# COURSE 11

## 5.2. Multistep methods for solving nonlinear eq. in $\mathbb{R}$

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 $F : D \to \Omega,\ D \subset \Omega^n$ to this equation.

Let $(x_0, ..., x_n) \in D$ be *the starting points*. We construct iteratively the sequence

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

with

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

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

In this case, the $F$-method is **a multistep method**.

It is based on interpolation methods with more than one interpolation node.

Let $\alpha \in \Omega$ be a solution of equation (1), let $(a, b) \subset \Omega$ be a neighborhood of $\alpha$ that isolates this solution and $x_0, ..., x_n \in (a, b)$, some given values.

Denote by $g$ the inverse function of $f$, assuming it exists. Because $\alpha = g(0)$, the problem reduces to approximating $g$ by an interpolation method with $n > 1$ nodes, for example Lagrange, Hermite, Birkhoff, etc...

## 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 $g(y_k)$, $k = 0, ..., n$:

$$g = L_n g + R_n g, \tag{4}$$

where

$$(L_n g)(y) = \sum_{k=0}^{n} \frac{(y-y_0)...(y-y_{k-1})(y-y_{k+1})...(y-y_n)}{(y_k-y_0)...(y_k-y_{k-1})(y_k-y_{k+1})...(y_k-y_n)} g(y_k). \tag{5}$$

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 1** *If $\alpha \in (a,b)$ is solution of equation (1), $f'$ 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*

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

*converges to $\alpha$.*

**Remark 2** *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**

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

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 $ord(F_1^L) = \frac{(1+\sqrt{5})}{2}$.

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 3** *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:*

$$f'(x) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \tag{6}$$

*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.

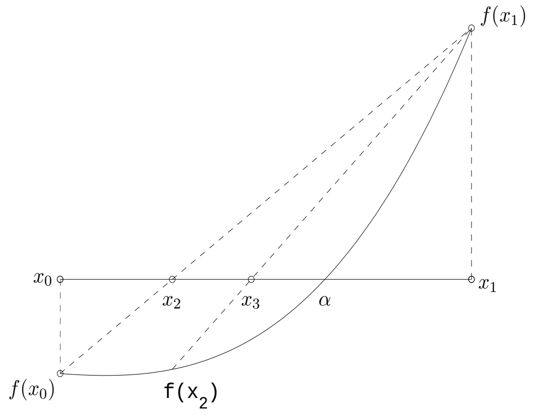
The formula for $x_{n+1}$ is obtained by setting $x = x_{n+1}$ and $y = 0$ in the equation of the secant line from $(x_{n-1}, f(x_{n-1}))$ to $(x_n, f(x_n))$:

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

we get

$$x_{n+1} = x_n - f(x_n)\left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right]. \tag{7}$$

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 $\varepsilon$ is a specified tolerance value.

**Example 4** *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

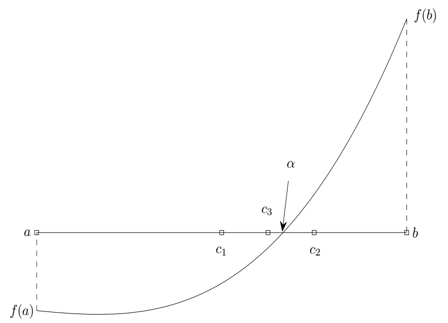
Let $f$ be a given function, continuous on an interval $[a, b]$, such that

$$f(a)f(b) < 0. \tag{8}$$

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

The bisection method is based on halving the interval $[a, b]$ to determine a smaller and smaller interval within $\alpha$ 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 $\alpha$ is obtained.



Bisection method

**The algorithm:**

Suppose $f(a)f(b) \leq 0$. Let $a_o = a$ and $b_o = b$.

```
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 $\varepsilon$ 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 5** *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 (8) 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

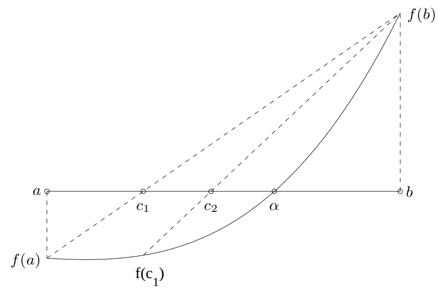
This method is also known as *regula falsi*, is similar to the Bisection method but has the advantage of being slightly faster than the latter. The function have to be continuous on $[a, b]$ with

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

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

$$c = b - f(b)\frac{b-a}{f(b)-f(a)} = \frac{af(b)-bf(a)}{f(b)-f(a)} \tag{9}$$

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 $\varepsilon$ is a specified tolerance value.

**Remark 6** *The bisection and the false position methods converge at a very low speed compared to the secant method.*

**Example 7** *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 (9)*

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

*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*

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

*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 8** *Compare the false position method, the secant method and Newton's method for solving the equation $x = cosx$, having as starting points $x_0 = 0.5$ și $x_1 = \pi/4$, 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 | 0.739085133215 |
| 6 | 0.739085133188 | 0.739085133215 | |
| 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.

## Hermite inverse interpolation

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

$$f\,(x)=0,\quad x\in\Omega. \tag{10}$$

Assume that $\alpha$ is a solution of equation $f(x)=0$ and $V\,(\alpha)$ is a neighborhood of $\alpha$. If $y_k=f(x_k)$, where $x_k\in V(\alpha)$, $k=0,...,m$, are approximations of $\alpha$, $r_k\in\mathbb{N}$, then if there exist $g^{(j)}(y_k)=(f^{-1})^{(j)}(y_k), j=0,...,r_k$, one considers the Hermite type interpolation problem.

**Theorem 9** *Let $\alpha$ be a solution of equation $f(x)=0$, $V(\alpha)$ a neighborhood of $\alpha$ and $x_0,x_1...,x_m\in V(\alpha)$. For $n=r_0+...+r_m+m$, where $r_k$ represents the multiplicity order of the nodes $x_k$, $k=0,...,m$, if*

*$f \in C^{n+1}(V(\alpha))$ and $f'(x) \neq 0$ for $x \in V(\alpha)$, we have the following Hermite approximation method for $\alpha$ :*

$$\alpha \approx F_n^H(x_0, ..., x_m) = \tag{11}$$

$$= (H_n g)\,(0) = \sum_{k=0}^{m} \sum_{j=0}^{r_k} \sum_{\nu=0}^{r_k-j} \frac{(-1)^{j+\nu}}{j!\nu!} f_k^{j+\nu} v_k(0) (\tfrac{1}{v_k(y)})^{(\nu)}_{y=f_k} g^{(j)}(f_k),$$

*where $f_k = f(x_k), k = 0, ..., m, g = f^{-1}$, and*

$$v_k(y) = (y - f_0)^{r_0+1} ... (y - f_{k-1})^{r_{k-1}+1} (y - f_{k+1})^{r_{k+1}+1} ... (y - f_m)^{r_m+1}.$$

For $g = f^{-1}$ the corresponding Hermite polynomial is

$$(H_n g)(y) = \sum_{k=0}^{m} \sum_{j=0}^{r_k} h_{kj}(y) g^{(j)}(y_k),$$

and it satisfies the conditions:

$$(H_n g)^{(j)}(y_k) = g^{(j)}(y_k), \;\; j = 0, ..., r_k; \quad k = 0, ..., m,$$

where $h_{kj}$ are the fundamental interpolating polynomials, i.e.,

$$h_{kj}^{(p)}(y_\nu) = 0,\ k \neq \nu,\ p = 0, ..., r_\nu$$
$$h_{kj}^{(p)}(y_k) = \delta_{pj},\ p = 0, ..., r_k$$

and the corresponding interpolation formula is

$$g = H_n g + R_n g,$$

where $R_n g$ is the remainder term.

Taking into account that

$$\alpha = g(0) \approx (H_n g)(0),$$

defines a new approximation to $\alpha$, we have that

$$F_n^H(x_0, ..., x_m) = (H_n g)(0)$$

is an approximation method for $\alpha$.

Regarding the order of Hermite-type inverse interpolation method $F_n^H$, we have two results, first for the case of equal information (the same multiplicity order for all the nodes $x_k$, $k = 0, ..., m$) and then for different multiplicities.

**Theorem 10** *(Equal information) The order* $ord(F_n^H)$ *is the unique positive root of the equation:*

$$t^{m+1} - (r+1)\sum_{j=0}^{m} t^j = 0,$$

*where* $r$ *is the multiplicity order of the points* $x_k$*,* $\forall k = 0, ..., m$.

**Theorem 11** *(Unequal information) The order of* $F_n^H$ *is the unique positive and real root of the equation:*

$$t^{m+1} - (r_m + 1)t^m - (r_{m-1} + 1)t^{m-1} - .... - (r_1 + 1)t - (r_0 + 1) = 0,$$

*where* $r_0, ..., r_m$ *are real numbers, permutation of the multiplicity orders of the nodes* $x_k$*,* $k = 0, ..., m$ *satisfying the conditions:*

$$r_0 + r_1 + ... + r_m > 1 \qquad (12)$$

*and*

$$r_m \geq r_{m-1} \geq ... \geq r_1 \geq r_0. \tag{13}$$

**Remark 12** *The order of the Taylor-type inverse interpolation method, can be expressed as the solution of equation*

$$t-(r_0+1)=0,$$

*where $r_0$ is the multiplicity order of the node $x_0$.*

**Particular cases.**

1) For $x_0$, $x_1 \in V(\alpha)$ with $r_0 = 0$, $r_1 = 1$, we have the following approximation method:

$$F_2^H(x_0,x_1) = x_1 - \left[\frac{f(x_1)}{f(x_0)-f(x_1)}\right]^2 (x_1-x_0) - \frac{f(x_1)}{f(x_0)-f(x_1)}\frac{f(x_0)}{f\,'(x_1)}.$$

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

$$t^2 - r_1 t - r_0 = 0,$$

so,

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

and $p = 1 + \sqrt{2}$.

2) For nodes $x_0$, $x_1$, $x_2 \in V(\alpha)$ with $r_0 = r_1 = 0$; $r_2 = 1$, the method is:

$$F_4^H(x_0, x_1, x_2) =$$
$$= \frac{f(x_2)^2}{f(x_1)-f(x_0)} \left[ \frac{x_0 f(x_1)}{[f(x_0)-f(x_2)]^2} - \frac{x_1 f(x_0)}{[f(x_1)-f(x_2)]^2} \right]$$
$$+ \frac{f(x_0)f(x_1)}{[f(x_2)-f(x_0)][f(x_2)-f(x_1)]} \left[ 1 + \frac{f(x_2)}{[f(x_2)-f(x_0)][f(x_2)-f(x_1)]} \right] \left[ x_2 - \frac{f(x_2)}{f'(x_2)} \right].$$

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

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

so $p = 2.548$.

3) For $x_0$, $x_1 \in V(\alpha)$ with $r_0 = r_1 = 1$, (double nodes), the approximation method is

$$F_3^H(x_0, x_1) = x_1 - \frac{3f(x_0)f(x_1)^2 - f(x_1)^3}{[f(x_0) - f(x_1)]^3}(x_0 - x_1) + \frac{f(x_0)f(x_1)}{[f(x_0) - f(x_1)]^2}\left[\frac{f(x_1)}{f\,'(x_0)} - \frac{f(x_0)}{f\,'(x_1)}\right].$$

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

$$t^{m+1} - (r+1)\sum_{j=0}^{m} t^j = 0,$$

so, $t^2 - 2t - 2 = 0$, and $p = 1 + \sqrt{3}$.