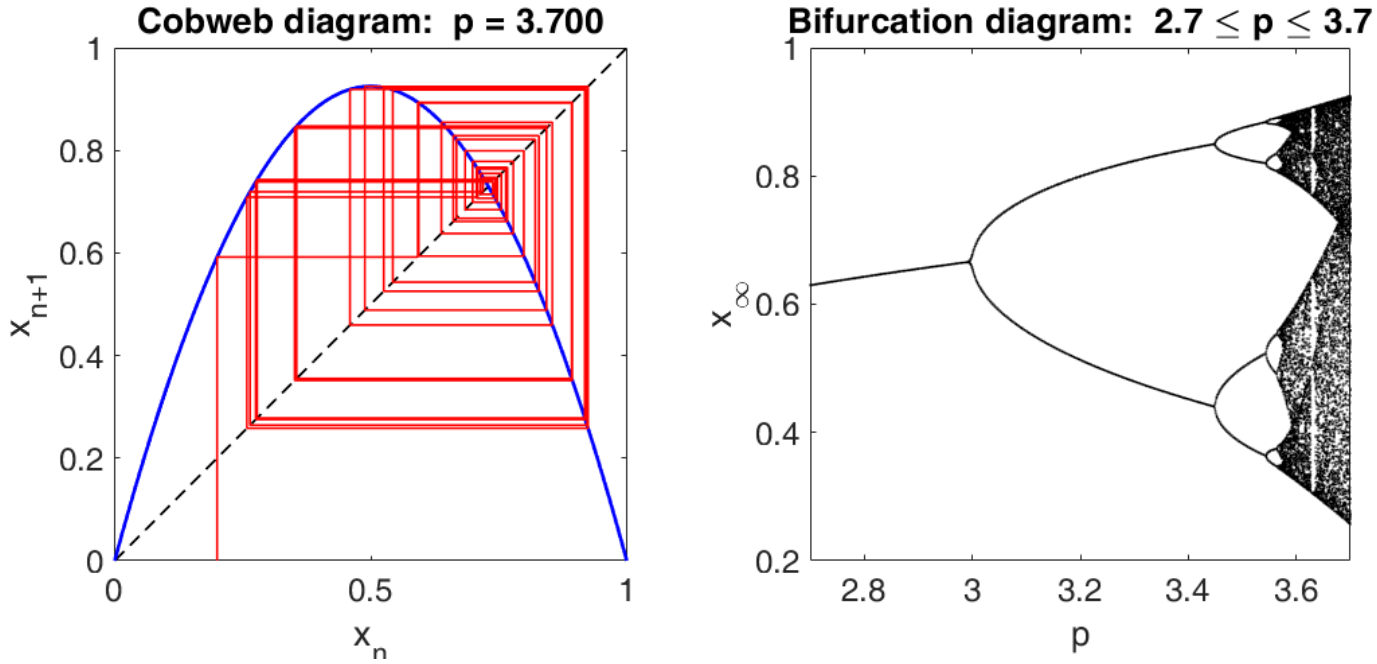
The case of unstable fixed points can be even more interesting than the case of stable points. For example, examine the following simple quadratic iteration, called the logistic map (where ***p*** is a constant parameter):

$$\boldsymbol{x_{n+1} = g(x_n) = p x_n (1 - x_n)}$$

The iteration fixed points are easy to find: $r_1 = 0, r_2 = 1 - \frac{1}{p}$.

Examining $\left|g'(r_{1,2})\right|$, you will find that for parameter values $\boldsymbol{p > 3}$, both fixed points lose stability, and stable cycles emerge instead. Moreover, when $\boldsymbol{p > 3.57}$, the system exhibits deterministic **chaos**!

The dependence of iteration behavior on parameter value *p* can be depicted using a so-called **bifurcation diagram**, which shows how the long-term asymptotic values $\lim_{n\to\infty} x_n \equiv x_\infty$ (understood as limit of all subsequences of x_n whenever the actual limit does not exist) change as a function of parameter *p*:



Note in particular the period doubling cascade emerging at $\boldsymbol{p = 3}$: at this parameter value a **period-2 cycle** emerges (like the one you observed in your very first in-class assignment devoted to direct iteration). It is interesting to note that the **period-2** values $\boldsymbol{r_{1,2}}$ are stable fixed points of a "doubled" iteration:

$$\boldsymbol{r = g_2(r) \equiv g(g(r))}$$

At greater values of p, these period-2 cycles lose stability, giving rise to period-4 cycles, which are fixed points of $g_4(x) \equiv g(g(g(g(x))))$. The distance between ***p*** values corresponding to these period-doubling bifurcation *p*-values decreases, eventually producing dense set of unstable periodic orbits. For such ***p*** values, a chaotic (deterministic but poorly predictable) sequence is produced, like the one you see in the left plot.