4 Root Finding algorithms

4.1 Rate of convergence

In numerical analysis, the order of convergence and the rate of convergence of a convergent sequence are quantities that represent how quickly the sequence approaches its limit. A sequence (x_n) that converges to x^* is said to have order of convergence $q \geq 1$ and rate of convergence μ if

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^q} = \mu$$

The rate of convergence μ is also called the asymptotic error constant.

4.1.1 Q-convergence definitions

Suppose that the sequence (x_k) converges to the number L. The sequence is said to converge *Q-linearly* to L if there exists a number $\mu \in (0, 1)$ such that:

$$\lim_{k\to\infty} \frac{|x_{k+1} - L|}{|x_k - L|} = \mu.$$

The number μ is called the rate of convergence.

The sequence is said to converge *Q-superlinearly* to L (i.e. faster than linearly) if

$$\lim_{k\to\infty} \frac{|x_{k+1} - L|}{|x_k - L|} = 0$$

and it is said to converge *Q-sublinearly* to L (i.e. slower than linearly) if

$$\lim_{k\to\infty} \frac{|x_{k+1} - L|}{|x_k - L|} = 1.$$

If the sequence converges *sublinearly* and additionally

$$\lim_{k\to\infty} \frac{|x_{k+2} - x_{k+1}|}{|x_{k+1} - x_k|} = 1$$

then it is said that the sequence (x_k) converges logarithmically to L.

In order to further classify convergence, the order of convergence is defined as follows. The sequence is said to converge with order q to L for $q \geq 1$ if:

$$\lim_{k\to\infty} \frac{|x_{k+1} - L|}{|x_k - L|^q} < M$$

For some positive constant $M > 0$ (not necessarily less than 1 if $q > 1$). In particular, convergence with order:

- $q = 1$ is called linear convergence (if $M < 1$),
- $q = 2$ is called quadratic convergence.
- $q = 3$ is called cubic convergence.
- etc.

Some sources require that q is strictly greater than 1 since the $q = 1$ case requires $M < 1$ so is best treated separately. It is not necessary, however, that q be an integer. For example, the secant method, when converging to a regular, simple root, has an order of $\varphi \approx 1.618$.

4.1.2 Order estimation

A practical method to calculate the order of convergence for a sequence is to calculate the following sequence, which converges to q:

$$q \approx \frac{\log \left|\frac{x_{k+1}-x_k}{x_k-x_{k-1}}\right|}{\log \left|\frac{x_k-x_{k-1}}{x_{k-1}-x_{k-2}}\right|}$$

4.1.3 Examples

Consider the following sequences with $L = 0$:

$$(a_k) = \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \ldots, \frac{1}{2^k}, \ldots\right\}.$$

To determine the type of convergence, we plug the sequence into the definition of Q-linear convergence,

$$\lim_{k\to\infty} \frac{\left|1/2^{k+1} - 0\right|}{\left|1/2^k - 0\right|} = \lim_{k\to\infty} \frac{2^k}{2^{k+1}} = \frac{1}{2}.$$

Thus, we find that (a_k) converges Q-linearly and has a convergence rate of $\mu = 1/2$.

The sequence:

$$(b_k) = \left\{1, 1, \frac{1}{4}, \frac{1}{4}, \frac{1}{16}, \frac{1}{16}, \ldots, \frac{1}{4\left\lfloor \frac{k}{2} \right\rfloor}, \ldots\right\}$$

does not converge linearly to 0 with a rate of $1/2$ under the Q-convergence definition. (Note that $\lfloor x \rfloor$ is the floor function, which gives the largest integer that is less than or equal to x.)

The sequence:

$$(c_k) = \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{16}, \frac{1}{256}, \frac{1}{65,536}, \ldots, \frac{1}{2^{2^k}}, \ldots\right\}$$

converges super-linearly. In fact, it is quadratically convergent.

Finally, the sequence

$$(d_k) = \left\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \ldots, \frac{1}{k+1}, \ldots\right\}$$

converges sublinearly and logarithmically.

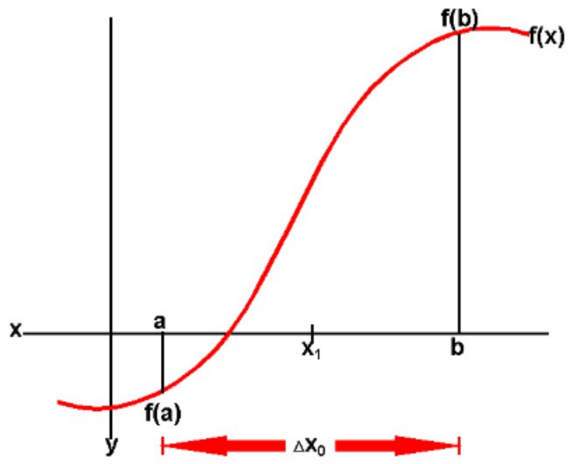
4.1.4 Bisection Method

The Bisection Method is a numerical method for estimating the roots of a polynomial $f(x)$. It is one of the simplest and most reliable but it is not the fastest method. Assume that $f(x)$ is continuous.

Given a continuous function $f(x)$:

1. Find points $\mathbf{a}$ and $\mathbf{b}$ such that $\mathbf{a} < \mathbf{b}$ and $\mathbf{f}(\mathbf{a}) * \mathbf{f}(\mathbf{b}) < 0$.
2. Take the interval $[\mathbf{a}, \mathbf{b}]$ and find its midpoint $\mathbf{x}_1$.
3. If $\mathbf{f}(\mathbf{x}_1) = 0$ then $\mathbf{x}_1$ is an exact root, else if $\mathbf{f}(\mathbf{x}_1) * \mathbf{f}(\mathbf{b}) < 0$ then let $\mathbf{a} = \mathbf{x}_1$, else if $\mathbf{f}(\mathbf{a}) * \mathbf{f}(\mathbf{x}_1) < 0$ then let $\mathbf{b} = \mathbf{x}_1$.
4. Repeat steps 2 & 3 until $\mathbf{f}(\mathbf{x}_\mathrm{i}) = 0$ or $|\mathbf{f}(\mathrm{x}_\mathrm{i})| <= \mathbf{DOA}$, where DOA stands for degree of accuracy.

This algorithm shrinks Δx until a solution is found.



References: http://www2.lv.psu.edu/ojj/courses/cmpsc-201/numerical

4.2 Regula-Falsi Method

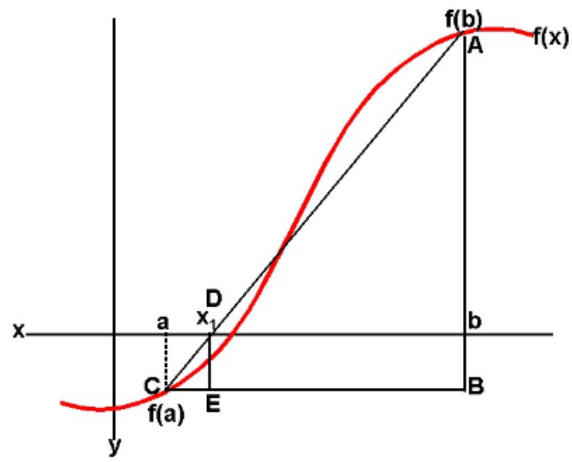
The Regula–Falsi Method is a numerical method for estimating the roots of a polynomial $f(x)$. A value x replaces the midpoint in the Bisection Method and serves as the new approximation of a root of $f(x)$. The objective is to make convergence faster. Assume that $f(x)$ is continuous.

Given a continuous function $f(x)$

1. Find points a and b such that $a < b$ and $f(a) * f(b) < 0$.
2. Take the interval $[a, b]$ and determine the next value of x_1.
3. If $f(x1) = 0$ then x_1 is an exact root, else if $f(x1) * f(b) < 0$ then let $a = x_1$, else if $f(a) * f(x1) < 0$ then let $b = x_1$.
4. Repeat steps 2 3 until $f(x_i) = 0$ or $|f(xi)| \leq$ DOA, where DOA stands for degree of accuracy.

If the root is in $[a, xi]$, then the next interpolation line is drawn between $(a, f(a))$ and $(xi, f(xi))$ in order to find x_{i+1}, otherwise, if the root is in $[xi, b]$, then the next interpolation line is drawn between $(xi, f(xi))$ and $(b, f(b))$.

This is why we say "This method brackets a zero". The difference to the secant method is the bracketing interval. Meaning that the new secant root is not computed from the last two secant roots, but from the last two where the function values have opposing signs.



References: http://www2.lv.psu.edu/ojj/courses/cmpsc-201/numerical

4.3 Secant Methods

The method for finding a zero of a function f, the secant method is defined by the recurrence relation.

$$x_n = x_{n-1} - f(x_{n-1})\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})} = \frac{x_{n-2}f(x_{n-1}) - x_{n-1}f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}.$$

As can be seen from this formula, two initial values x0 and x1 are required. Ideally, they should be chosen close to the desired zero.

Derivation of the method Starting with initial values x0 and x1, we construct a line through the points (x0, f(x0)) and (x1, f(x1)), as shown in the picture above. In slope–intercept form, the equation of this line is

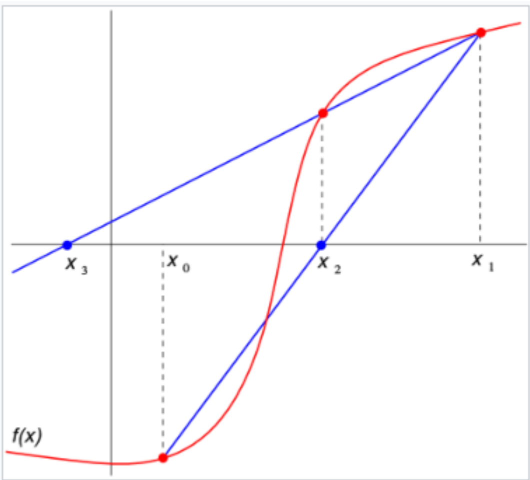
$$y = \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1) + f(x_1).$$

The root of this linear function, that is the value of x such that y = 0 is

$$x = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}.$$

We then use this new value of x as x_2 and repeat the process, using x_1 and x_2 instead of x_0 and x_1. We continue this process, solving for x_3, x_4, etc. until we reach a sufficiently high level of precision (a sufficiently small difference between x_n and x_{n1}):

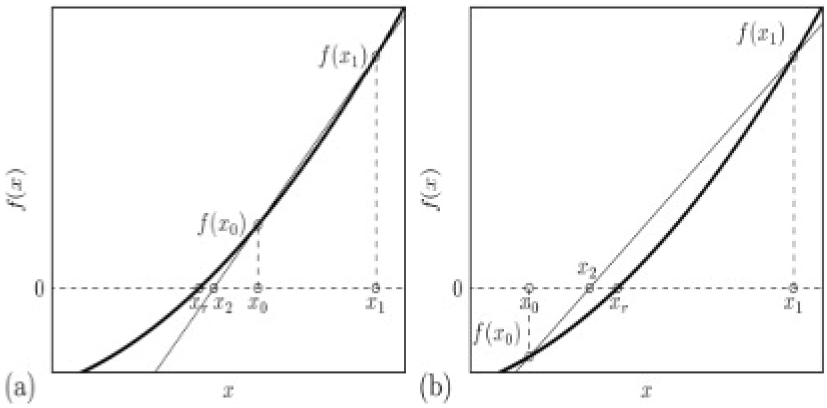
$$\begin{aligned}
x_2 &= x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)} \\
x_3 &= x_2 - f(x_2)\frac{x_2 - x_1}{f(x_2) - f(x_1)}, \\
&\vdots \\
x_n &= x_{n-1} - f(x_{n-1})\frac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}.
\end{aligned}$$



The first two iterations of the secant method. The red curve shows the function *f*, and the blue lines are the secants. For this particular case, the secant method will not converge to the visible root.

The secant method can be interpreted as a method in which the derivative is replaced by an approximation and is thus a quasi-Newton method. If we compare Newton's method with the secant method, we see that Newton's method converges faster (order 2 against $\phi 1.6$).

Note that the secant method does not bracket a root in every iteration, for example:



4.4 Newton-Raphson

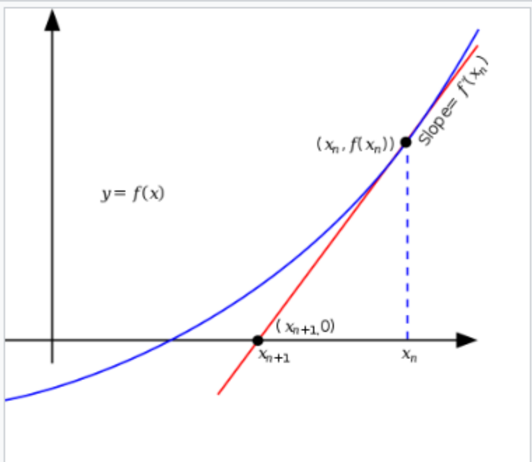
Suppose $f(x)$ is differentiable, and that x_n is such that $f(x_n) \neq 0$. If the tangent line to the curve $f(x)$ at $x = x_n$ intercepts the x-axis at x_{n+1} then the slope is given by:

$$f'(x_n) = \frac{f(x_n) - 0}{x_n - x_{n+1}}.$$

Solving for x_{n+1} gives us:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

But although this is a root of the tangent, it might not be a root for the function be are considering.



x_{n+1} is a better approximation than x_n for the root x of the function f (blue curve)

4.4.1 Brent's Method

Motivation: Is there a way to combine superlinear convergence with the sureness of bisection? Yes. We can keep track of whether a supposedly superlinear method is actually converging the way it is supposed to, and, if it is not, we can intersperse bisection steps so as to guarantee at least linear convergence.

The method is guaranteed (by Brent) to converge, so long as the function can be evaluated within the initial interval known to contain a root. Brent's method combines root bracketing, bisection, and inverse quadratic interpolation to converge from the neighborhood of a zero crossing. While the false position and secant methods assume approximately linear behavior between two prior root estimates, inverse quadratic interpolation uses three prior points to fit an inverse quadratic function (x as a quadratic function of y) whose value at y = 0 is taken as the next estimate of the root x. Of course, one must have contingency plans for what to do if the root falls outside of the brackets.

4.4.2 Lagrange Polynomials:

Through any two points there is a unique line. Through any three points, a unique quadratic.

Lets (x_1, y_1), (x_2, y_2) and (x_3, y_3) be such that $x_i \neq x_j \ \forall i \in 1, 2, 3$. Then the quadratic polynomial that passes through all those points is given by:

$$f(x) = a_0 + a_1 x + a_2 x^2$$

where (a_1, a_2, a_3) are yet to be determined. Notice that we have a $3x3$ system of equations:

$$\begin{aligned} f(x_1) &= a_0 + a_1 x_1 + a_2 x_1^2 = y_1 \\ f(x_2) &= a_0 + a_1 x_2 + a_2 x_2^2 = y_2 \\ f(x_3) &= a_0 + a_1 x_3 + a_2 x_3^2 = y_3 \end{aligned}$$

To isolate a_2 as a function of a_1:

$$\begin{aligned} a_0 + a_1 x_1 + a_2 x_1^2 - y_1 &= a_0 + a_1 x_2 + a_2 x_2^2 - y_2 \\ a_1(x_1 - x_2) - y_1 + y_2 &= a_2(x_2^2 - x_1^2) \\ a_2 &= a_1 \frac{(x_1 - x_2)}{(x_2^2 - x_1^2)} + \frac{(y_2 - y_1)}{(x_2^2 - x_1^2)} \end{aligned}$$

To isolate a_0 as a function of a_1:

$$\begin{aligned} a_0 &= y_3 - a_1 x_3 - a_2 x_3^2 \\ a_0 &= y_3 - a_1 x_3 - \left(a_1 \frac{(x_1 - x_2)}{(x_2^2 - x_1^2)} + \frac{(y_2 - y_1)}{(x_2^2 - x_1^2)} \right) x_3^2 \\ a_0 &= -a_1 \left(x_3 + \frac{(x_1 - x_2)}{(x_2^2 - x_1^2)} x_3^2 \right) + \left(y_3 + \frac{(y_1 - y_2)}{(x_2^2 - x_1^2)} x_3^2 \right) \end{aligned}$$

Then we solve for a_1:

$$\begin{aligned} a_0 + a_1 x_1 + a_2 x_1^2 &= y_1 \\ -a_1 \left(x_3 + \frac{(x_1 - x_2)}{(x_2^2 - x_1^2)} x_3^2 \right) + \left(y_3 + \frac{(y_1 - y_2)}{(x_2^2 - x_1^2)} x_3^2 \right) + a_1 x_1 + a_1 \frac{(x_1 - x_2)}{(x_2^2 - x_1^2)} + \frac{(y_2 - y_1)}{(x_2^2 - x_1^2)} x_1^2 &= y_1 \\ -a_1 \left(x_3 + \frac{(x_1 - x_2)}{(x_2^2 - x_1^2)} x_3^2 - x_1 - \frac{(x_1 - x_2)}{(x_2^2 - x_1^2)} \right) = y_1 - \left(y_3 + \frac{(y_1 - y_2)}{(x_2^2 - x_1^2)} x_3^2 \right) - \frac{(y_2 - y_1)}{(x_2^2 - x_1^2)} x_1^2 \\ -a_1 \frac{\left((x_2^2 - x_1^2)(x_3 - x_1) + (x_1 - x_2) x_3^2 - (x_1 - x_2) \right)}{(x_2^2 - x_1^2)} = \frac{(x_2^2 - x_1^2)(y_1 - y_3) - (y_1 - y_2) x_3^2 - (y_2 - y_1) x_1^2}{(x_2^2 - x_1^2)} \\ a_1 = \frac{(x_2^2 - x_1^2)(y_1 - y_3) - (y_1 - y_2) x_3^2 - (y_2 - y_1) x_1^2}{(x_2^2 - x_1^2)(x_1 - x_3) - (x_1 - x_2) x_3^2 - (x_2 - x_1)} \end{aligned}$$

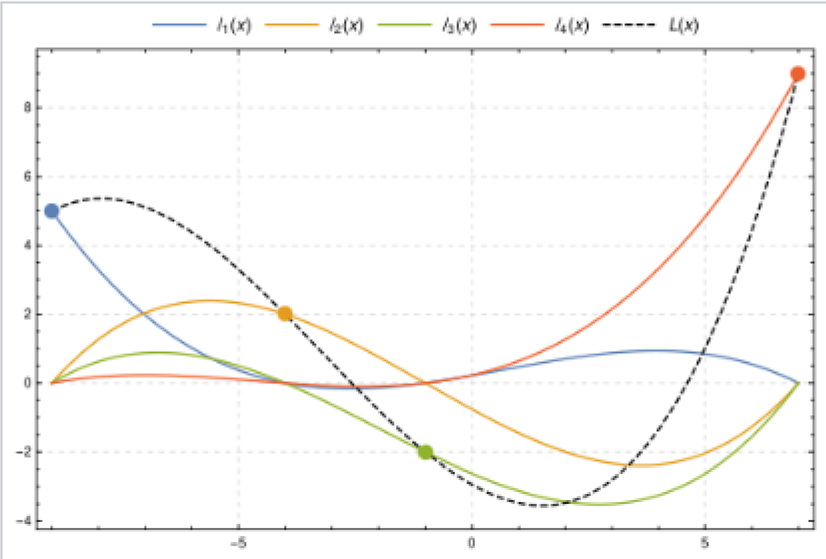
Solving for a_0 and a_2 we can then construct Lagrange interpolating polynomial.

Given a set of $k + 1$ nodes $\{x_0, x_1, \ldots, x_k\}$, which must all be distinct, $x_j \neq x_m$ for indices $j \neq m$, the Lagrange basis for those nodes is the set of polynomials $\{\ell_0(x), \ell_1(x), \ldots, \ell_k(x)\}$ each of degree k which take values $\ell_j(x_m) = 0$ if $m \neq j$ and $\ell_j(x_j) = 1$ st:

$$\ell_j(x) = \frac{(x - x_0)}{(x_j - x_0)} \cdots \frac{(x - x_{j-1})}{(x_j - x_{j-1})} \frac{(x - x_{j+1})}{(x_j - x_{j+1})} \cdots \frac{(x - x_k)}{(x_j - x_k)}$$

The Lagrange interpolating polynomial for those nodes through the corresponding values $\{y_0, y_1, \ldots, y_k\}$ is the linear combination:

$$L(x) = \sum_{j=0}^{k} y_j \ell_j(x)$$



This image shows, for four points ($(-9, 5)$, $(-4, 2)$, $(-1, -2)$, $(7, 9)$), the (cubic) interpolation polynomial $L(x)$ (dashed, black), which is the sum of the *scaled* basis polynomials $y_0\ell_0(x)$, $y_1\ell_1(x)$, $y_2\ell_2(x)$ and $y_3\ell_3(x)$. The interpolation polynomial passes through all four control points, and each *scaled* basis polynomial passes through its respective control point and is 0 where x corresponds to the other three control points.

4.4.3 Specificis of Brent's Method

After the first iteration, either using secant or bisection, we will be left with 3 point $[a, f(a)], [b, f(b)], [c, f(c)]$. Then using Lagrange's formula to compute the inverse quadratic (x as a function of y) we get:

$$x = \frac{[y - f(a)][y - f(b)]c}{[f(c) - f(a)][f(c) - f(b)]} + \frac{[y - f(b)][y - f(c)]a}{[f(a) - f(b)][f(a) - f(c)]} + \frac{[y - f(c)][y - f(a)]b}{[f(b) - f(c)][f(b) - f(a)]}$$

Setting y to zero gives a result for the next root estimate, which can be written as

$$x = b + P/Q$$

where, in terms of

$$R \equiv f(b)/f(c), \quad S \equiv f(b)/f(a), \quad T \equiv f(a)/f(c)$$

we have

$$P = S[T(R - T)(c - b) - (1 - R)(b - a)]$$
$$Q = (T - 1)(R - 1)(S - 1).$$

Brent's method guards against this problem by maintaining brackets on the root and checking where the interpolation would land before carrying out the division. When the correction P/Q would not land within the bounds, or when the bounds are not collapsing rapidly enough, the algorithm takes a bisection step. Thus, Brent's method combines the sureness of bisection with the speed of a higher-order method when appropriate. We recommend it as the method of choice for general one-dimensional root finding where a function's values only (and not its derivative or functional form) are available.

References: Chapter 9.3 Numerical Recipes

5 Numerical Differentiation:

5.0.1 One sided derivatives:

By definition:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Take a small h, say 0.0001, and evaluate the last expression. Looks pretty straightforward, no? No. So, how do we choose h ? There are two separate issues:

- Rounding error.
- Truncation error

For example, if $f(x)$ is excess demand, and is calculated with truncation error ϵ_f, f' will have error of approximately $\sqrt{\epsilon_f}$. This is numerically relevant, for example if $\epsilon_f = 10^{-5}$, then $\sqrt{\epsilon_f} = 0.003$. Because ϵ_f is typically greater than machine precision (which is typically around 10^{-15} for double precision reals), the best lower bound is $\sim \sqrt{\epsilon_m}$. The actual error will often be much larger, because $\epsilon_f \gg \epsilon_m$ (Since doing calculations in general will increase round-off error).

5.0.2 Two sided derivatives:

An even more precise but but even slower approach is to define a two-sided derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

This also requires 3 function evaluations since we often need $f(x)$ for other reasons, so you are already going to compute it. So two-sided derivatives require 2 additional evaluations compared to 1 for one-sided. The benefit is that for one sided derivatives, the truncation error is $\sim h$. For two sided, it is $\sim h^2$.

Chapter 10 distributed in class.