COURSE 10

4.3. Iterative methods for solving linear systems

Because of round-off errors, direct methods become less efficient than iterative methods for large systems (>100 000 variables).

An iterative scheme for linear systems consists of converting the system

$$Ax = b \tag{1}$$

to the form

$$x = \tilde{b} - Bx.$$

After an initial guess for $x^{(0)}$, the sequence of approximations of the solution $x^{(0)}, x^{(1)}, ..., x^{(k)}, ...$is generated by computing

$$x^{(k)} = \tilde{b} - Bx^{(k-1)}, \quad \text{for } k = 1, 2, 3, ...$$

4.3.1. Jacobi iterative method

Consider the $n \times n$ linear system,

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ \qquad\qquad\qquad ... \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n, \end{cases}$$

where we assume that the diagonal terms a_{11}, a_{22}, ..., a_{nn} are all nonzero.

We begin our iterative scheme by solving each equation for one of the variables:

$$\begin{cases} x_1 = u_{12}x_2 + ... + u_{1n}x_n + c_1 \\ x_2 = u_{21}x_1 + ... + u_{2n}x_n + c_2 \\ ... \\ x_n = u_{n1}x_1 + ... + u_{nn-1}x_{n-1} + c_n, \end{cases}$$

where $u_{ij} = -\frac{a_{ij}}{a_{ii}}$, $c_i = \frac{b_i}{a_{ii}}$, $i = 1, ..., n$.

Let $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)})$ be an initial approximation of the solution. The $k+1$-th approximation is:

$$\begin{cases} x_1^{(k+1)} = u_{12}x_2^{(k)} + ... + u_{1n}x_n^{(k)} + c_1 \\ x_2^{(k+1)} = u_{21}x_1^{(k)} + u_{23}x_3^{(k)} + ... + u_{2n}x_n^{(k)} + c_2 \\ ... \\ x_n^{(k+1)} = u_{n1}x_1^{(k)} + ... + u_{nn-1}x_{n-1}^{(k)} + c_n, \end{cases}$$

for $k = 0, 1, 2, ...$

An algorithmic form:

$$x_i^{(k)} = \frac{b_i - \sum\limits_{j=1, j\neq i}^{n} a_{ij}x_j^{(k-1)}}{a_{ii}}, \ i = 1, 2, ..., n, \ \text{for } k \geq 1.$$

The iterative process is terminated when a convergence criterion is satisfied.

Stopping criterions: $\left|x^{(k)} - x^{(k-1)}\right| < \varepsilon$ or $\frac{\left|x^{(k)} - x^{(k-1)}\right|}{\left|x^{(k)}\right|} < \varepsilon$, with $\varepsilon > 0$ - a prescribed tolerance.

Example 1 *Solve the following system using the Jacobi iterative method. Use* $\varepsilon = 10^{-3}$ *and* $x^{(0)} = (0\ 0\ 0\ 0)$ *as the starting vector.*

$$\begin{cases} 7x_1 - 2x_2 + x_3 = 17 \\ x_1 - 9x_2 + 3x_3 - x_4 = 13 \\ 2x_1 + 10x_3 + x_4 = 15 \\ x_1 - x_2 + x_3 + 6x_4 = 10. \end{cases}$$

These equations can be rearranged to give

$$\begin{aligned} x_1 &= (17 + 2x_2 - x_3)/7 \\ x_2 &= (-13 + x_1 + 3x_3 - x_4)/9 \\ x_3 &= (15 - 2x_1 - x_4)/10 \\ x_4 &= (10 - x_1 + x_2 - x_3)/6 \end{aligned}$$

and, for example,

$$\begin{aligned} x_1^{(1)} &= (17 + 2x_2^{(0)} - x_3^{(0)})/7 \\ x_2^{(1)} &= (-13 + x_1^{(0)} + 3x_3^{(0)} - x_4^{(0)})/9 \\ x_3^{(1)} &= (15 - 2x_1^{(0)} - x_4^{(0)})/10 \\ x_4^{(1)} &= (10 - x_1^{(0)} + x_2^{(0)} - x_3^{(0)})/6. \end{aligned}$$

Substitute $x^{(0)} = (0, 0, 0, 0)$ *into the right-hand side of each of these equations to get*

$$x_1^{(1)} = (17 + 2 \cdot 0 - 0)/7 = 2.428\ 571\ 429$$
$$x_2^{(1)} = (-13 + 0 + 3 \cdot 0 - 0)/9 = -1.444\ 444\ 444$$
$$x_3^{(1)} = (15 - 2 \cdot 0 - 0)/10 = 1.5$$
$$x_4^{(1)} = (10 - 0 + 0 - 0)/6 = 1.666\ 666\ 667$$

and so $x^{(1)} = (2.428\ 571\ 429, -1.444\ 444\ 444, 1.5, 1.666\ 666\ 667)$. *The Jacobi iterative process:*

$$x_1^{(k+1)} = \left(17 + 2x_2^{(k)} - x_3^{(k)}\right)/7$$
$$x_2^{(k+1)} = \left(-13 + x_1^{(k)} + 3x_3^{(k)} - x_4^{(k)}\right)/9$$
$$x_3^{(k+1)} = \left(15 - 2x_1^{(k)} - x_4^{(k)}\right)/10$$
$$x_4^{(k+1)} = \left(10 - x_1^{(k)} + x_2^{(k)} - x_3^{(k)}\right)/6, \qquad k \geq 1.$$

We obtain a sequence that converges to

$$\mathbf{x}^{(9)} = (2.000127203, -1.000100162, 1.000118096, 1.000162172).$$

4.3.2. Gauss-Seidel iterative method

Almost the same as Jacobi method, except that each x-value is improved using the most recent approx. of the other variables.

For a $n \times n$ system, the $k+1$-th approximation is:

$$\begin{cases} x_1^{(k+1)} = u_{12}x_2^{(k)} + ... + u_{1n}x_n^{(k)} + c_1 \\ x_2^{(k+1)} = u_{21}x_1^{(k+1)} + u_{23}x_3^{(k)} + ... + u_{2n}x_n^{(k)} + c_2 \\ ... \\ x_n^{(k+1)} = u_{n1}x_1^{(k+1)} + ... + u_{nn-1}x_{n-1}^{(k+1)} + c_n, \end{cases}$$

with $k = 0, 1, 2, ...$; $u_{ij} = -\frac{a_{ij}}{a_{ii}}$, $c_i = \frac{b_i}{a_{ii}}$, $i = 1, ..., n$ (as in Jacobi method).

Algorithmic form:

$$x_i^{(k)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}}{a_{ii}}$$

for each $i = 1, 2, ...n,$ and for $k \geq 1$.

Stopping criterions: $\left|x^{(k)} - x^{(k-1)}\right| < \varepsilon$, or $\frac{\left|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}\right|}{\left|\mathbf{x}^{(k)}\right|} < \varepsilon$, with ε - a prescribed tolerance, $\varepsilon > 0$.

Remark 2 *Because the new values can be immediately stored in the location that held the old values, the storage requirements for* $\mathbf{x}$ *with the Gauss-Seidel method is half than that for Jacobi method and the rate of convergence is faster.*

Example 3 *Solve the following system using the Gauss-Seidel iterative method. Use* $\varepsilon = 10^{-3}$ *and* $\mathbf{x}^{(0)} = (0\ 0\ 0\ 0)$ *as the starting vector.*

$$\begin{cases} 7x_1 - 2x_2 + x_3 = 17 \\ x_1 - 9x_2 + 3x_3 - x_4 = 13 \\ 2x_1 + 10x_3 + x_4 = 15 \\ x_1 - x_2 + x_3 + 6x_4 = 10 \end{cases}$$

We have

$$
\begin{aligned}
x_1 &= (17 + 2x_2 - x_3)/7 \\
x_2 &= (-13 + x_1 + 3x_3 - x_4)/9 \\
x_3 &= (15 - 2x_1 - x_4)/10 \\
x_4 &= (10 - x_1 + x_2 - x_3)/6,
\end{aligned}
$$

and, for example,

$$
\begin{aligned}
x_1^{(1)} &= (17 + 2x_2^{(0)} - x_3^{(0)})/7 \\
x_2^{(1)} &= (-13 + x_1^{(1)} + 3x_3^{(0)} - x_4^{(0)})/9 \\
x_3^{(1)} &= (15 - 2x_1^{(1)} - x_4^{(0)})/10 \\
x_4^{(1)} &= (10 - x_1^{(1)} + x_2^{(1)} - x_3^{(1)})/6,
\end{aligned}
$$

which provide the following Gauss-Seidel iterative process:

$$x_1^{(k+1)} = \left(17 + 2x_2^{(k)} - x_3^{(k)}\right)/7$$
$$x_2^{(k+1)} = \left(-13 + x_1^{(k+1)} + 3x_3^{(k)} - x_4^{(k)}\right)/9$$
$$x_3^{(k+1)} = \left(15 - 2x_1^{(k+1)} - x_4^{(k)}\right)/10$$
$$x_4^{(k+1)} = \left(10 - x_1^{(k+1)} + x_2^{(k+1)} - x_3^{(k+1)}\right)/6, \quad \textit{for } k \geq 1.$$

Substitute $\mathbf{x}^{(0)} = (0, 0, 0, 0)$ *into the right-hand side of each of these equations to get*

$$x_1^{(1)} = (17 + 2 \cdot 0 - 0)/7 = 2.428\ 571\ 429$$
$$x_2^{(1)} = (-13 + 2.428\ 571\ 429 + 3 \cdot 0 - 0)/9 = -1.1746031746$$
$$x_3^{(1)} = (15 - 2 \cdot 2.428\ 571\ 429 - 0)/10 = 1.0142857143$$
$$x_4^{(1)} = (10 - 2.428\ 571\ 429 - 1.1746031746 - 1.0142857143)/6$$
$$= 0.8970899472$$

and so

$$\mathbf{x}^{(1)} = (2.428571429 - 1.1746031746, 1.0142857143, 0.8970899472).$$

Similar procedure generates a sequence that converges to

$$\mathbf{x}^{(5)} = (2.000025, -1.000130, 1.000020.0.999971).$$

4.3.3. Relaxation method

In case of convergence, the Gauss-Seidel method is faster than Jacobi method. The convergence can be more improved using **relaxation method (SOR method) (**SOR=Succesive Over Relaxation)

Algorithmic form of the method:

$$x_i^{(k)} = \frac{\omega}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) + (1-\omega) x_i^{(k-1)}$$

for each $i = 1, 2, ...n,$ and for $k \geq 1$.

For $0 < \omega < 1$ the procedure is called **under relaxation method,** that can be used to obtain convergence for systems which are not convergent by Gauss-Siedel method.

For $\omega > 1$ the procedure is called **over relaxation method,** that can be used to accelerate the convergence for systems which are convergent by Gauss-Siedel method.

By Kahan's Theorem follows that the method converges for $0 < \omega < 2$.

Remark 4 *For $\omega = 1$, relaxation method is Gauss-Seidel method.*

Example 5 *Solve the following system, using relaxation iterative method. Use $\varepsilon = 10^{-3}$, $\mathbf{x}^{(0)} = (1\ 1\ 1)$ and $\omega = 1.25$,*

$$\begin{array}{rcrcrcr} 4x_1 & + & 3x_2 & & & = & 24 \\ 3x_1 & + & 4x_2 & - & x_3 & = & 30 \\ & - & x_2 & + & 4x_3 & = & -24 \end{array}$$

We have

$$\begin{aligned} x_1^{(k)} &= 7.5 - 0.937x_2^{(k-1)} - 0.25x_1^{(k-1)} \\ x_2^{(k)} &= 9.375 - 9.375x_1^{(k)} + 0.3125x_3^{(k-1)} - 0.25x_2^{(k-1)} \\ x_3^{(k)} &= -7.5 + 0.3125x_2^{(k)} - 0.25x_3^{(k-1)}, \quad \textit{for } k \geq 1. \end{aligned}$$

The solution is $(3, 4, -5)$.

4.3.4 The matriceal formulations of the iterative methods

Split the matrix A into the sum

$$A = D + L + U,$$

where D is the diagonal of A, L the lower triangular part of A, and U the upper triangular part of A. That is,

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}, \quad L = \begin{bmatrix} 0 & \cdots & & 0 \\ a_{21} & \ddots & & \ddots \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix},$$

$$U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ 0 & \cdots & & 0 \end{bmatrix}$$

The system $Ax = b$ can be written as

$$(D + L + U)\mathbf{x} = \mathbf{b}.$$

The **Jacobi method** in matriceal form is given by:

$$D\mathbf{x}^{(k)} = -(L+U)\mathbf{x}^{(k-1)} + \mathbf{b}$$

the **Gauss-Seidel method** in matriceal form is given by:

$$(D+L)\mathbf{x}^{(k)} = -U\mathbf{x}^{(k-1)} + \mathbf{b}$$

and **the relaxation method** in matriceal form is given by:

$$(D+\omega L)\mathbf{x}^{(k)} = ((1-\omega)D - \omega U)\mathbf{x}^{(k-1)} + \omega\mathbf{b}$$

Convergence of the iterative methods

Remark 6 *The convergence (or divergence) of the iterative process in the Jacobi and Gauss-Seidel methods does not depend on the initial guess, but depends only on the character of the matrices themselves. However, a good first guess in case of convergence will make for a relatively small number of iterations.*

A sufficient condition for convergence:

Theorem 7 (**Convergence Theorem**) *If A is strictly diagonally dominant, then the Jacobi, Gauss-Seidel and relaxation methods converge for any choice of the starting vector $\mathbf{x}^{(0)}$.*

Example 8 *Consider the system of equations*

$$\begin{bmatrix} 3 & 1 & 1 \\ -2 & 4 & 0 \\ -1 & 2 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix}.$$

The coefficient matrix of the system is strictly diagonally dominant since

$$\begin{aligned} |a_{11}| &= |3| = 3 > |1| + |1| = 2 \\ |a_{22}| &= |4| = 4 > |-2| + |0| = 2 \\ |a_{33}| &= |-6| = 6 > |-1| + |2| = 3. \end{aligned}$$

Hence, if the Jacobi or Gauss-Seidel method are used to solve the system of equations, they will converge for any choice of the starting vector $\mathbf{x}^{(0)}$.

Example 9 *Consider the linear system*

$$4x_1 + x_2 = 3$$
$$2x_1 + 5x_2 = 1.$$

a) Perform two iterations of the Jacobi method to this system, beginning with the vector $x = [3, 11]$.

b) Perform two iterations of the Gauss-Seidel method to this system, beginning with the vector $x = [3, 11]$.

(Solutions of the system are $7/9$ *and* $-1/9$*).*

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

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

$$f(x) = 0, \quad x \in \Omega. \tag{2}$$

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

$$x_0, x_1, ..., x_{n-1}, x_n, ... \tag{3}$$

with

$$x_i = F(x_{i-n}, ..., x_{i-1}), \quad i = n, \dots. \tag{4}$$

The problem consists in choosing F and $x_0, ..., x_{n-1} \in D$ such that the sequence (3) to be convergent to the solution of the equation (2).

Definition 10 *The procedure of approximation the solution of equation (2) by the elements of the sequence (3), computed as in (4), is called* F**-method.**

The numbers $x_0, x_1, ..., x_{n-1}$ *are called* **the starting points** *and the* k*-th element of the sequence (3) is called an approximation of* k*-th order 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 11 *If the sequence (3) converges to the solution of the equation (2) then the* F*-method is convergent, otherwise it is divergent.*

Definition 12 *Let $\alpha \in \Omega$ be a solution of the equation (2) 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*

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

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

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

Let $\alpha \in \Omega$ be a solution of the equation (2) 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 13 *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 14 *If $p = 1$ the convergence condition is $|F'(x)| < 1$.*

If $p > 1$ there always exists a neighborhood of α where the F-method converges.

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

Theorem 15 *Let α be a solution of equation (2), $V(\alpha)$ a neighborhood of α, $x, x_i \in V(\alpha)$, f fulfills the necesary continuity conditions.*

Then we have the following method, denoted by F_m^T, for approximating α:

$$F_m^T(x_i) = x_i + \sum_{k=1}^{m-1} \frac{(-1)^k}{k!}[f(x_i)]^k g^{(k)}(f(x_i)), \tag{5}$$

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

Remark 17 *We have an upper bound for the absolute error in approximating* α *by* x_{i+1} :

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

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 18 *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 (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

According to Remark 14, there always 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 (2) and $x_{n+1} = F_2^T(x_n)$, for approximation error, Remark 17 gives

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

Lemma 19 *Let $\alpha \in (a,b)$ be a solution of equation (2) and let $x_n = F_2^T(x_{n-1})$. Then*

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

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

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

■

In practical applications the following evaluation is more useful:

Lemma 20 *If* $f \in C^2[a,b]$ *and* F_2^T *is convergent, then there exists* $n_0 \in \mathbb{N}$ *such that*

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

Remark 21 *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

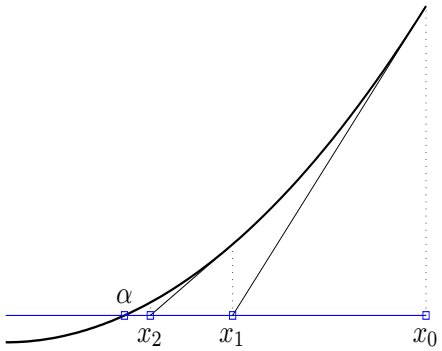
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

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

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \ n \geq 0. \tag{6}$$



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 22 *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.