COURSE 11

5. Numerical methods for solving nonlinear equations in $\mathbb{R}$

The roots of the iterative methods are traced back to Egyptians and Babylonians (cc. 1800 B.C.).

Example of nonlinear equation. Kepler's Equation: consider a two-body problem like a satellite orbiting the earth (or a planet revolving around the sun). Kepler discovered that the orbit is an ellipse and the central body F (earth) is in a focus of the ellipse. The speed of the satellite P is not uniform: near the earth it moves faster than far away. It is used Kepler's law to predict where the satellite will be at a given time. If we want to know the position of the satellite for t = 9 minutes, then we have to solve the equation $f(E) = E - 0.8 \sin E - 2\pi/10 = 0$.

Let $f : \Omega \to \mathbb{R}$, $\Omega \subset \mathbb{R}$. Consider the equation

$$f(x) = 0, \quad x \in \Omega. \tag{1}$$

We attach a mapping $F : D \to D,\ D \subset \Omega^n$ to this equation.

Let $(x_0, ..., x_{n-1}) \in D$. Using F and the numbers $x_0, x_1, ..., x_{n-1}$ we construct iteratively the sequence

$$x_0, x_1, ..., x_{n-1}, x_n, ... \tag{2}$$

with

$$x_i = F\left(x_{i-n}, ..., x_{i-1}\right), \quad i = n, \ldots. \tag{3}$$

The problem consists in choosing F and $x_0, ..., x_{n-1} \in D$ such that the sequence (2) to be convergent to the solution of the equation (1).

Definition 1 *The procedure of approximation the solution of equation (1) by the elements of the sequence (2), computed as in (3), is called* F**-method.**

The numbers $x_0, x_1, ..., x_{n-1}$ *are called* **the starting (initial) points** *and the* k*-th element of the sequence (2) is called an approximation of* k*-th index of the solution.*

If the set of starting points has only one element then the F-method is **an one-step method;** if it has more than one element then the F-method is **a multistep method**.

Definition 2 *If the sequence (2) converges to the solution of the equation (1) then the F-method is convergent, otherwise it is divergent.*

Definition 3 *Let $\alpha \in \Omega$ be a solution of the equation (1) and let $x_0, x_1, ..., x_{n-1}, x_n, ...$ be the sequence generated by a given F-method. The number p having the property*

$$\lim_{x_i \to \alpha} \frac{|\alpha - F(x_{i-n}, ..., x_i)|}{|\alpha - x_i|^p} = C \neq 0, \quad C = constant,$$

is called the order *of the F-method.*

For one-step methods, the above condition reduces to

$$\lim_{x_i \to \alpha} \frac{|\alpha - F(x_i)|}{|\alpha - x_i|^p} = \lim_{x_i \to \alpha} \frac{|\alpha - x_{i+1}|}{|\alpha - x_i|^p} = C \neq 0, \quad C = constant,$$

We construct some classes of F-methods based on the interpolation procedures.

Let $\alpha \in \Omega$ be a solution of the equation (1) and $V(\alpha)$ a neighborhood of α. Assume that f has inverse on $V(\alpha)$ and denote $g := f^{-1}$. Since

$$f(\alpha) = 0$$

it follows that

$$\alpha = g(0).$$

This way, the approximation of the solution α is reduced to the approximation of $g(0)$.

Definition 4 *The approximation of g by means of an interpolating method, and of α by the value of g at the point zero is called* **the inverse interpolation procedure.**

5.1. One-step methods

Let F be a one-step method, i.e., for a given x_i we have $x_{i+1} = F(x_i)$.

Remark 5 *If $p = 1$, a sufficient convergence condition is $|F'(x)| < 1$.*

All information on f are given at a single point, the starting value $\Rightarrow$ we are lead to Taylor interpolation.

Theorem 6 *Let α be a solution of equation (1), $V(\alpha)$ a neighborhood of α, $x, x_i \in V(\alpha)$, f fulfills the necessary continuity conditions. Then we have the following method, denoted by F_m^T, for approximating α:*

$$F_m^T(x_i) = x_i + \sum_{k=1}^{m-1} \frac{(-1)^k}{k!} [f(x_i)]^k g^{(k)}(f(x_i)), \tag{4}$$

where $g = f^{-1}$.

Proof. There exists $g = f^{-1} \in C^m[V(0)]$. Let $y_i = f(x_i)$ and consider Taylor interpolation formula

$$g(y) = (T_{m-1}g)(y) + (R_{m-1}g)(y),$$

with

$$(T_{m-1}g)(y) = \sum_{k=0}^{m-1} \frac{1}{k!}(y - y_i)^k g^{(k)}(y_i),$$

and $R_{m-1}g$ is the corresponding remainder.

Since $\alpha = g(0)$ and $g \approx T_{m-1}g$, it follows

$$\alpha \approx (T_{m-1}g)(0) = x_i + \sum_{k=1}^{m-1} \frac{(-1)^k}{k!} y_i^k g^{(k)}(y_i).$$

Hence,

$$x_{i+1} := F_m^T(x_i) = x_i + \sum_{k=1}^{m-1} \frac{(-1)^k}{k!} [f(x_i)]^k g^{(k)}(f(x_i))$$

is an approximation of α, and F_m^T is an approximation method for the solution α. ■

Concerning the order of the method F_m^T we state:

Theorem 7 *If* $g = f^{-1}$ *satisfies* $g^{(m)}(0) \neq 0$, *then* $\text{ord}(F_m^T) = m$.

Remark 8 *We have an upper bound for the absolute error in approximating* α *by* x_{i+1} :

$$\left|\alpha - F_m^T(x_i)\right| \leq \frac{1}{m!}[f(x_i)]^m M_m g, \quad \textit{with } M_m g = \max_{y \in V(0)} \left|g^{(m)}(y)\right|.$$

Particular cases.

1) Case $m = 2$.

$$F_2^T(x_i) = x_i - \frac{f(x_i)}{f'(x_i)}.$$

This method is called **Newton's method** (**the tangent method**). Its order is 2.

2) Case $m = 3$.

$$F_3^T(x_i) = x_i - \frac{f(x_i)}{f'(x_i)} - \frac{1}{2}\left[\frac{f(x_i)}{f'(x_i)}\right]^2 \frac{f''(x_i)}{f'(x_i)},$$

with $\text{ord}(F_3^T) = 3$. So, this method converges faster than F_2^T.

3) Case $m = 4$.

$$F_4^T(x_i) = x_i - \frac{f(x_i)}{f'(x_i)} - \frac{1}{2}\frac{f''(x_i)f^2(x_i)}{[f'(x_i)]^3} + \frac{\left(f'''(x_i)f'(x_i) - 3[f''(x_i)]^2\right)f^3(x_i)}{3![f'(x_i)]^5}.$$

Remark 9 *The higher the order of a method is, the faster the method converges. Still, this doesn't mean that a higher order method is more efficient (due to computation requirements). By the contrary, the most efficient are the methods of relatively low order, due to their low complexity (methods* F_2^T *and* F_3^T*).*

Newton's method (Newton-Raphson method) named after Isaac Newton (1642–1726) and Joseph Raphson (1648–1715), is a root-finding algorithm which produces successively better approximations to the roots of a real-valued function.

This method is so efficient in computing $\sqrt{a}$, that it is a choice even today in modern codes.

Some comments on the history of this method

The traces of this methods can be found in ancient times (Babylon and Egypt, 1800 B.C.), as it appears in the computation of the square root of a number.

Different methods, either algebraically equivalent to Newton's method or of Newton type were known in antiquity to India and then to Arabic culture.

In solving nonlinear problems, Newton ($\approx$1669) and subsequently Raphson (1690) have dealt only with polynomial equations ($x^3-2x-5=0$ is "the classical equation where the Newton method is applied"). Newton has also considered such iterations in solving Kepler's equation $x - e\sin x = M$.

Newton has considered the process of successively computing the corrections, which were added finally altogether to form the final approximation. He didn't compute the successive approximations, but

computes a sequence of polynomials, and only at the end arrives at an approximation for the root: let x_0 be a given first estimate of the solution α of $f(x) = 0$. Write $g_0(x) = f(x)$, and suppose $g_0(x) = \sum_{i=0}^{n} a_i x^i$. Writing $e_0 = \alpha - x_0$ we obtain by binomial expansion about the given x_0 a new polynomial equation in the variable e_0:

$$0 = g_0(\alpha) = g_0(x_0 + e_0) = \sum_{i=0}^{n} a_i (x_0 + e_0)^i$$
$$= \sum_{i=0}^{n} a_i \left[\sum_{j=0}^{i} \binom{i}{j} x_0^j {e_0}^{i-j} \right] = g_1(e_0).$$

Neglecting terms involving higher powers of e_0 (linearizing the explicitly computed polynomial g_1) produces

$$0 = g_1(e_0) \approx \sum_{i=0}^{n} a_i (x_0^i + i x_0^{i-1} e_0) = \sum_{i=0}^{n} a_i x_0^i + e_0 \sum_{i=0}^{n} a_i i x_0^{i-1}$$

from which we deduce that

$$e_0 \approx c_0 = \left(-\sum_{i=0}^{n} a_i x_0^i \right) \Big/ \left(\sum_{i=0}^{n} a_i i x_0^{i-1} \right)$$

and set $x_1 = x_0 + c_0$. Formally this correction can be written $c_0 = -g_0(x_0)/g_0'(x_0) = -f(x_0)/f'(x_0)$.

Now repeat the process, but instead of expanding the original polynomial g_0 about x_1 expand the polynomial g_1 about the point c_0, that is considered to be a first estimate of the solution e_0 of the new equation $g_1(x) = 0$. Thus similarly obtain $0 = g_1(e_0) = g_1(c_0 + e_1) = g_2(e_1)$, where the polynomial g_2 is explicitly computed. Linearizing again produces $e_1 \approx c_1 = -g_1(c_0)/g_0'(c_0)$, corresponding $x_2 = x_1 + c_1$.

Raphson has considered the approximations updated at each step (the usual iterations), a process equivalent to what we use nowadays. However, the derivatives of f (which could be calculated with the "fluxions" of that time) do not appear in their formulae. For more than a century, the belief was that these two variants (of Newton and Raphson) represent two different methods.

Simpson (1740) was the first to apply the method to transcendent equations, using the "fluxions" (the fluxions $\dot{x}$ are "essentially" equivalent to dx/dt.) He even extended it to the solving of nonlinear systems

of two equations, subsequently generalized to the usual form from today.

The formulation of the method using $f'(x)$ was published by Lagrange in 1798.

Due to the Fourier's influential book *Analyse des Équations Determinées* (1837), where Raphson and Simpson were not mentioned, the name of the method remained "Newton". Some authors use "the Newton-Raphson method".

When there exists a neighborhood of α where the F-method is convergent, choosing x_0 in such a neighborhood allows approximating α by terms of the sequence

$$x_{i+1} = F_2^T(x_i) = x_i - \frac{f(x_i)}{f'(x_i)}, \quad i = 0, 1, ...,$$

with a prescribed error ε.

If α is a solution of equation (1) and $x_{n+1} = F_2^T(x_n)$, for approximation error, Remark 8 gives

$$\left|\alpha - x_{n+1}\right| \leq \tfrac{1}{2}[f(x_n)]^2 M_2 g.$$

Lemma 10 *Let $\alpha \in (a,b)$ be a solution of equation (1) and let $x_n = F_2^T(x_{n-1})$. Then*

$$|\alpha - x_n| \leq \tfrac{1}{m_1}|f(x_n)|, \quad \textit{with } m_1 \leq m_1 f = \min_{a \leq x \leq b}\left|f'(x)\right|.$$

Proof. We use the mean formula

$$f(\alpha) - f(x_n) = f'(\xi)(\alpha - x_n),$$

with $\xi \in$ to the interval determined by α and x_n. From $f(\alpha) = 0$ and $|f'(x)| \geq m_1$ for $x \in (a,b)$, it follows $|f(x_n)| \geq m_1 |\alpha - x_n|$, that is

$$|\alpha - x_n| \leq \tfrac{1}{m_1}|f(x_n)|.$$

■

In practical applications the following evaluation is more useful:

Lemma 11 *If* $f \in C^2[a,b]$ *and* F_2^T *is convergent, then there exists* $n_0 \in \mathbb{N}$ *such that*

$$|x_n - \alpha| \leq |x_n - x_{n-1}|, \quad n > n_0.$$

Proof. We start with Taylor formula

$$f(x_n) = f(x_{n-1}) + (x_n - x_{n-1}) f'(x_{n-1}) + \tfrac{1}{2}(x_n - x_{n-1})^2 f''(\xi),$$

where ξ belongs to the interval determined by x_{n-1} and x_n.

Since $x_n = F_2^T(x_{n-1})$, it follows that

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

thus we obtain

$$f(x_n) = \tfrac{1}{2}(x_n - x_{n-1})^2 f''(\xi).$$

Consequently,

$$|f(x_n)| \leq \tfrac{1}{2}(x_n - x_{n-1})^2 M_2 f,$$

and Lemma 10 yields $|\alpha - x_n| \leq \frac{1}{m_1}|f(x_n)|$ so

$$|\alpha - x_n| \leq \frac{1}{2m_1}(x_n - x_{n-1})^2 M_2 f.$$

Since F_2^T is convergent, there exists $n_0 \in \mathbb{N}$ such that

$$\frac{1}{2m_1}|x_n - x_{n-1}| M_2 f < 1, \quad n > n_0.$$

Hence,

$$|\alpha - x_n| \leq |x_n - x_{n-1}|, \quad n > n_0.$$

■

Remark 12 *The starting value is chosen randomly. If, after a fixed number of iterations the required precision is not achieved, i.e., condition* $|x_n - x_{n-1}| \leq \varepsilon$, *does not hold for a prescribed positive* ε, *the computation has to be started over with a new starting value.*

A modified form of Newton's method: - the same value during the computation of f':

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_0)}, \quad k = 0, 1, \ldots.$$

It is very useful because it doesn't request the computation of f' at x_j, $j = 1, 2, \ldots$ but the order is no longer equal to 2.

Another way for obtaining Newton's method.

We start with x_0 as an initial guess, sufficiently close to the α. Next approximation x_1 is the point at which the tangent line to f at $(x_0, f(x_0))$ crosses the Ox-axis. The value x_1 is much closer to the root α than x_0.

We write the equation of the tangent line at $(x_0, f(x_0))$:

$$y - f(x_0) = f'(x_0)(x - x_0).$$

If $x = x_1$ is the point where this line intersects the Ox-axis, then $y = 0$

$$-f(x_0) = f'(x_0)(x_1 - x_0),$$

and solving for x_1 gives

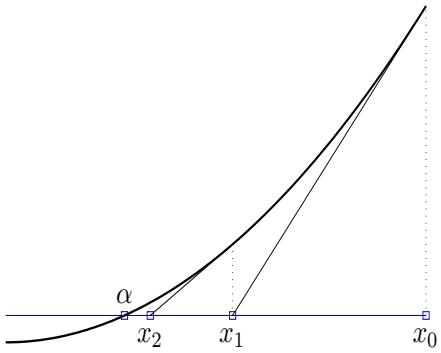
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

By repeating the process using the tangent line at $(x_1, f(x_1))$, we obtain for x_2

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

For the general case we have

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \ n \geq 0. \tag{5}$$



The algorithm:

Let x_0 be the initial approximation.

for $n = 0, 1, ..., ITMAX$

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

A stopping criterion is:

$$|f(x_n)| \leq \varepsilon \text{ or } \left|x_{n+1} - x_n\right| \leq \varepsilon \text{ or } \frac{\left|x_{n+1} - x_n\right|}{\left|x_{n+1}\right|} \leq \varepsilon,$$

where ε is a specified tolerance value.

Example 13 *Use Newton's method to compute a root of* $x^3 - x^2 - 1 = 0$, *to an accuracy of* 10^{-4}. *Use* $x_0 = 1$.

Sol. *The derivative of f is* $f'(x) = 3x^2 - 2x$. *Using* $x_0 = 1$ *gives* $f(1) = -1$ *and* $f'(1) = 1$ *and so the first Newton's iterate is*

$$x_1 = 1 - \frac{-1}{1} = 2 \text{ and } f(2) = 3,\ f'(2) = 8.$$

The next iterate is

$$x_2 = 2 - \frac{3}{8} = 1.625.$$

Continuing in this manner we obtain the sequence of approximations which converges to 1.465571.

5.2. Multistep methods

Lagrange inverse interpolation

Let $y_k = f(x_k)$, $k = 0, ..., n$, hence $x_k = g(y_k)$. We attach the Lagrange interpolation formula to y_k and $g(y_k)$, $k = 0, ..., n$:

$$g = L_n g + R_n g, \tag{6}$$

where

$$(L_n g)(y) = \sum_{k=0}^{n} \frac{(y-y_0)...(y-y_{k-1})(y-y_{k+1})...(y-y_n)}{(y_k-y_0)...(y_k-y_{k-1})(y_k-y_{k+1})...(y_k-y_n)} g(y_k). \tag{7}$$

Taking

$$F_n^L(x_0, ..., x_n) = (L_n g)(0),$$

F_n^L is a $(n+1)-$ steps method defined by

$$\begin{aligned} F_n^L(x_0, ..., x_n) &= \sum_{k=0}^{n} \frac{y_0 \cdots y_{k-1} y_{k+1} \cdots y_n}{(y_k-y_0)...(y_k-y_{k-1})(y_k-y_{k+1})...(y_k-y_n)} (-1)^n g(y_k) \\ &= \sum_{k=0}^{n} \frac{y_0 \cdots y_{k-1} y_{k+1} \cdots y_n}{(y_k-y_0)...(y_k-y_{k-1})(y_k-y_{k+1})...(y_k-y_n)} (-1)^n x_k. \end{aligned}$$

Concerning the convergence of this method we state:

Theorem 14 *If* $\alpha \in (a,b)$ *is solution of equation* $f(x) = 0$, f' *is bounded on* (a,b), *and the starting values satisfy* $|\alpha - x_k| < 1/c, \quad k = 0,...,n$, *with* c =*constant, then the sequence*

$$x_{i+1} = F_n^L\left(x_{n-i},...,x_i\right), \quad i = n, n+1,...$$

converges to α.

Remark 15 *The order* $ord(F_n^L)$ *is the positive solution of the equation*

$$t^{n+1} - t^n - ... - t - 1 = 0.$$

Particular cases.

1) For $n = 1$, the nodes x_0, x_1, we get **the secant method**

$$F_1^L\left(x_0, x_1\right) = x_1 - \frac{\left(x_1 - x_0\right) f\left(x_1\right)}{f\left(x_1\right) - f\left(x_0\right)},$$

Thus,

$$x_{k+1} := F_1^L\left(x_{k-1}, x_k\right) = x_k - \frac{\left(x_k - x_{k-1}\right) f\left(x_k\right)}{f\left(x_k\right) - f\left(x_{k-1}\right)}, \quad k = 1, 2, ...$$

is the new approximation obtained using the previous approximations x_{k-1}, x_k.

The *order* of this method is the positive solution of equation:

$$t^2 - t - 1 = 0,$$

so $ord(F_1^L) = \frac{(1+\sqrt{5})}{2} \approx 1.618$. (the golden ratio).

A modified form of the secant method: if we keep x_1 fixed and we change every time the same interpolation node, i.e.,

$$x_{k+1} = x_k - \frac{\left(x_k - x_1\right) f\left(x_k\right)}{f\left(x_k\right) - f\left(x_1\right)}, \quad k = 2, 3, \ldots.$$

2) For $n = 2$, the nodes x_0, x_1, x_2 and we get

$$F_2^L(x_0, x_1, x_2) = \frac{x_0 f(x_1) f(x_2)}{[f(x_0) - f(x_1)][f(x_0) - f(x_2)]} + \frac{x_1 f(x_0) f(x_2)}{[f(x_1) - f(x_0)][f(x_1) - f(x_2)]} + \frac{x_2 f(x_0) f(x_1)}{[f(x_2) - f(x_0)][f(x_2) - f(x_1)]}.$$

The *order* of this method is the positive solution of equation:

$$t^3 - t^2 - t - 1 = 0,$$

so $ord(F_2^L) = 1.8394$.

Comparing the Newton's method and secant method with respect to the time needed for finding a root with some given precision, we have:

-Newton's method has more computation at one step: it is necessary to evaluate $f(x)$ and $f'(x)$. Secant method evaluates just $f(x)$ (supposing that $f(x_{previous})$ is stored.)

-The number of iterations for Newton's method is smaller (its order is $p_N = 2$). Secant method has order $p_S = 1.618$ and we have that three steps of this method are equivalent with two steps of Newton's method.

- It is proved that if the time for computing $f'(x)$ is greater than (0.44$\times$the time for computing $f(x)$), then the secant method is faster.

Remark 16 *The computation time is not the unique criterion in choosing the method! Newton's method is easier to apply. If $f(x)$ is not explicitly known (for example, it is the solution of the numerical integration of a differential equation), then its derivative is computed numerically. If we consider the following expression for the numerical computation of derivative:*

$$f'(x) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{8}$$

then the Newton's method becomes the secant method.

Another way of obtaining secant method.

Based on approx. the function by a straight line connecting two points on the graph of f (not required f to have opposite signs at the initial points).

The first point, x_2, of the iteration is taken to be the point of intersection of the Ox-axis and the secant line connecting two starting points

$(x_0, f(x_0))$ and $(x_1, f(x_1))$. The next point, x_3, is generated by the intersection of the new secant line, joining $(x_1, f(x_1))$ and $(x_2, f(x_2))$ with the Ox-axis. The new point, x_3, together with x_2, is used to generate the next point, x_4, and so on.

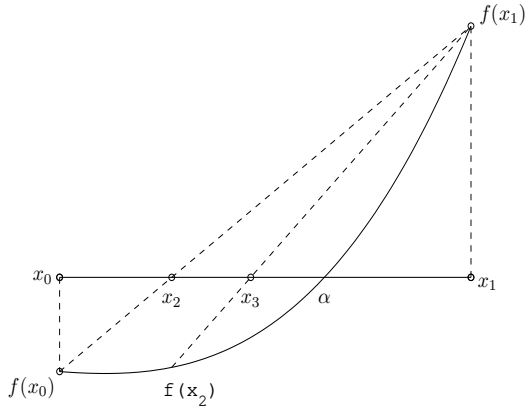
The formula for x_{n+1} is obtained by setting $x = x_{n+1}$ and $y = 0$ in the equation of the secant line from $(x_{n-1}, f(x_{n-1}))$ to $(x_n, f(x_n))$:

$$\frac{x - x_n}{x_{n-1} - x_n} = \frac{y - f(x_n)}{f(x_{n-1}) - f(x_n)} \Leftrightarrow x = x_n + \frac{(x_{n-1} - x_n)(y - f(x_n))}{f(x_{n-1}) - f(x_n)},$$

we get

$$x_{n+1} = x_n - f(x_n) \left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \right]. \tag{9}$$

Note that x_{n+1} depends on the two previous elements of the sequence $\Rightarrow$ two initial guesses, x_0 and x_1,for generating $x_2, x_3, \ldots$.



The algorithm:

Let x_0 and x_1 be two initial approximations.

for $n = 1, 2, ..., ITMAX$

$$x_{n+1} \leftarrow x_n - f(x_n) \left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \right].$$

A suitable stopping criterion is

$$|f(x_n)| \leq \varepsilon \text{ or } \left| x_{n+1} - x_n \right| \leq \varepsilon \text{ or } \frac{\left| x_{n+1} - x_n \right|}{\left| x_{n+1} \right|} \leq \varepsilon,$$

where ε is a specified tolerance value.

Example 17 *Use the secant method with* $x_0 = 1$ *and* $x_1 = 2$ *to solve* $x^3 - x^2 - 1 = 0$, *with* $\varepsilon = 10^{-4}$.

Sol. *With* $x_0 = 1$, $f(x_0) = -1$ *and* $x_1 = 2$, $f(x_1) = 3$, *we have*

$$x_2 = 2 - \frac{(2-1)(3)}{3-(-1)} = 1.25$$

from which $f(x_2) = f(1.25) = -0.609375$. *The next iterate is*

$$x_3 = 1.25 - \frac{(1.25-2)(-0.609375)}{-0.609375-3} = 1.3766234.$$

Continuing in this manner the iterations lead to the approximation 1.4655713.

Examples of other multi-step methods

1. THE BISECTION METHOD

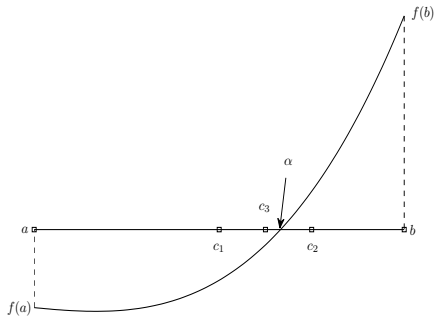
Let f be a given function, continuous on an interval $[a, b]$, such that

$$f(a)f(b) < 0. \tag{10}$$

By the Mean Value Theorem, it follows that there exists at least one zero α of f in (a, b).

The bisection method is based on halving the interval $[a, b]$ to determine a smaller and smaller interval within α must lie.

First we give the midpoint of $[a, b]$, $c = (a + b)/2$ and then compute the product $f(c)f(b)$. If the product is negative, then the root is in the interval $[c, b]$ and we take $a_1 = c$, $b_1 = b$. If the product is positive, then the root is in the interval $[a, c]$ and we take $a_1 = a$, $b_1 = c$. Thus, a new interval containing α is obtained.



Bisection method

The algorithm:

Suppose $f(a)f(b) \leq 0$. Let $a_0 = a$ and $b_0 = b$.

for $n = 0, 1, ...,$ITMAX

$c \leftarrow \frac{a_n + b_n}{2}$

if $f(a_n)f(c) \leq 0$, set $a_{n+1} = a_n, b_{n+1} = c$

else, set $a_{n+1} = c, b_{n+1} = b_n$

The process of halving the new interval continues until the root is located as accurately as desired, namely

$$\frac{|a_n - b_n|}{|a_n|} < \varepsilon,$$

where a_n and b_n are the endpoints of the n-th interval $[a_n, b_n]$ and ε is a specified precision. The approximation of the solution will be $\frac{a_n+b_n}{2}$.

Some other stopping criterions: $|a_n - b_n| < \varepsilon$ or $|f(a_n)| < \varepsilon$.

Example 18 *The function* $f(x) = x^3 - x^2 - 1$ *has one zero in* $[1, 2]$. *Use the bisection algorithm to approximate the zero of* f *with precision* 10^{-4}.

Sol. *Since* $f(1) = -1 < 0$ *and* $f(2) = 3 > 0$, *then (10) is satisfied. Starting with* $a_0 = 1$ *and* $b_0 = 2$, *we compute*

$$c_0 = \frac{a_0 + b_0}{2} = \frac{1+2}{2} = 1.5 \text{ and } f(c_0) = 0.125.$$

Since $f(1.5)f(2) > 0$, *the function changes sign on* $[a_0, c_0] = [1, 1.5]$.

To continue, we set $a_1 = a_0$ *and* $b_1 = c_0$; *so*

$$c_1 = \frac{a_1 + b_1}{2} = \frac{1 + 1.5}{2} = 1.25 \text{ and } f(c_1) = -0.609375$$

Again, $f(1.25)f(1.5) < 0$ *so the function changes sign on* $[c_1, b_1] = [1.25,\ 1.5]$. *Next we set* $a_2 = c_1$ *and* $b_2 = b_1$. *Continuing in this manner we obtain a sequence* $(c_i)_{i>0}$ *which converges to* 1.465454, *the solution of the equation.*

2. THE METHOD OF FALSE POSITION

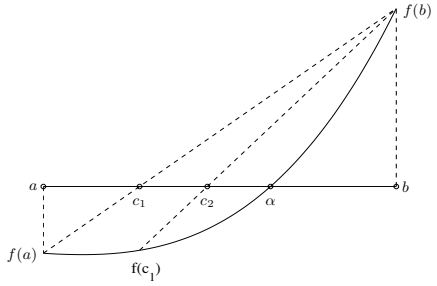
This method is also known as *regula falsi*, is similar to the Bisection method but has the advantage of being slightly faster than the latter. The function have to be continuous on $[a, b]$ with

$$f(a)f(b) < 0.$$

The point c is selected as point of intersection of the Ox-axis, and the straight line joining the points $(a,\ f(a))$ and $(b, f(b))$. From the equation of the secant line, it follows that

$$c = b - f(b)\frac{b-a}{f(b)-f(a)} = \frac{af(b) - bf(a)}{f(b)-f(a)} \tag{11}$$

Compute $f(c)$ and repeat the procedure between the values at which the function changes sign, that is, if $f(a)f(c) < 0$ set $b = c$, otherwise set $a = c$. At each step we get a new interval that contains a root of f and the generated sequence of points will eventually converge to the root.



Method of false position.

The algorithm:

Given a function f continuous on $[a_0, b_0]$, with $f(a_0)f(b_0) < 0$,

input: a_0, b_0

for $n = 0, 1, ..., ITMAX$

$c \leftarrow \frac{f(b_n)a_n - f(a_n)b_n}{f(b_n) - f(a_n)}$

if $f(a_n)f(c) < 0$, set $a_{n+1} = a_n, b_{n+1} = c$ **else** set $a_{n+1} = c, b_{n+1} = b_n$.

Stopping criterions: $|f(a_n)| \leq \varepsilon$ or $|a_n - a_{n-1}| \leq \varepsilon$, where ε is a specified tolerance value.

Remark 19 *The bisection and the false position methods converge at a very low speed compared to the secant method.*

Example 20 *The function* $f(x) = x^3 - x^2 - 1$ *has one zero in* $[1, 2]$. *Use the method of false position to approximate the zero of* f *with precision* 10^{-4}.

Sol. *A root lies in the interval* $[1, 2]$ *since* $f(1) = -1$ *and* $f(2) = 3$. *Starting with* $a_0 = 1$ *and* $b_0 = 2$, *we get using (11)*

$$c_0 = 2 - \frac{3(2-1)}{3-(-1)} = 1.25 \text{ and } f(c_0) = -0.609375.$$

Here, $f(c_0)$ *has the same sign as* $f(a_0)$ *and so the root must lie on the interval* $[c_0, b_0] = [1.25, 2]$. *Next we set* $a_1 = c_0$ *and* $b_1 = b_0$ *to get the next approximation*

$$c_1 = 2 - \frac{3-(2-1.25)}{3-(-0.609375)} = 1.37662337 \text{ and } f(c_1) = -0.2862640.$$

Now $f(x)$ *change sign on* $[c_1, b_1] = [1.37662337, 2]$. *Thus we set* $a_2 = c_1$ *and* $b_2 = b_1$. *Continuing in this manner the iterations lead to the approximation* 1.465558.

Example 21 *Compare the false position method, the secant method and Newton's method for solving the equation* $x = cosx$, *having as starting points* $x_0 = 0.5$ *şi* $x_1 = \pi/4$, *respectively* $x_0 = \pi/4$.

n	**(a)** x_n **False position**	**(b)** x_n **Secant**	**(c)** x_n **Newton**
0	0.5	0.5	0.5
1	0.785398163397	0.785398163397	0.785398163397
2	0.736384138837	0.736384138837	0.739536133515
3	0.739058139214	0.739058139214	0.739085178106
4	0.739084863815	0.739085149337	0.739085133215
5	0.739085130527	0.739085133215	
6	0.739085133188		
7	0.739085133215		

The extra condition from the false position method usually requires more computation than the secant method, and the simplifications

from the secant method come with more iterations than in the case of Newton's method.

Example 22 *Consider the equation* $x^2 - x - 3 = 0$. *Give the next two iterations for approximating the solution of this equation using:*

a) Newton's method starting with $x_0 = 0$.

b) secant, false position and bisection methods starting with $x_0 = 0$ *and* $x_1 = 4$.