Chapter 1.

Nonlinear and transcendental algebraic equations

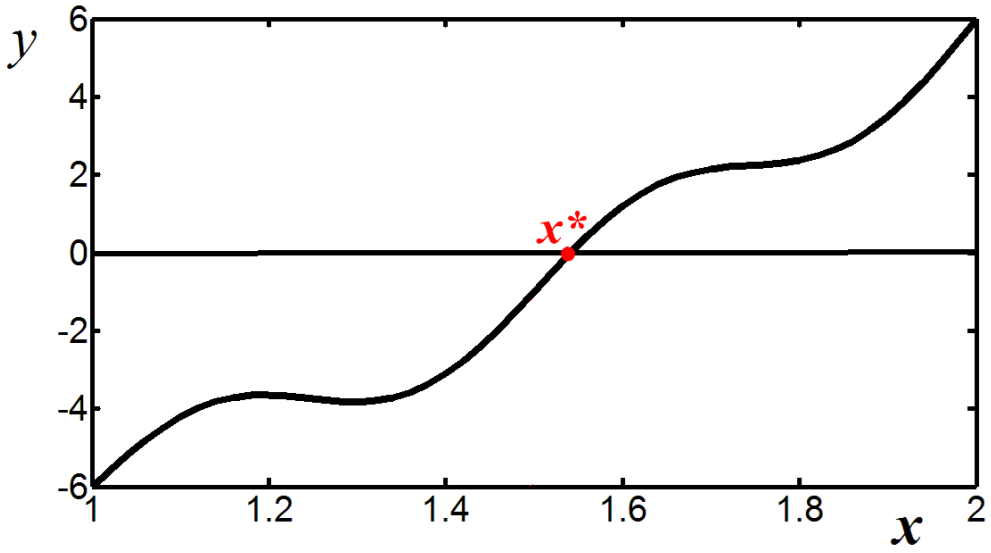
Problem:

if there exists a true/exact solution x^*, such that $f(x^*)=0$, then which way can we calculate an approximate value of x^* , admitting a small error, say ± 0.0001 ?

x^* is called a root of the equation

Theorem. If the function $y=f(x)$ is continuous on $a \le x \le b$, and the signs of $f(a)$ and $f(b)$ are opposite, then there exists x^* such that $f(x^*)=0$.

(for a proof, see course of Math. Analysis)



$$4x^2 + \sin(4\pi x) - 10 = 0, \quad 1 \le x \le 2$$

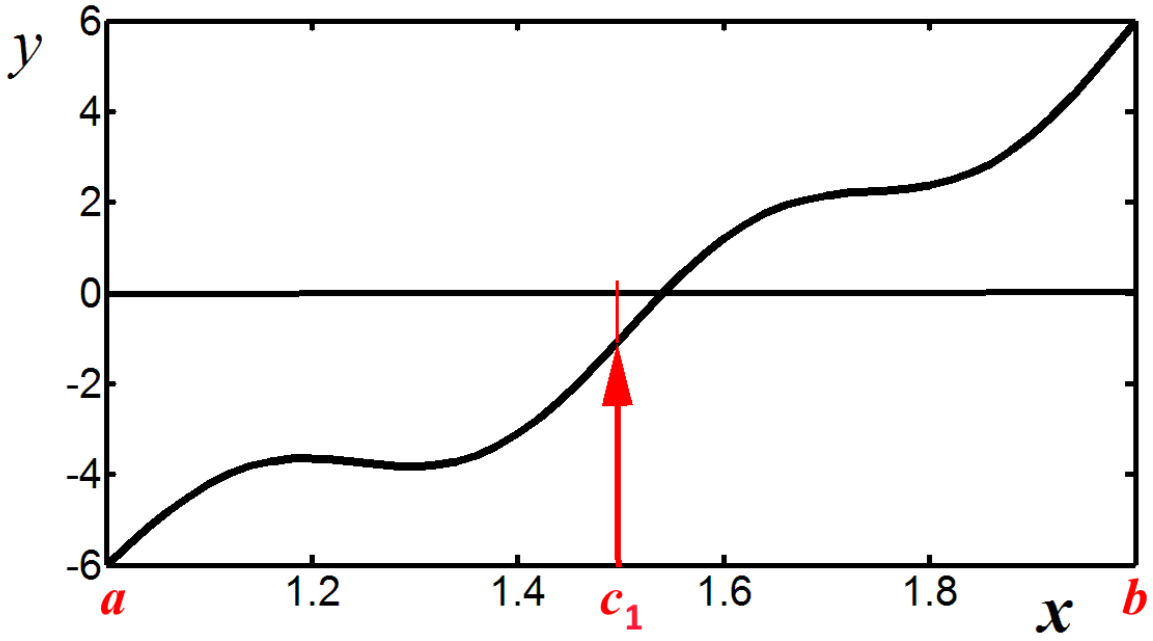
The solution exists!

1) Bysection method for calculation of x^*

It consists of repeatedly bisecting the given interval, and then selecting subinterval in which the function changes sign, and therefore must contain a root.

The method is also called the **interval halving** method, and the **dichotomy method**.

At each step the method divides the interval in two parts by computing the midpoint $c_1 = (a+b)/2$ and then selecting the part in which function $f(x)$ changes sign.



Algorithm of calculations: Suppose $f(a)<0,\ f(b)>0.$

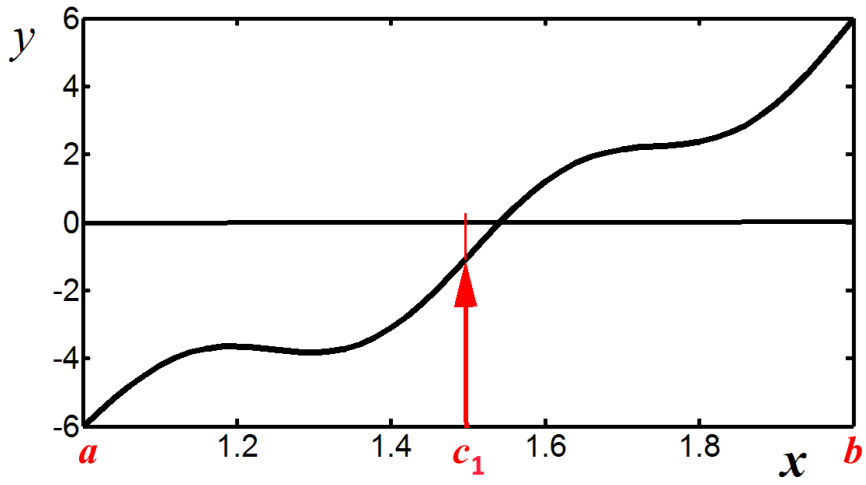
Calculate $c_1=(a+b)/2$ *and* $f(c_1)$

If $f(c_1)<0$, then root is in the right subsegment,

therefore we denote $a_2=c_1$, $b_2=b$

If $f(c_1)>0$, then we denote

$a_2= a$, $b_2= c_1$



Then procedure repeats:

$c_k=(a_k+b_k)/2$, $k= 2,\ldots$

If $f(c_k)<0$, then $a_{k+1}= c_k$, $b_{k+1}= b_k$

If $f(c_k)>0$, then $a_{k+1}= a_k$, $b_{k+1}= c_k$

(If $f(c_k)=0$, then c_k *is a root*).

At each step, $f(a_k)<0$, $f(b_k)>0$. The procedure continues until the subinterval is sufficiently small.

Length of the subinterval $b_k - a_k = (b - a) / 2^{k-1}$

shows a maximum error in the calculated approximate solution c_k **:**

$$| c_k - x^* | \leq (b - a) / 2^k$$

where x^* **is the "true" solution.**

k**=10 =>** $(b-a)/1024$

k**=20 =>** $(b-a)/1048576$

In engineering problems, typically, the error (tolerance) of 0.0001 or 0.00001 is O.K.

That is, calculations can be stopped if in successive approximations, c_k **and** c_{k+1} **, 4 or 5 digits to the right of decimal point remain the same after rounding.**

Now we consider another method, which often needs smaller number of steps for obtaining a solution of the same accuracy

2) Method of chords

Suppose that the curve

$y=f(x)$ is convex: $f''(x)>0$

Draw a straight segment (chord) connecting its endpoints:

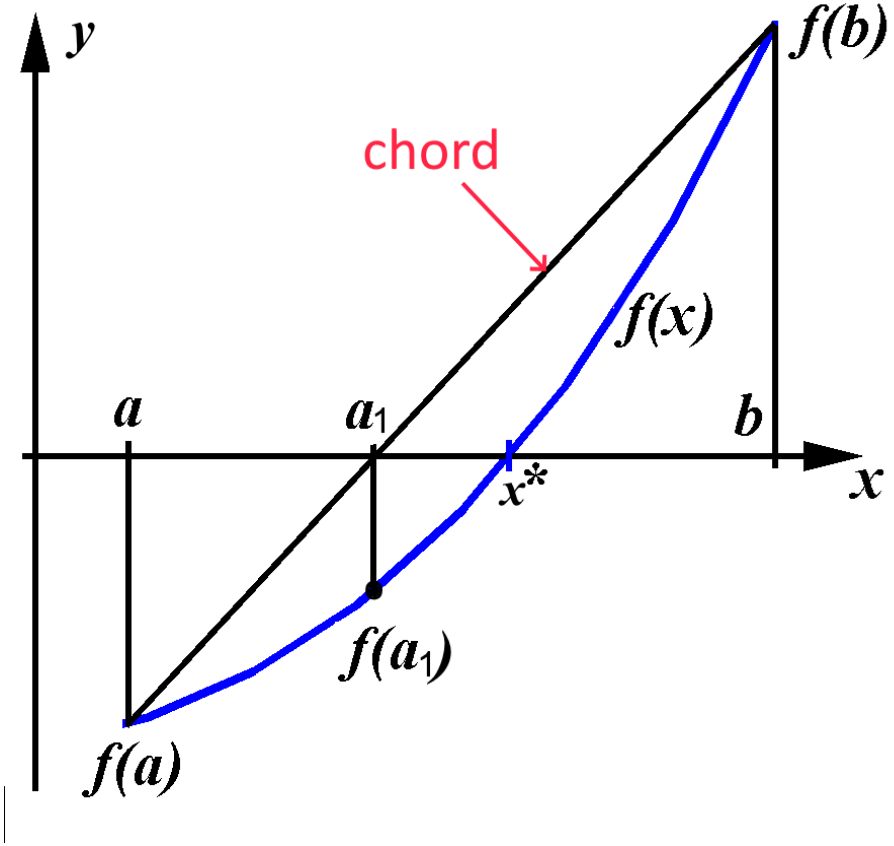
$$y=f(a)+(x-a)[f(b)-f(a)]/(b-a)$$

Find intersection of the segment with the axis $y=0$:

$$0=f(a)+(a_1-a)[f(b)-f(a)]/(b-a)$$

$$-(b-a)f(a)=(a_1-a)[f(b)-f(a)]$$

$$a_1-a=-(b-a)f(a)/[f(b)-f(a)]$$



$a_1 = a - (b-a)f(a)/[f(b)-f(a)]$

- first step towards the solution

$a_2 = a_1 - (b-a_1)f(a_1)/[f(b)-f(a_1)]$ **- second step**

$a_{k+1} = a_k - (b-a_k)f(a_k)/[f(b)-f(a_k)]$ $k=1,2,\ldots$

We get a sequence of approximate solutions

$$a_k \rightarrow x^*$$

the sequence is monotonously increasing as $f(a_k)<0$

Example: let's solve the equation

$$e^x + 2x^2 = 2, \quad 0 \le x \le 1$$

3) Method of tangent lines (Newton's method)

The same problem: solve

$$f(x)=0\ , \quad a \le x \le b$$

Suppose curve $y=f(x)$ *is convex:* $f''(x)>0$

Line tangent to curve at $x=b, y=f(b)$:

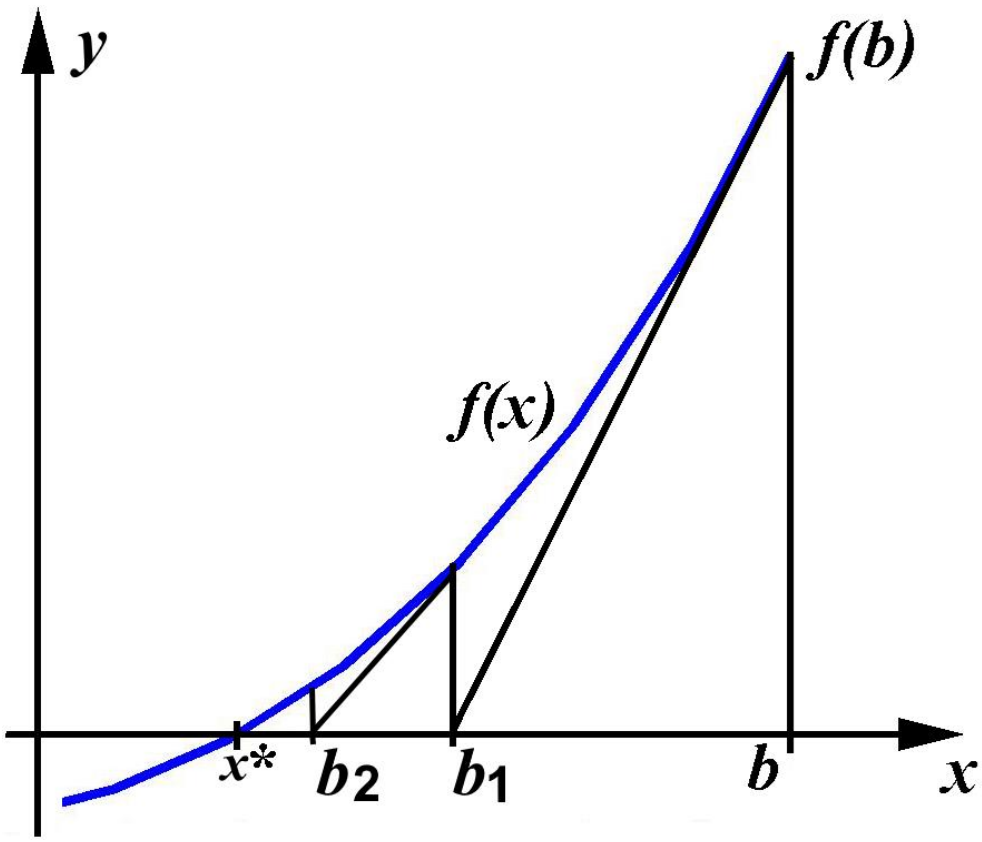
$$y=f(b)+(x-b)f'(b)$$

intersection: $0=f(b)+(b_1-b)f'(b)$

$$-f(b)=(b_1-b)f'(b)$$

$$-f(b)/f'(b)=(b_1-b)$$

$b_1=b-f(b)/f'(b)$ - *first step towards solution,*



$b_1 = b - f(b)/f'(b)$ - *first step towards solution,*

$b_2 = b_1 - f(b_1)/f'(b_1)$ $\qquad$ $b_{k+1} = b_k - f(b_k)/f'(b_k),\ k=2, \ldots$

sequence is monotonously decreasing

Example: finding square root

$$x^2 = d$$

$$f(x) = x^2 - d \qquad f'(x) = 2x$$

$$b_{k+1} = b_k - (b_k^2 - d)/(2b_k)$$

$$b_{k+1} = b_k - b_k/2 + d/(2b_k) \qquad b_{k+1} = (b_k + d/b_k)/2$$

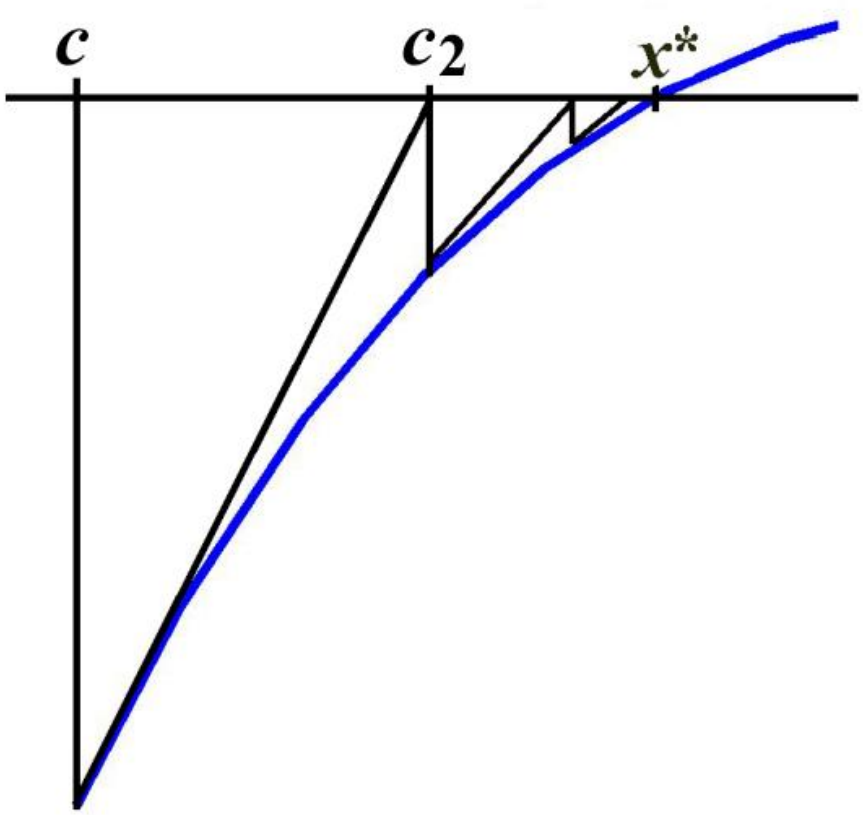
initial b – arbitrary value

Example: $e^x + 2x^2 - 2 = 0$

(same as in the method of chords)

If curve y=$f(x)$ is concave:

$f''(x)<0$



then the left endpoint a of the given segment is recommended as initial point for the start of iterations.

We have the same formula

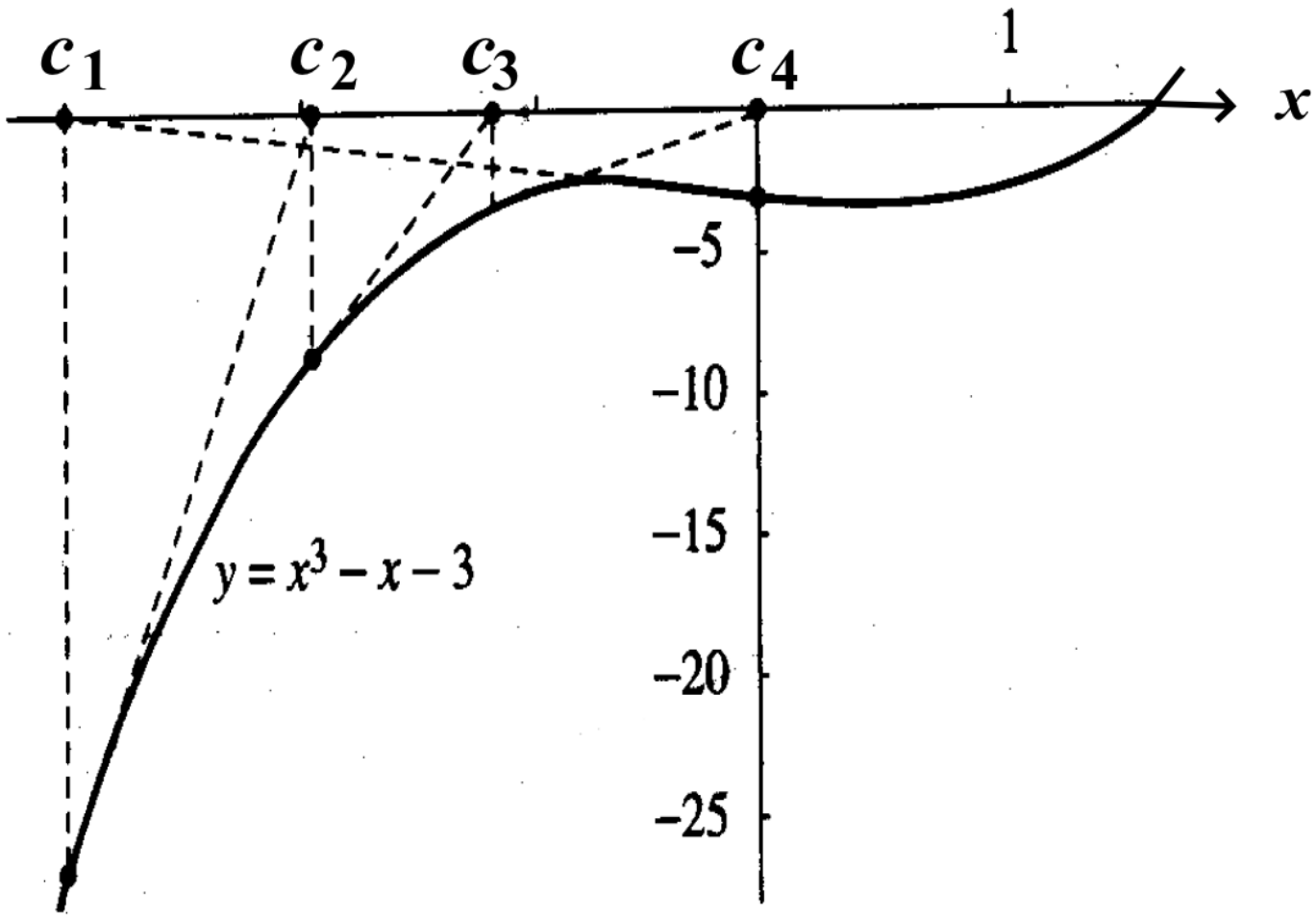
$$c_{k+1}=c_k - f(c_k)/ f'(c_k),$$

where $f(c_k)<0$; therefore, the sequence c_k is monotonously increasing.

Sometimes, the method does not work (bad cases) :

(*) If $f'(c_k)=0$ for some k, then the method can no longer be applied.

(**) Iterations can stuck in a cycle:



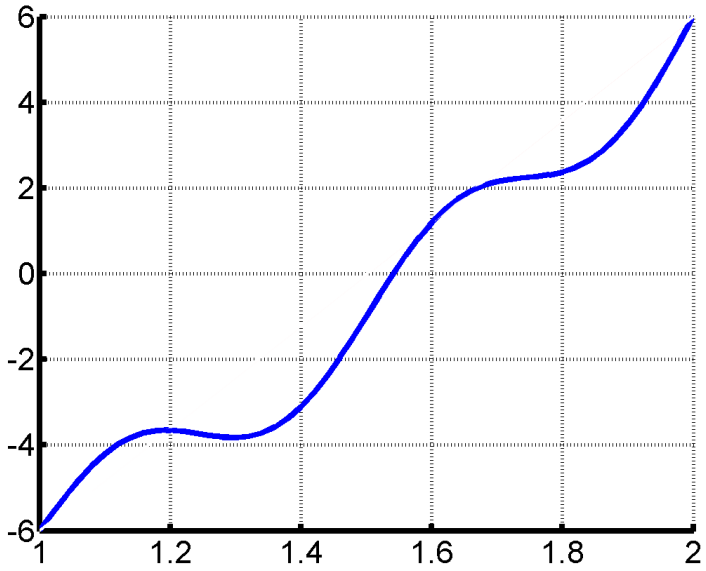
The case f(a)>0 and f(b)<0 can be reduced to the previous one by multiplying the equation by -1 :

$$-f(x)=0$$

Notice on the number of roots:

$$4x^2 + \sin(4\pi x) - 10 = 0 \,, \qquad 1 \le x \le 2$$

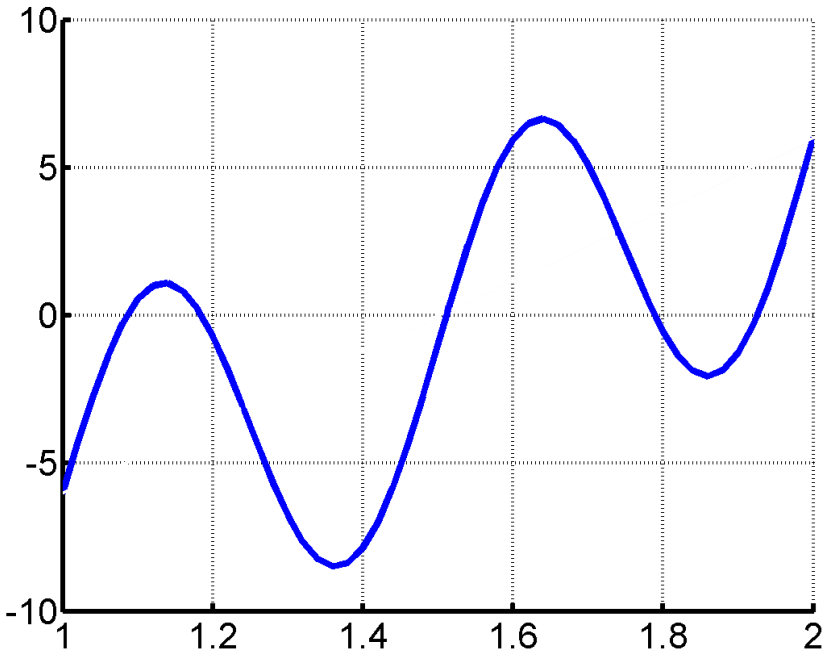
1 root:



Possibly, there exist non-unique roots

$$4x^2 + 6*\sin(4\pi x) - 10 = 0, \quad 1 \le x \le 2$$

5 roots:



It is recommended to “separate” roots by plotting a graph

4) Iteration method

Example 1.

$x - 0.1 \sin x - 2 = 0$

$x = 0.1 \sin x + 2$

$c_1 = 2$ - initial approximation

$c_2 = 0.1 \sin c_1 + 2 = 2.0909297$

$c_3 = 0.1 \sin c_2 + 2 = 2.0867753$

$c_4 = 0.1 \sin c_3 + 2 = 2.0869810$

$c_5 = 0.1 \sin c_4 + 2 = 2.0869709$

$c_6 = 0.1 \sin c_5 + 2 = 2.0869714$

$c_7 = 0.1 \sin c_6 + 2 = 2.0869713$

$c_8 = 0.1 \sin c_7 + 2 = 2.0869713$

- approximate solutions

In general: The equation $f(x)=0$ can be transformed to the form $x = \varphi(x)$ by a simple addition of x to both sides:

$$x = x+f(x)$$

$$x = \varphi(x)$$

Then choose an initial value c_1 and start iterations:

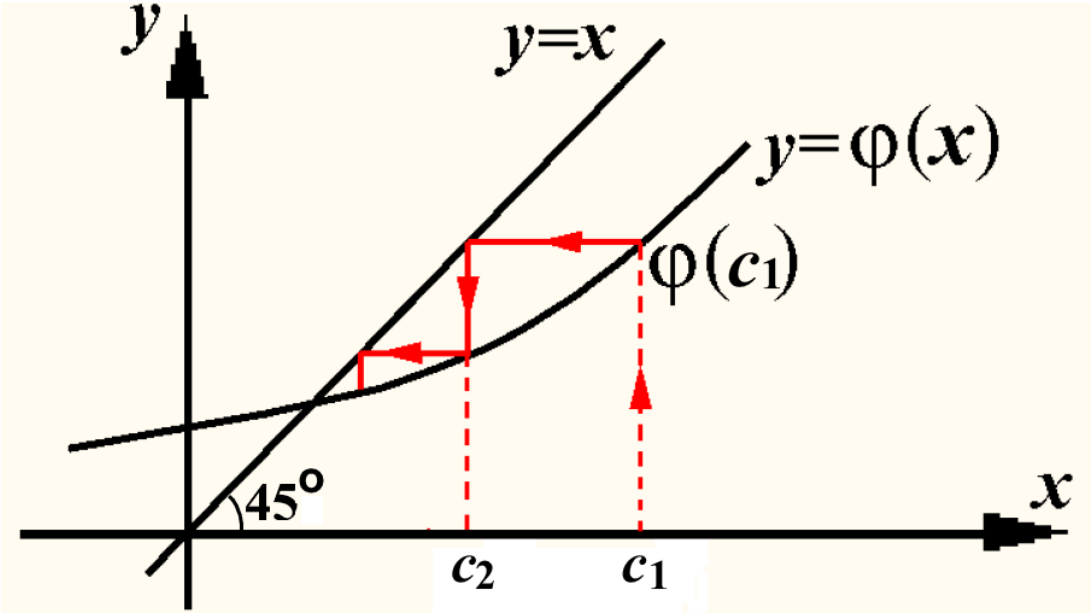
$$c_{k+1} = \varphi(c_k)\,, \qquad k=1, 2,...$$

If c_{k+1} and c_k become very close to each other, then

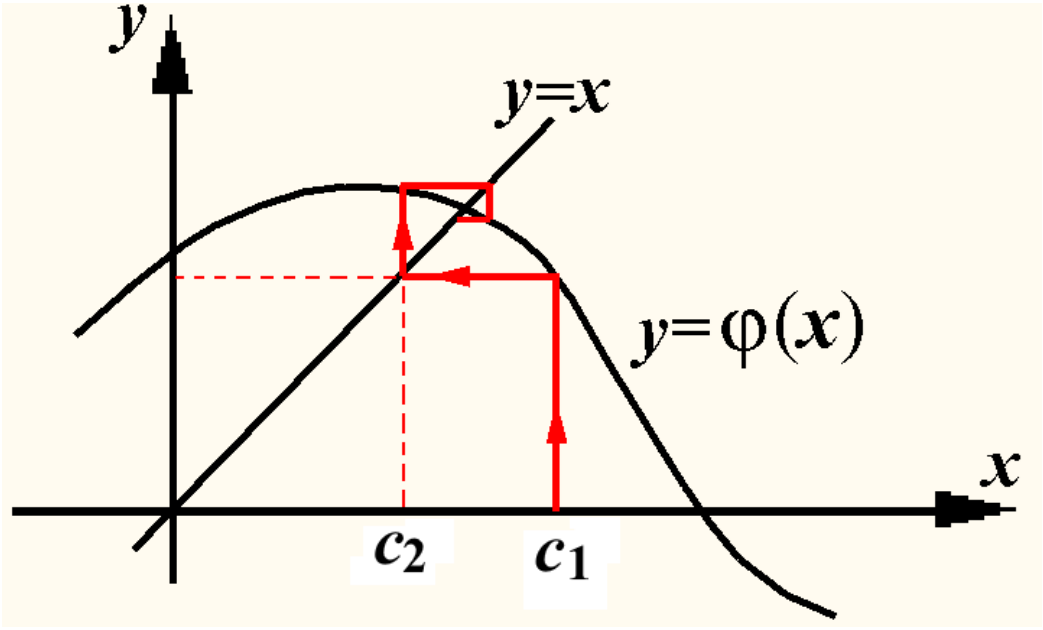
$$c_k \approx x^* \text{ - solution}$$

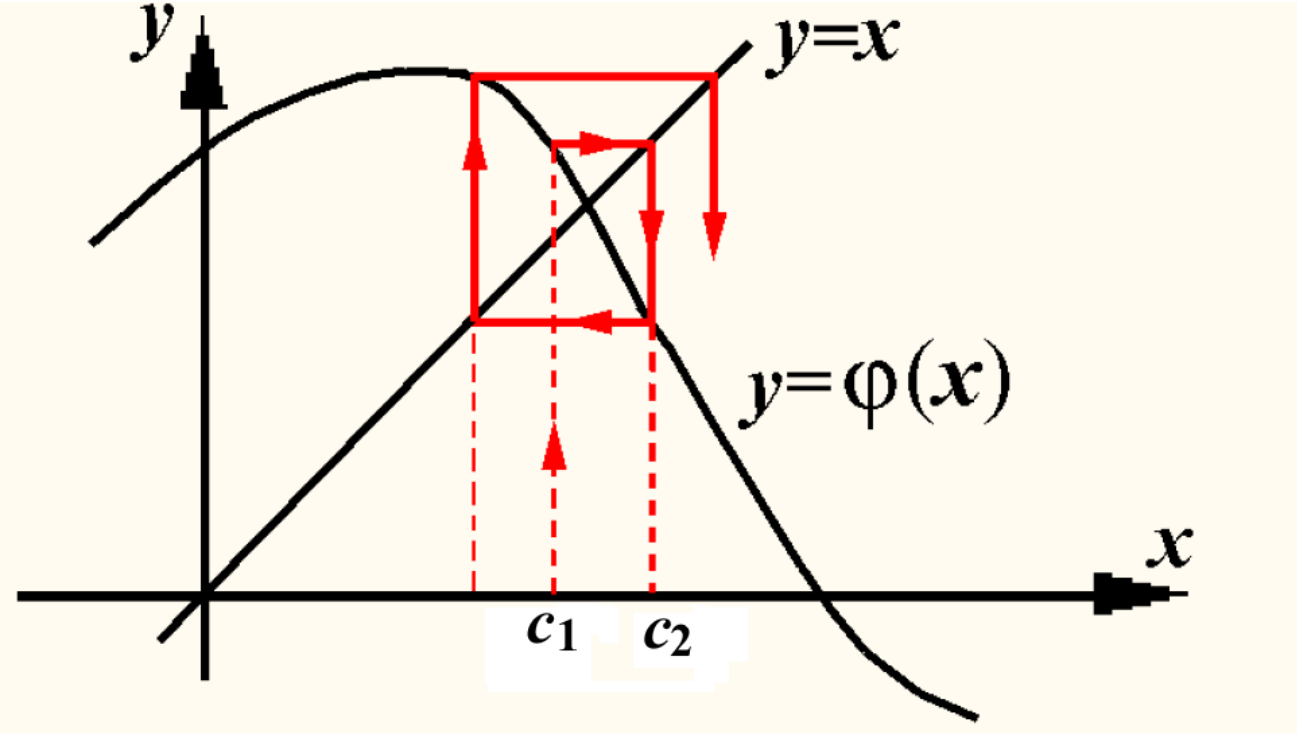
Geometric illustration of solving $x=\varphi(x)$:

Let us plot $y=x$, $y=\varphi(x)$



$$c_2 = \varphi(c_1)$$





In this figure, we see a divergence of successive approximations c_k , which move away from the exact solution x^* .

The convergence or divergence of the sequence c_k

depends on the slope of curve $y=\varphi(x)$ *to the x-axis, that is on the module of the first derivative :*

$$\left|\varphi'(x)\right|$$

Theorem *(sufficient condition for the convergence of iterations):*

If $\left|\varphi'(x)\right| < 1$ at $a \le x \le b$, *then*

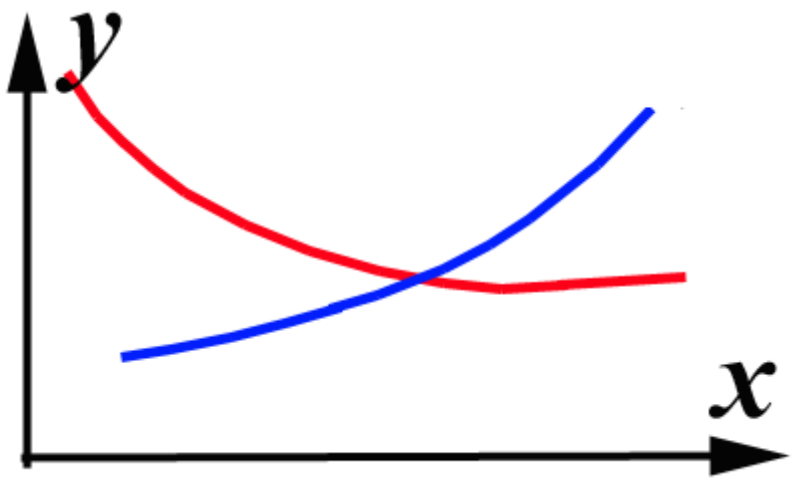
$c_k \to x^*$ *at* $k \to \infty$, *and*

x^* *is the unique root of the equation* $x=\varphi(x)$.

Chapter 2. Systems of nonlinear equations

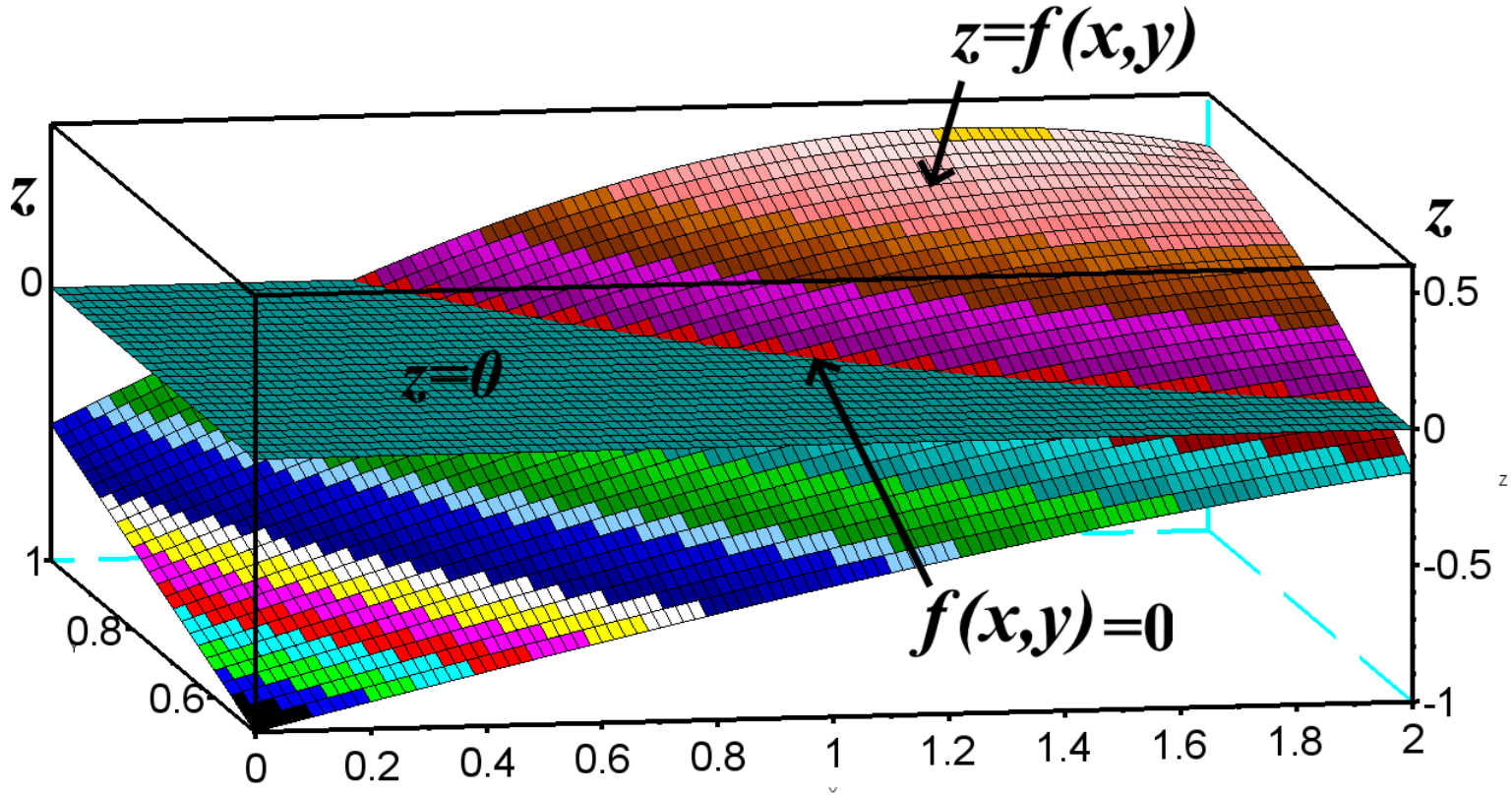
Now we turn to a system of 2 nonlinear equations with 2 unknowns x and y :

$$\begin{cases} f(x, y) = 0 \\ g(x, y) = 0 \end{cases}$$



Geometric interpretation in 2D:
typically each function determines a curve in (x, y) plane.
Intersection of the curves gives a solution.

Interpretation in 3D:



another surface illustrates function $z=g(x,y)$

Method of tangents (Newton's method)

Let us recall the method of tangent lines for a single equation

$$f(x)=0 \qquad a \le x \le b \ .$$

Algorithm for calculation of successive approximations to a solution:

$$c_{k+1}=c_k - f(c_k)/ f'(c_k) \qquad k=1,2,...$$

This was obtained by plotting a line which is tangent to curve $y= f(x)$ at an initial point $x=c_1$, $y= f(c_1)$ on the curve.

Method of tangents (Newton's method)

Let us recall the method of tangent lines for a single equation

$$f(x)=0 \qquad a \le x \le b\ .$$

Algorithm for calculation of successive approximations to a solution:

$$c_{k+1}=c_k - f(c_k)/f'(c_k) \qquad k=1,2,...$$

This was obtained by plotting a line which is tangent to curve $y=f(x)$ at an initial point $x=c_1$, $y=f(c_1)$ on the curve.

Similarly, in the case of 2 equations, we should obtain 2 planes which are tangent to surfaces $z=f(x, y)$, $z=g(x, y)$ at initial points $x_1, y_1, z_{1f}=f(x_1, y_1)$ and $x_1, y_1, z_{1g}=g(x_1,y_1)$.

Then we should:

1) **find intersections of the tangent planes with plane $z=0$; such intersections are 2 straight lines,**

2) **find intersection of the 2 lines in the plane $z=0$. It gives us a point x_2, y_2 which is the next approximation to the root.**

Formula for calculation of x_2, y_2 :

[in case of 1 equation it was $c_2 = c_1 - f(c_1)/f'(c_1)$]

$$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} - \mathrm{inv}(F'_1) \times \begin{pmatrix} f(x_1,y_1) \\ g(x_1,y_1) \end{pmatrix}$$

column vectors — 2 x 2 matrix — vector

$$F'_1 = \begin{pmatrix} \partial f/\partial x & \partial f/\partial y \\ \partial g/\partial x & \partial g/\partial y \end{pmatrix} \Bigg|_{\text{at } x_1, y_1}$$

matrix

inv – inverse of the matrix

$$F' = \begin{pmatrix} \partial f/\partial x & \partial f/\partial y \\ \partial g/\partial x & \partial g/\partial y \end{pmatrix}$$

$$\text{inv } F' = \frac{1}{f_x g_y - g_x f_y} \begin{pmatrix} \partial g/\partial y & -\partial f/\partial y \\ -\partial g/\partial x & \partial f/\partial x \end{pmatrix}$$

see Algebra

$f_x = \partial f/\partial x, \quad f_y = \partial f/\partial y, \ldots.$

Proof of the formula.

If a surface in space *(x,y,z)* is given by expression *G(x,y,z)*=0, then equation of the tangent plane is

$$G_x \cdot (x-x_0) + G_y \cdot (y-y_0) + G_z \cdot (z-z_0) = 0$$

(partial derivatives)

In the case under consideration $z=f(x,y)$ ➔ $f(x,y)-z=0$**:**

$$f_x \cdot (x-x_1) + f_y \cdot (y-y_1) - (z- z_{1f}) = 0$$

$g(x,y)-z=0$**:**

$$g_x \cdot (x-x_1) + g_y \cdot (y-y_1) - (z- z_{1g}) = 0$$

Find intersections with plane $z=0$**:**

$$f_x \cdot (x-x_1) + f_y \cdot (y-y_1) + z_{1f} = 0$$

$$g_x \cdot (x-x_1) + g_y \cdot (y-y_1) + z_{1g} = 0$$

Now find intersection of the lines:

$$\left.\begin{aligned} f_x \cdot (x_2-x_1) + f_y \cdot (y_2-y_1) + z_{1f} = 0 \\ g_x \cdot (x_2-x_1) + g_y \cdot (y_2-y_1) + z_{1g} = 0 \end{aligned}\right\}$$

$$\left.\begin{aligned} f_x\cdot(x_2-x_1) + f_y\cdot(y_2-y_1) + z_{1f}=0 \\ g_x\cdot(x_2-x_1) + g_y\cdot(y_2-y_1) + z_{1g}=0 \end{aligned}\right\}$$

All derivatives are calculated at $x=x_1, y=y_1$

In matrix form:

$$\begin{pmatrix} f_x & f_y \\ g_x & g_y \end{pmatrix} \times \begin{pmatrix} x_2-x_1 \\ y_2-y_1 \end{pmatrix} + \begin{pmatrix} z_{1f} \\ z_{1g} \end{pmatrix} = 0$$

$$\mathbf{F'_1} \times \begin{pmatrix} x_2-x_1 \\ y_2-y_1 \end{pmatrix} + \begin{pmatrix} z_{1f} \\ z_{1g} \end{pmatrix} = 0$$

multiply by $\mathbf{inv(F'_1)}$**:**

$$\begin{pmatrix} x_2 - x_1 \\ \\ y_2 - y_1 \end{pmatrix} + \mathbf{inv(F'_1)} \times \begin{pmatrix} z_{1f} \\ \\ z_{1g} \end{pmatrix} = 0$$

$$\begin{pmatrix} x_2 \\ \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ \\ y_1 \end{pmatrix} - \mathbf{inv(F'_1)} \times \begin{pmatrix} f(x_1,y_1) \\ \\ g(x_1,y_1) \end{pmatrix}$$

General formula:

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} - \mathrm{inv}(F'_k) \times \begin{pmatrix} f(x_k,y_k) \\ g(x_k,y_k) \end{pmatrix}$$

column vectors — matrix — column vector

k=1, 2, 3, . . .

Chapter 3
Systems of linear algebraic equations

$$a x + b y + c z = e$$
$$f x + g y + h z = l$$
$$p x + q y + s z = t$$

x, y, z *are unknowns*

Change the notations:

$$a x_1 + b x_2 + c x_3 = b_1$$
$$f x_1 + g x_2 + h x_3 = b_2$$
$$p x_1 + q x_2 + s x_3 = b_3$$

x_1, x_2, x_3 *are unknowns*

Methods for computation of the solution:

1. Using the inverse $\mathrm{inv}(A)$ of matrix A
2. Gaussian elimination of unknowns
3. LU factorization
4. Iteration method
5.
6.

$$a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 + \ldots + a_{1n}\,x_n = b_1\,,$$

$$a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 + \ldots + a_{2n}\,x_n = b_2\,,$$

- -

$$a_{i1}\,x_1 + a_{i2}\,x_2 + a_{i3}\,x_3 + \ldots + a_{in}\,x_n = b_i\,,$$

- -

$$a_{n1}\,x_1 + a_{n2}\,x_2 + a_{n3}\,x_3 + \ldots + a_{nn}\,x_n = b_n\,.$$

where a_{ij} - given real numbers

x_i - unknowns to be found

The system can be written in matrix form:

$$Ax=b$$

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ - & - & - & - & - \\ a_{i1} & a_{i2} & a_{i3} & \dots & a_{in} \\ - & - & - & - & - \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}$$

$$x=\begin{pmatrix} x_1 \\ x_2 \\ - \\ x_i \\ - \\ x_n \end{pmatrix}$$ column vector

$$b=\begin{pmatrix} b_1 \\ b_2 \\ - \\ b_i \\ - \\ b_n \end{pmatrix}$$ column vector

$Ax=b$

If $\det A \neq 0$,

$$\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix} \neq 0$$

then there exists a unique solution x **of the system.**

1) Method of the inverse of matrix A

$$Ax=b$$

we denote by $\mathrm{inv}(A)$ or A^{-1} the inverse of matrix A

How to compose A^{-1} : by computing its cofactors, see Algebra

If we got A^{-1}, then multiplying the system by A^{-1}

$$A^{-1}Ax = A^{-1}b$$

$$A^{-1}A = I = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 1 \end{pmatrix}$$

identity matrix

$Ix = A^{-1}b$ **and we obtain the solution:** $x = A^{-1}b$

2. Gaussian elimination

Carl Friedrich Gauss 1777-1855

$$a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 + \ldots + a_{1n}\,x_n = b_1\,,$$
$$a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 + \ldots + a_{2n}\,x_n = b_2\,,$$
$$- -$$
$$a_{i1}\,x_1 + a_{i2}\,x_2 + a_{i3}\,x_3 + \ldots + a_{in}\,x_n = b_i\,,$$
$$- -$$
$$a_{n1}\,x_1 + a_{n2}\,x_2 + a_{n3}\,x_3 + \ldots + a_{nn}\,x_n = b_n\,.$$

Suppose that $a_{11} \neq 0$. Then

$$x_1 = (b_1 - a_{12} x_2 - a_{13} x_3 - \ldots - a_{1n} x_n)/ a_{11}$$

we multiply this by a_{i1}:

$$a_{i1} x_1 = a_{i1}(b_1 - a_{12} x_2 - a_{13} x_3 - \ldots - a_{1n} x_n)/a_{11} \qquad (*)$$

and replace $a_{i1} x_1$ in i^{th} equation by (*):

$$a_{i1} x_1 + a_{i2} x_2 + \ldots + a_{ij} x_j + \ldots + a_{in} x_n = b_i \qquad i^{th} \text{ equation}$$

$$a_{i1}(b_1 - a_{12} x_2 - a_{13} x_3 - \ldots - a_{1n} x_n)/a_{11}$$
$$+ a_{i2} x_2 + \ldots + a_{ij} x_j + \ldots + a_{in} x_n = b_i$$

We now sum up the coefficients in front of the same x_j

$$a_{i2}^{(1)} x_2 + \ldots + a_{ij}^{(1)} x_j + \ldots + a_{in}^{(1)} x_n = b_i^{(1)}$$

where $a_{i2}^{(1)} = a_{i2} - a_{i1}a_{12}/a_{11}$

$a_{i3}^{(1)} = a_{i3} - a_{i1}a_{13}/a_{11}$

$a_{ij}^{(1)} = a_{ij} - a_{i1}a_{1j}/a_{11}$

$b_i^{(1)} = b_i - a_{i1}b_1/a_{11}$ $\qquad i=2,3,\ldots, n$

After elimination of x_1 , the system becomes

$$a_{11}\,x_1 + a_{12}\,x_2 + \ldots + a_{1j}\,x_j + \ldots + a_{1n}\,x_n = b_1$$

$$a_{22}^{(1)}\,x_2 + \ldots + a_{2j}^{(1)}\,x_j + \ldots + a_{2n}^{(1)}\,x_n = b_2^{(1)}$$

- -

$$a_{i2}^{(1)}\,x_2 + \ldots + a_{ij}^{(1)}\,x_j + \ldots + a_{in}^{(1)}\,x_n = b_i^{(1)}$$

- -

$$a_{n2}^{(1)}\,x_2 + \ldots + a_{nj}^{(1)}\,x_j + \ldots + a_{nn}^{(1)}\,x_n = b_n^{(1)}$$

Suppose that $a_{22}^{(1)} \neq 0$

Keeping on the elimination of unknowns, in the same way we obtain

$$a_{11}\,x_1 + a_{12}\,x_2 + \ldots + a_{1j}\,x_j + \ldots + a_{1n}\,x_n = b_1$$

$$a_{22}^{(1)}\,x_2 + \ldots + a_{2j}^{(1)}\,x_j + \ldots + a_{2n}^{(1)}\,x_n = b_2^{(1)}$$

$$- -$$

$$a_{n-1,\,n-1}^{(n-2)}\,x_{n-1} + a_{n-1,n}^{(n-2)}\,x_n = b_{n-1}^{(n-2)}$$

$$a_{nn}^{(n-1)}\,x_n = b_n^{(n-1)}$$

If $a_{nn}^{(n-1)} \neq 0$, then $x_n = b_n^{(n-1)} / a_{nn}^{(n-1)}$

$x_{n-1} =$

- It could happen that $a_{ii}^{(i-1)}=0$.

Example. $A=\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

$$0\cdot x_1 + x_2 = b_1$$

$$x_1 + 0\cdot x_2 = b_2$$

Theorem The system of n linear algebraic equations with $\det A \neq 0$ can be transformed to an equivalent system with nonzero $a_{ii}^{(i-1)}$ by transposition (interchange) of columns or rows.

The number of arithmetic operations necessary for obtaining a solution with Gaussian elimination is

$$2n(n+1)(n+2)/3 + n(n-1)$$

(a proof is available in textbooks).

3) LU factorization method

Let *Ax=b* denote the linear system to be solved, where *A* is *n*x*n* size matrix. In Gaussian elimination, the system was reduced to the upper triangular system *Ux=g* with

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ . & & \cdots & . \\ . & & \cdots & . \\ . & & \cdots & . \\ 0 & \cdots & 0 & u_{nn} \end{bmatrix}$$

Let us introduce an auxiliary lower triangular matrix *L* :

$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ m_{21} & 1 & \cdots & 0 \\ . & & \cdots & . \\ . & & \cdots & . \\ . & & \cdots & . \\ m_{n1} & \cdots & m_{nn-1} & 1 \end{bmatrix}$$

The relationship of the matrices L and U to the original A is given by the following theorem:

Theorem. Let A be a matrix with $det\ A \neq 0$. Then if U is produced as Gaussian elimination without interchange of rows/columns, then there exists triangular matrix L such that $LU=A$ and this is called factorization of A.

The factorization leads to a slightly different way of solving the system $Ax=b$. It can be rewritten as $LUx=b$. We denote $Ux=g$ and obtain the two simple systems

$$Lg=b \quad \text{and} \quad Ux=g \ .$$

Both L and U are triangular, therefore solutions can be easily calculated by substitution. The computational cost is here reduced drastically as compared to Gaussian one.

Instead of constructing L and U by using the elimination steps, it is possible to solve directly elements of these matrices.
We will illustrate the direct computation of L and U in the case n=3:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

1st row of A: $1=1\cdot u_{11}$ $1=1\cdot u_{12}$ $-1=1\cdot u_{13}$

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$1= m_{21}\cdot 1$

$-2= m_{31}\cdot 1$

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -2 & m_{32} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

$-2 = -1 + u_{23}$

$2 = 1 + u_{22}$

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -2 & m_{32} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & u_{33} \end{bmatrix}$$

$1 = 2 - m_{32} + u_{33}$

$1 = -2 + m_{32}$

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & -2 \\ -2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -2 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 2 \end{bmatrix}$$

L U

Take, for example, $b=\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$

$$LUx=b$$

$$Lg = b$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -2 & 3 & 1 \end{pmatrix} \begin{pmatrix} g_1 \\ g_2 \\ g_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad \begin{array}{l} \Rightarrow \quad g_1=1 \\ \Rightarrow \quad g_2=0 \\ \Rightarrow \quad g_3=3 \end{array}$$

Finally, we solve $Ux=g$:

$$\begin{pmatrix} 1 & 1 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix}$$

$$\Rightarrow \quad x_1 + x_2 - x_3 = 1 \quad \Rightarrow x_1 = 1$$

$$\Rightarrow \quad x_2 = 3/2$$

$$\Rightarrow \quad x_3 = 3/2$$

x=linsolve(A, d) *solves the system* **Ax+d=0** **with LU factorization**

x=linsolve(A,-b) *solves the system* **Ax=b**

Chapter 3-2. Systems of linear algebraic equations

4) Iteration method $Ax=b$ $\det A \neq 0$

$$A=\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ - & - & - & - & - \\ a_{i1} & a_{i2} & a_{i3} & \dots & a_{in} \\ - & - & - & - & - \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix}$$

$$x=\begin{pmatrix} x_1 \\ x_2 \\ - \\ x_i \\ - \\ x_n \end{pmatrix} \qquad b=\begin{pmatrix} b_1 \\ b_2 \\ - \\ b_i \\ - \\ b_n \end{pmatrix}$$

Iteration method defines a sequence of approximate solutions $x^{(k)}$ that converge to the exact solution $x^{(k)} \to x^*$ as $k \to \infty$

Let us transform $Ax=b$ to the form $x=Cx+d$, then choose some $x^{(1)}$ and organize iterations:

$$x^{(k+1)}=Cx^{(k)}+d\,, \qquad k=1,2,\ldots$$

Example: $1.01\,x_1 + 0.2\,x_2 = 3 \;\Rightarrow\; (1+0.01)\,x_1 + 0.2\,x_2 = 3$

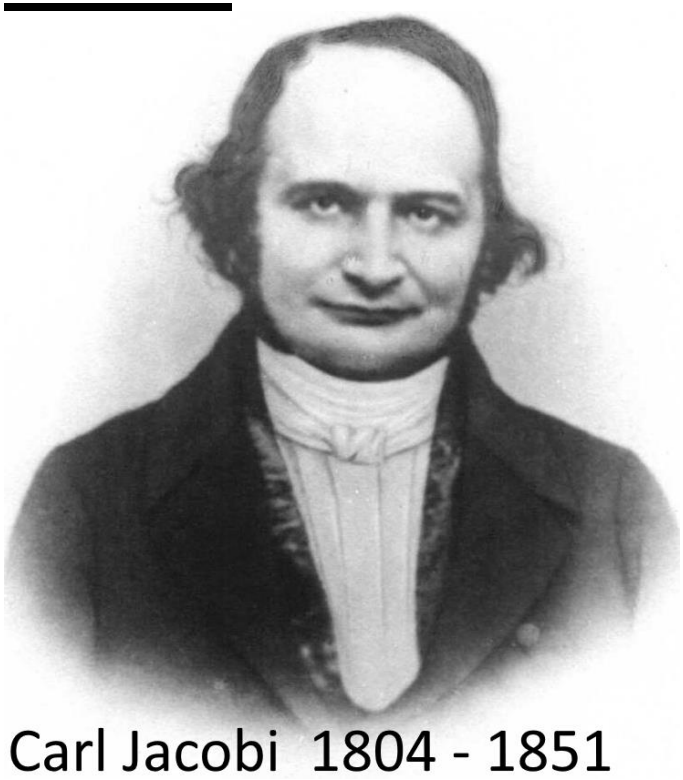
$0.05\,x_1 + 1.08\,x_2 = 2 \;\Rightarrow\; 0.05\,x_1 + (1+0.08)\,x_2 = 2$

$$x_1 = 3 - 0.01\,x_1 - 0.2\,x_2$$
$$x_2 = 2 - 0.05\,x_1 - 0.08\,x_2$$

choose $x_1^{(1)}=3\,,\; x_2^{(1)}=2$

$$x_1^{(k+1)} = 3 - 0.01\,x_1^{(k)} - 0.2\,x_2^{(k)}$$
$$x_2^{(k+1)} = 2 - 0.05\,x_1^{(k)} - 0.08\,x_2^{(k)}$$

A way of transforming $Ax=b$ to the form $x=Cx+d$ is <u>Jacobi iterative method</u>



Carl Jacobi 1804 - 1851

We illustrate it for the case n=3 :

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

Suppose the diagonal elements a_{ii} are non-zero and divide the first equation by a_{11}, the second by a_{22} and third by a_{33}:

$$x_1 = \frac{1}{a_{11}}(b_1 - a_{12}x_2 - a_{13}x_3)$$

$$x_2 = \frac{1}{a_{22}}(b_2 - a_{21}x_1 - a_{23}x_3)$$

$$x_3 = \frac{1}{a_{33}}(b_3 - a_{31}x_1 - a_{32}x_2)$$

We choose

$$x^{(1)} = \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ x_3^{(1)} \end{pmatrix}$$

and define iterations:

$$x_1^{(k+1)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)})$$

$$x_2^{(k+1)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)})$$

$$x_3^{(k+1)} = \frac{1}{a_{33}}(b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)})$$

$$x^{(k+1)} = Cx^{(k)} + \beta \ , \qquad \beta_i = b_i / a_{ii}$$

diagonal elements of matrix C are zeros in the Jakobi method

$$x^{(k+1)} = Cx^{(k)} + \beta$$

$$C = \begin{pmatrix} c_{11} & c_{12} & c_{13} & \dots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \dots & c_{2n} \\ - & - & - & - & - \\ c_{i1} & c_{i2} & c_{i3} & \dots & c_{in} \\ - & - & - & - & - \\ c_{n1} & c_{n2} & c_{n3} & \dots & c_{nn} \end{pmatrix}$$

Theorem. A sufficient condition for convergence of $x^{(k)}$ to exact solution $x^{(k)} \to x^*$ is $\|C\| < 1$, where $\|C\|$ is a norm of matrix C.

(Proof is omitted).

The convergence means $x_i^{(k)} \to x_i^*$ for any i

For the properties of a norm, see textbook by S.Sastry.

There are 3+ types of matrix norms:

1) $$\|C\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |c_{ij}|$$ **"column" norm**

2) $$\|C\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |c_{ij}|$$ **"row" norm**

$$x_1 = 0.01\, x_1 - 0.2\, x_2 + 3$$
$$x_2 = -0.05\, x_1 + 0.08\, x_2 + 2$$

3) **Euclidean norm:**

$$\|C\|_e = \left(\sum_{i,j=1}^{n} |c_{ij}|^2 \right)^{1/2}$$

Column vector norms:

For the vector

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

some useful norms are

$$\| x \|_1 = |x_1| + |x_2| + \cdots + |x_n| = \sum_{i=1}^{m} |x_i|$$

$$\| x \|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2} = \left[\sum_{i=1}^{n} |x_i|^2 \right]^{1/2} = \| x \|_e$$

$$\| x \|_\infty = \max_i |x_i| .$$

The norm $\|\bullet\|_2$ is called the *Euclidean* norm since it is just the formula for distance in the [ ] Euclidean space.

$$A=\begin{pmatrix} 0 & 1 & 0 \\ 1.1 & -0.5 & 0 \\ 0 & 1 & 1.2 \end{pmatrix}$$

Scilab :

norm(A,1)

norm(A,'inf') **(Matlab : norm(A,'infty')**

Note: condition $\|C\| < 1$ is analogous to the condition $|\varphi'(x)| < 1$ in the section of Chapter 1 addressing the nonlinear equation $f(x)=0$ $x=\varphi(x)$

Theorem (on the accuracy of successive approximations $x^{(k)}$) : $x^{(k+1)}=Cx^{(k)}+\beta$

1)

$$\|x^{(k)} - x^*\| \le \frac{\|C\|^k}{1-\|C\|} \|\beta\|$$

2)

$$\|x^{(k)} - x^*\| \le \frac{\|C\|}{1-\|C\|} \|x^{(k+1)}-x^{(k)}\|$$

P. Seidel 1821-1896

A modified version of the Jacobi iteration method is Seidel method:

Seidel suggested to immediately insert the calculated $x_i^{(k+1)}$ ***into the right-hand sides of next equations. This accelerates the convergence*** $x^{(k)} \rightarrow x^*$

$$x_1^{(k+1)} = \frac{1}{a_{11}}(b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)})$$

$$x_2^{(k+1)} = \frac{1}{a_{22}}(b_2 - a_{21}x_1^{(k+1)} - a_{23}x_3^{(k)})$$

$$x_3^{(k+1)} = \frac{1}{a_{33}}(b_3 - a_{31}x_1^{(k+1)} - a_{32}x_2^{(k+1)})$$

The number of arithmetic operations per iteration is $\approx n^2$

Therefore, the effectiveness of iterative methods essentially depends on the required number of iterations k

Chapter 4. Trapezoids method for calculation of definite integrals

$$\int_a^b f(x)\,dx$$

When do you need to use a numerical method?

1st case: Function $f(x)$ is given by a formula; however, the integral cannot be expressed in terms of elementary functions $sin(x)$, $cos(x)$, $tan(x)$, $exp(x)$,...

For example,

$$\int_a^b e^{x \sin(\cos(\sin x))}\,dx$$

2nd case: values of function $f(x)$ are only given at finite number of points of the segment $[a,b]$:

$$y_0, y_1, y_2, \ldots, y_n$$

$$at \qquad a=x_0, x_1, x_2, \ldots, x_n=b$$

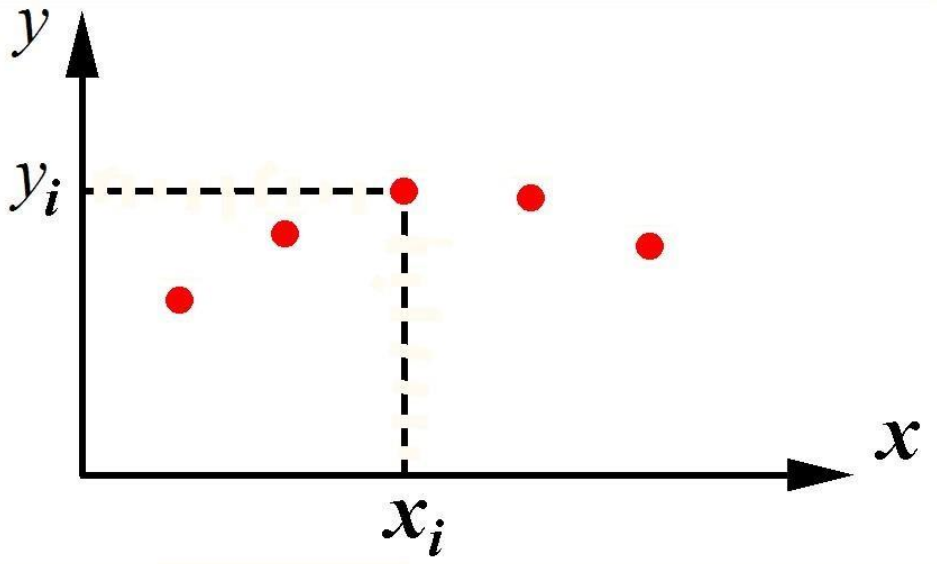
(for example, y_i are results of some experiments).

$$I=\int_{x_0}^{x_n} f(x)\,dx = \ ?$$

In case 1, when $f(x)$ is given by a formula, we can choose points x_i ourselves and calculate $y_i=f(x_i)$

Assume that $h_i = x_{i+1} - x_i = \text{const} = h$

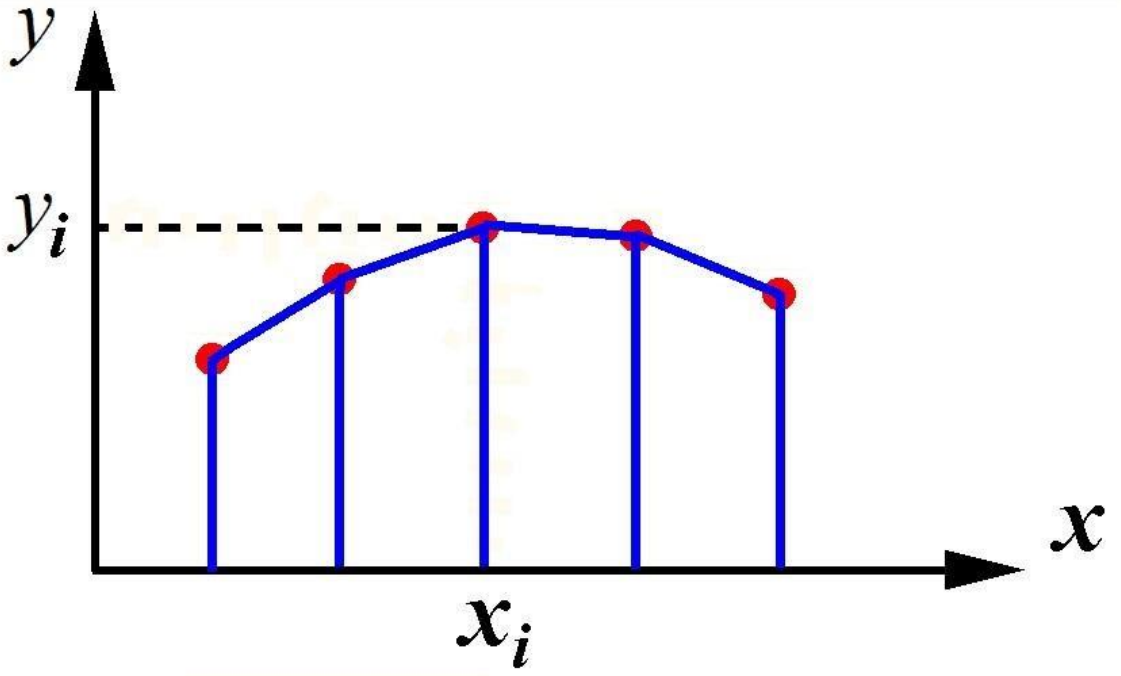
$$I = \int_{x_0}^{x_n} f(x)\,dx = \ ?$$



Trapezoids formula:

$$I = \int_{x_0}^{x_n} f(x)\,dx \approx h\left(y_0/2 + y_1 + y_2 + \ldots + y_{n-1} + y_n/2 \right)$$

Derivation of the formula:



Suppose that $f(x)$ is well approximated on $[x_i, x_{i+1}]$ by the linear function

$$f(x) \approx y_i + (y_{i+1} - y_i)(x - x_i)/h$$

$$\int_{x_i}^{x_{i+1}} [\, y_i + (y_{i+1} - y_i)(x - x_i)/h \,]\, dx =$$

$$= y_i h + (y_{i+1} - y_i) \int (x - x_i)\, dx / h =$$

$$= y_i h + (y_{i+1} - y_i) [(x_{i+1} - x_i)^2 - (x_i - x_i)^2]/2h =$$

$$= y_i h + (y_{i+1} - y_i)\, h / 2 =$$

$$= (y_i + y_{i+1})\, h/2$$

Actually, this expression is evident, as an integral is known to be the area below the plot of function *f(x)* ; *in the case of linear function the area is halved sum of bases* × *height of trapezoid.*

$$\int_{x_0}^{x_1} f(x)\,dx \approx h\,(y_0 + y_1)/2$$

$$\int_{x_0}^{x_2} f(x)\,dx \approx h\,(y_0+y_1+y_1+y_2)/2 = h\,(y_0+2y_1+y_2)/2$$

$$\int_{x_0}^{x_3} f(x)\,dx \approx h\,(y_0+2y_1+2y_2+y_3)/2$$

Summation over all segments $[\,x_i\,,\,x_{i+1}]$, $i = 0, 1, 2, \ldots$ **gives**

$$\int_{x_0}^{x_n} f(x)\,dx \approx h\left(y_0/2 + y_1 + y_2 + \ldots + y_{n-1} + y_n/2 \right)$$

***Theorem* on the error of the trapezoid formula:**

$$\int_{x_0}^{x_n} f(x)\,dx = h\left(y_0/2 + y_1 + y_2 + \ldots + y_{n-1} + y_n/2\right) - f''(c)\,h^2\,(x_n - x_0)/12$$

↑

Error

where c is some point in the interval $x_0 < x < x_n$

(Proof is omitted).

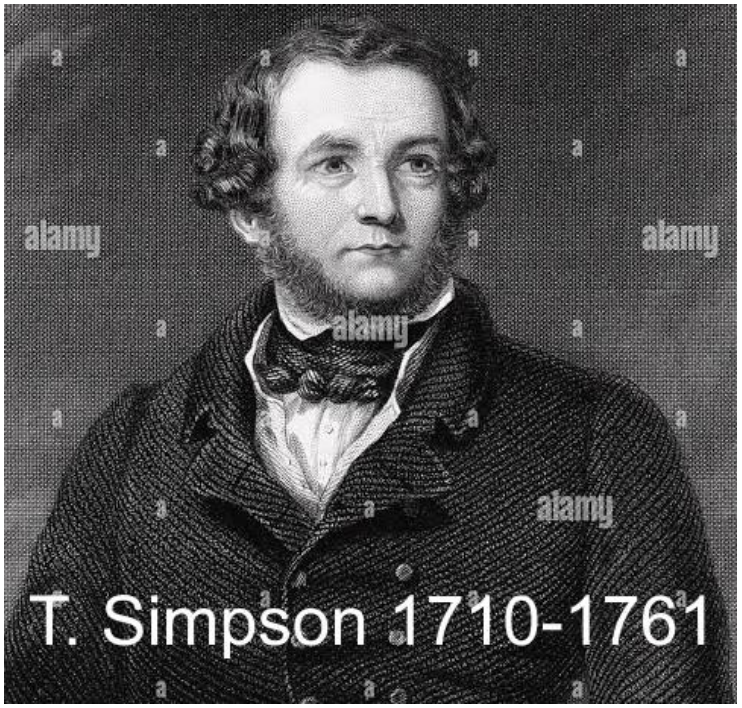
Another example

$$\int_{5}^{13} \sqrt{2x-1}\ dx$$

```
clear
x=5: 0.5 : 13
y=sqrt(2*x-1)
Int=inttrap(x,y)
32.663890
```

```
x=5: 0.1 : 13
y=sqrt(2*x-1)
Int=inttrap(x,y)
32.666556
```

Chapter 4. Simpson's method for calculation of definite integrals



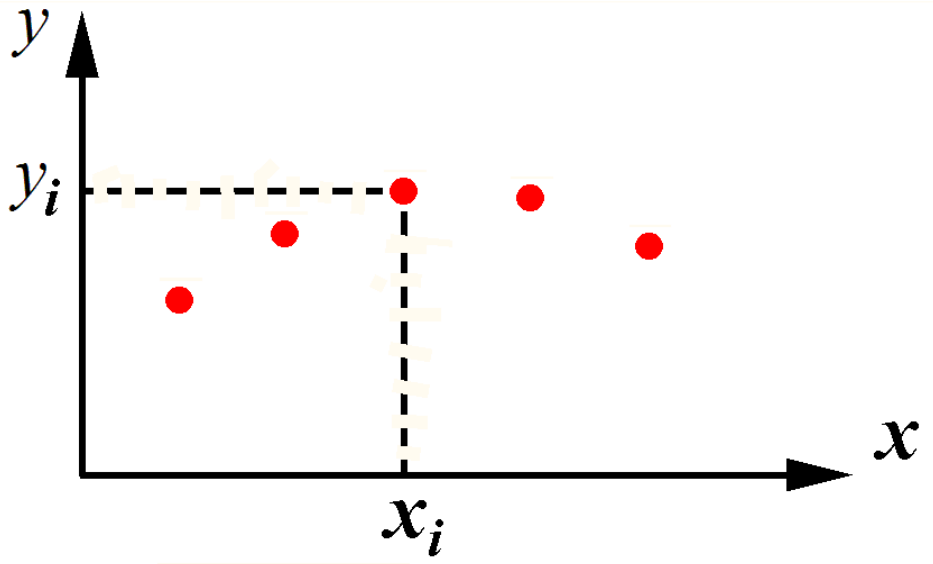
$$\int_a^b f(x)dx = ?$$

Again, we assume that values of $f(x)$ are given at a finite number of points of the segment [a,b]:

$x_0, x_1, x_2, \ldots, x_n$ $\qquad$ $y_0, y_1, y_2, \ldots, y_n$

In addition, we assume for simplicity that

$$x_{i+1} - x_i = \text{const} = h$$



$$\int_{x_0}^{x_n} f(x)\,dx = \ ?$$

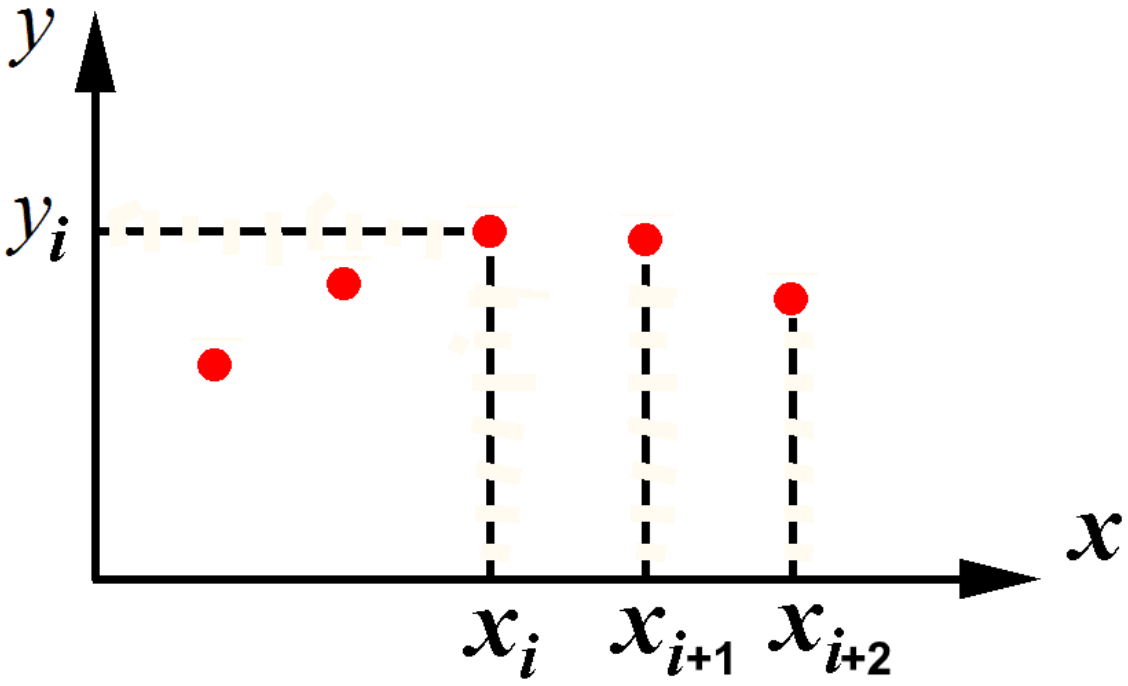
Thomas Simpson's formula:

$$\int_{x_0}^{x_n} f(x)\,dx \approx h[\,y_0 + y_n + 2(y_2 + y_4 + \ldots + y_{n-2}) + 4(y_1 + y_3 + \ldots + y_{n-1})\,] / 3$$

It provides a higher accuracy (smaller error) of integral calculation than trapezoid formula.

The way of obtaining Simpson's formula:

we approximate $f(x)$ by pieces of <u>parabolas</u> which pass through three neighboring points (x_i, y_i) (x_{i+1}, y_{i+1}) (x_{i+2}, y_{i+2}) (not by straight segments as in trapezoids method).



$$f(x) \approx y_i (x - x_{i+1})(x - x_{i+2}) / (2h^2) -$$
$$- y_{i+1} (x - x_i)(x - x_{i+2}) / h^2 +$$
$$+ y_{i+2} (x - x_i)(x - x_{i+1}) / (2h^2)$$

$$f(x) \approx y_i (x - x_{i+1})(x - x_{i+2})/(2h^2) -$$
$$- y_{i+1} (x - x_i)(x - x_{i+2})/h^2 +$$
$$+ y_{i+2} (x - x_i)(x - x_{i+1})/(2h^2)$$

Suppose $x_0 = 0$, then on segment [0, 2h]:

$$f(x) \approx y_0 (x - h)(x - 2h)/(2h^2) -$$
$$- y_1 (x - 0)(x - 2h)/h^2 +$$
$$+ y_2 (x - 0)(x - h)/(2h^2)$$

Integral of parabola can be expressed in analytical form

$$\int_{x_0}^{x_2} f(x)\,dx \approx h(y_0 + 4y_1 + y_2)/3$$

$$\int_{x_0}^{x_2} f(x)\,dx \approx h(y_0 + 4y_1 + y_2)/3$$

$$\int_{x_0}^{x_4} f(x)\,dx = \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx \approx$$

$$\approx h(y_0 + 4y_1 + y_2)/3 + h(y_2 + 4y_3 + y_4)/3 =$$

$$= h(y_0 + 4y_1 + 2y_2 + 4y_3 + y_4)/3$$

$$\int_{x_0}^{x_6} f(x)\,dx \approx h(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + 4y_5 + y_6)/3$$

and so on. Eventually:

$$\int_{x_0}^{x_n} f(x)\,dx \approx h[\, y_0 + y_n + 2(y_2 + y_4 + \ldots + y_{n-2}) + 4(y_1 + y_3 + \ldots + y_{n-1})\,] / 3$$

Next expression shows an error of Simpson's formula

$$\int_{x_0}^{x_n} f(x)\,dx = h[\, y_0 + y_n + 2(y_2 + y_4 + \ldots + y_{n-2}) + 4(y_1 + y_3 + \ldots + y_{n-1})\,] / 3 - f^{(4)}(c)\, h^4 (x_n - x_0) / 180$$

where $x_0 < c < x_n$

Practical Runge's rule for estimation of the error:

Suppose that an integral

$$\int_a^b f(x)dx$$

is calculated using the splitting of [a, b] into n **subsegments, and denote the result by** I_n **.**
Then calculate the same integral using the splitting into $2n$ **subsegments, denote the result by** I_{2n}**.**

Runge's rule: the estimate of the error is

$$|I^* - I_{2n}| \le q\,|I_{2n} - I_n|$$

where I^* **is the exact value of integral,**

q= 1/3 in case of trapezoids method,

q= 1/15 in case of Simpson's method.

A numerical example is given here.

Example 6.20 Evaluate

$$I = \int_0^1 \int_0^1 e^{x+y}\, dx\, dy,$$

using the trapezoidal and Simpson's rules. With $h = k = 0.5$, we have the following table of values of e^{x+y}.

	x		
y	0	0.5	1.0
0	1	1.6487	2.7183
0.5	1.6487	2.7183	4.4817
1.0	2.7183	4.4817	7.3891

Using the 'trapezoidal rule' from Eq. (6.94) repeatedly, we obtain

$$I = \frac{0.25}{4}[1.0 + 4(1.6487) + 6(2.7183) + 4(4.4817) + 7.3891]$$

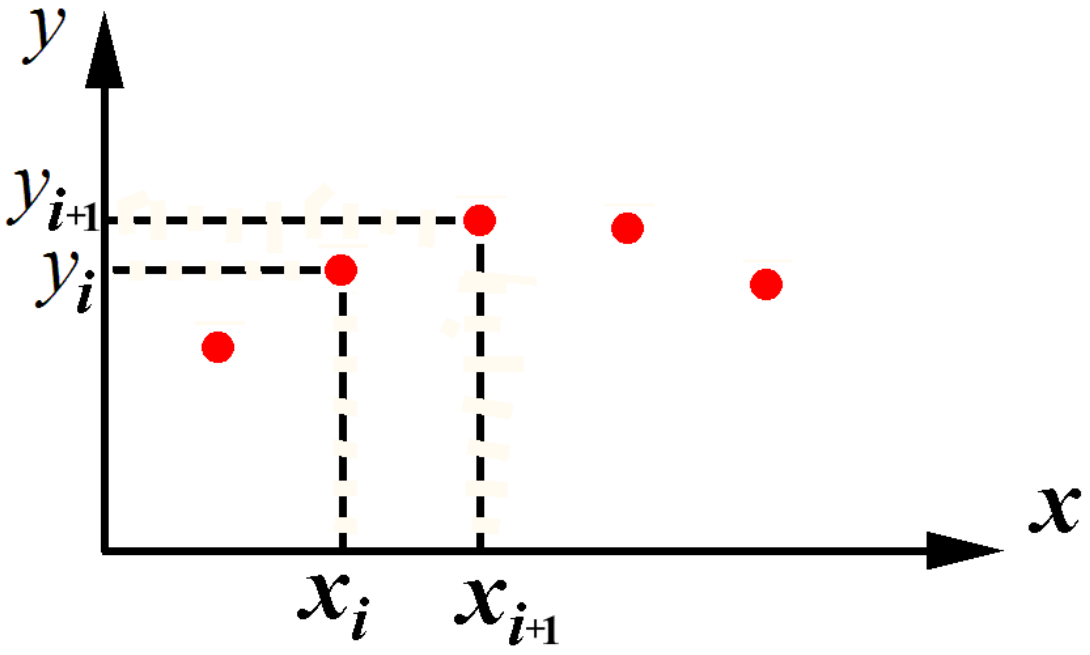
Chapter 5. Interpolation by polynomials

Suppose that values of a function $f(x)$ are given at finite number of points/nodes $x_0, x_1, x_2, \ldots, x_n$ (the function is given by table/array): $y_0, y_1, y_2, \ldots, y_n$.

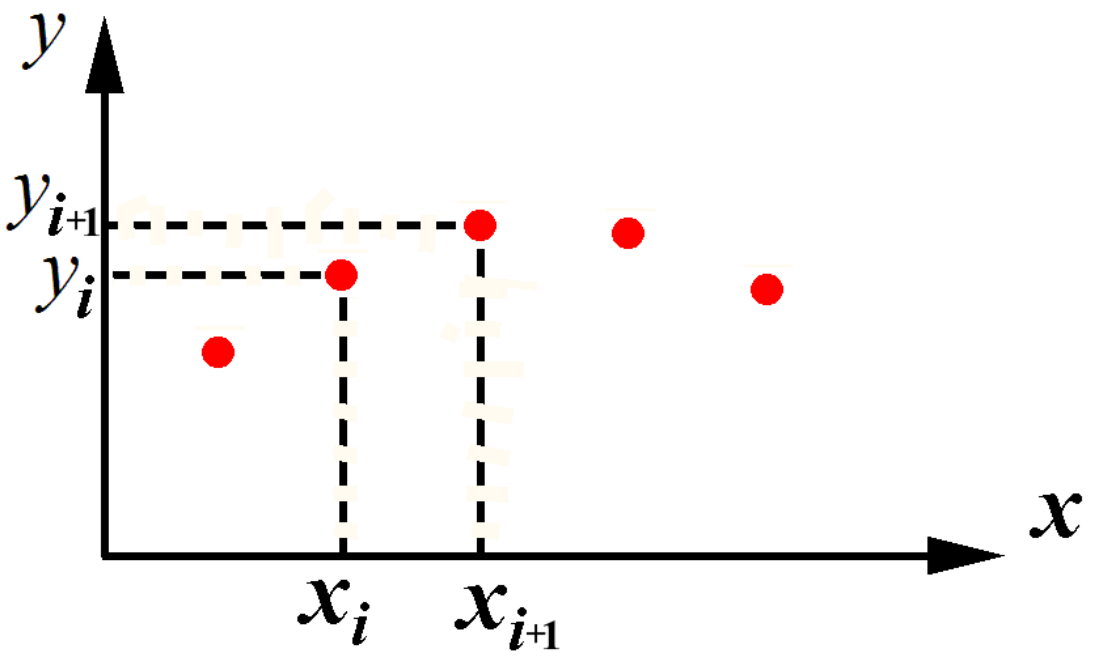
What value can be adopted as approximate value of f at a point x lying between nodes?

If $x_i < x < x_{i+1}$, then $f(x) = ?$ approximately ?

Term "interpolation" means approximation between nodes



An evident approach is to plot a polynomial $P(x)$ that passes through the given points (x_i , y_i), and then to assume that $f(x) \approx P(x)$.



(We already used a linear function $P_1(x)$ to obtain trapezoids formula, as well as a parabola $P_2(x)$ to obtain Simpson's formula).

Is it possible to plot a single polynomial that passes through all $n+1$ points (x_i , y_i) in the plane (x, y) ?

A general form of n^{th} degree polynomial is:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

It involves $n+1$ coefficient a_i .

We can impose the condition $P_n(x_i) = y_i$, $i=0,1, \ldots n$

$$a_0 + a_1 x_i + a_2 x_i^2 + \ldots + a_n x_i^n = y_i$$

and obtain the system of $n+1$ algebraic equations with respect to a_i . The system can be solved with methods described in Chapter 2.

Unfortunately, this way of getting $P_n(x)$ needs a great deal of computations.

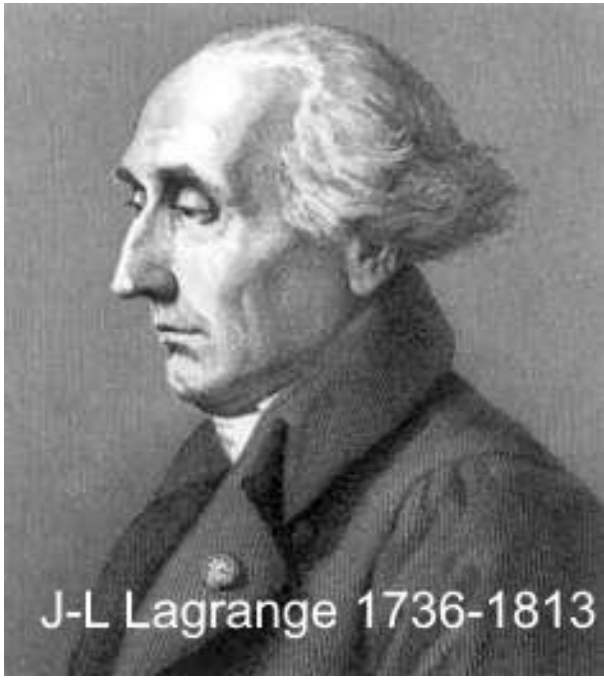
Lagrange suggested a form of the required polynomial which does not need solving any algebraic equations.

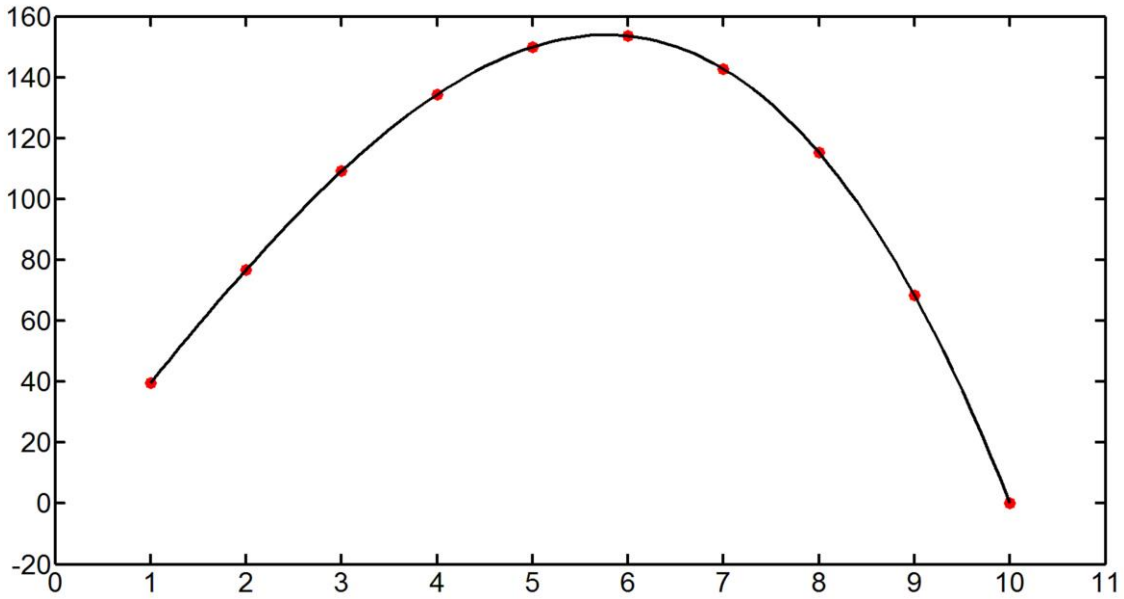
Lagrange's polynomial:

$$P(x) = y_0 \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)} +$$

$$+ y_1 \frac{(x-x_0)(x-x_2)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_n)} +$$

$$\ldots + y_n \frac{(x-x_0)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_0)(x_n-x_2)\ldots(x_n-x_{n-1})}$$


J-L Lagrange 1736-1813



Regarding the error of approximation $f(x) \approx P(x)$ **at** $x_i < x < x_{i+1}$ **:**

Theorem

If there exist derivatives $f^{n+1}(x)$ of function $f(x)$ up to the order $n+1$ on segment $[x_0, x_n]$, then

$$f(x) - P(x) = f^{n+1}(c) \cdot (x-x_0)(x-x_1) \ldots (x-x_n)/(n+1)!$$

where $x_0 \le c \le x_n$, ! is the factorial

(proof is omitted)

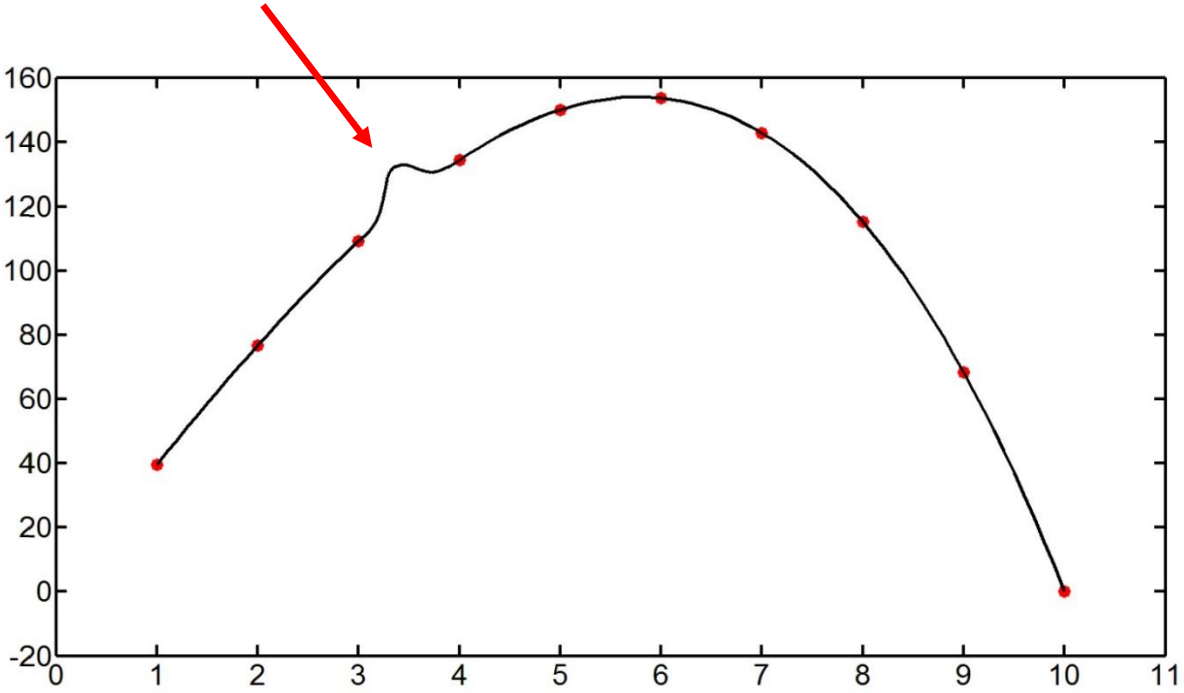
Example

Estimate the error of approximation of the function $y=\ln x$ by Lagrange's polynomial of degree 2. Use the points:

x_i	$y_i=\ln x$
2.0	0.69315
2.5	0.91629
3.0	1.09861

```
x0= 1
x1= 2
x2= 3
x3= 5
      y0= 2
      y1= 2.9
      y2= 4.2
      y3= 6
plot(x0,y0,'o',x1,y1,'o',x2,y2,'o',x3,y3,'o')        // nodal points
      for i=1:101
      x= 1+ (i-1)*4*0.01
      y=y0* (x-x1)*(x-x2)*(x-x3)   / ((x0-x1)*(x0-x2)*(x0-x3) ) +...
         y1* (x-x0)*(x-x2)*(x-x3)   / ((x1-x0)*(x1-x2)*(x1-x3) ) + ...
         y2* (x-x0)*(x-x1)*(x-x3)   / ((x2-x0)*(x2-x1)*(x2-x3) ) +...
         y3* (x-x0)*(x-x1)*(x-x2)   / ((x3-x0)*(x3-x1)*(x3-x2))
         xp(i)=x
         yp(i)=y
      end
plot(xp,yp,'k','LineWidth',3)                          // polynomial
            ypp=interpln([x0 x1 x2 x3; y0 y1 y2 y3],xp)
            plot(xp,ypp,'r','LineWidth',3)
```

Sometimes, Lagrange polynomial exhibits an unusual behavior:



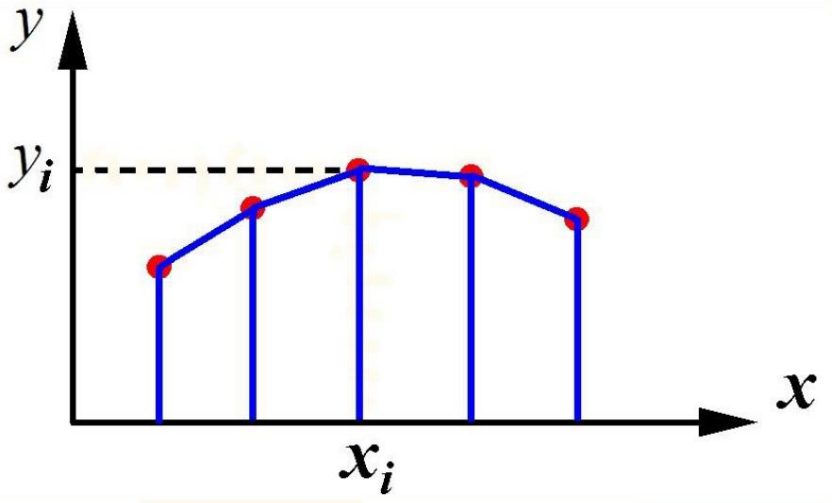
This may happen when degree *n* of the polynomial is large

This is a specific property of polynomials.

As a result, we get a bad approximation of function *f(x)*.

Even at *n*=4 and *n*=5 Lagrangian's polynomial happens to show inappropriate behavior:

Chapter 6. Interpolation by splines



recall approximation/interpolation by the linear function:

$$f(x) \approx y_i + (y_{i+1} - y_i)(x - x_i)/h$$

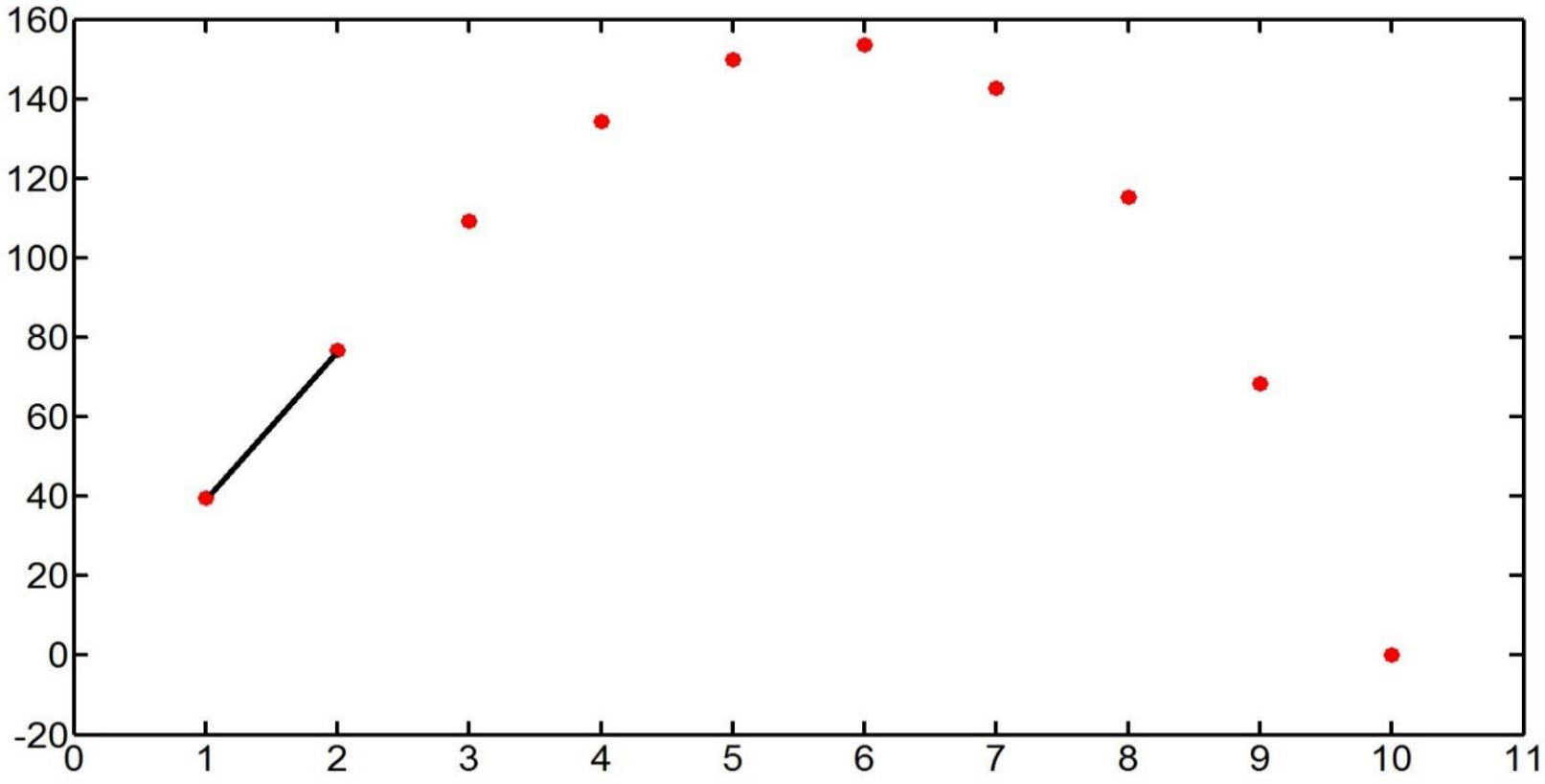
No spikes, splashes in this case. However, a drawback:

there are jumps, discontinuities of the first-order derivative

which is $(y_{i+1} - y_i)/h$ *when* $x_i < x < x_{i+1}$,

and $(y_{i+2} - y_{i+1})/h$ *when* $x_{i+1} < x < x_{i+2}$.

That is why, it was suggested to use splines for interpolation purposes



Spline $S(x)$ is a train of linked parabolas of degree 2 or 3, whose derivative dS/dx is continuous everywhere, including points x_i **(invented in 1946).**

$$S(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2$$

at $x_i \le x \le x_{i+1}$, $i = 0, 1, \ldots, n-1$

Let us find coefficients a_i , b_i , c_i using 3 conditions:

1) The condition that $S(x)$ equals to given y_i at the left endpoint x_i :

$$S(x_i) = y_i \quad => \quad a_i = y_i \qquad i = 0, 1, \ldots, n-1$$

$$S(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2$$

at $x_i \le x \le x_{i+1}$, $i = 0, 1, \ldots, n-1$

Let us find coefficients a_i , b_i , c_i using 3 conditions:

1) The condition that $S(x)$ equals to given y_i at the left endpoint x_i :

$$S(x_i) = y_i \quad => \quad a_i = y_i \qquad i = 0, 1, \ldots, n-1$$

2)The condition that $S(x)$ equals to given value y_{i+1} at x_{i+1} :

$$S(x_{i+1}) = y_{i+1} \quad =>$$

$$=> \quad a_i + b_i \cdot (x_{i+1} - x_i) + c_i \cdot (x_{i+1} - x_i)^2 = y_{i+1}$$

$$S(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2$$

at $x_i \le x \le x_{i+1}$, $i = 0, 1, \ldots, n-1$

Let us find coefficients a_i , b_i , c_i using 3 conditions:

1) The condition that $S(x)$ equals to given y_i at the left endpoint x_i :

$$S(x_i) = y_i \quad => \quad a_i = y_i \qquad i = 0, 1, \ldots, n-1$$

2)The condition that $S(x)$ equals to given value y_{i+1} at x_{i+1} :

$$S(x_{i+1}) = y_{i+1} \quad =>$$

$$=> \quad a_i + b_i \cdot (x_{i+1} - x_i) + c_i \cdot (x_{i+1} - x_i)^2 = y_{i+1}$$

3) The condition that dS/dx is continuous at node x_{i+1} :

$$dS/dx = b_i + 2\,c_i \cdot (x - x_i) \qquad at \;\; x_i \le x \le x_{i+1}$$

$$b_i + 2\,c_i \cdot (x_{i+1} - x_i) = b_{i+1} + 2\,c_{i+1} \cdot (x_{i+1} - x_{i+1})$$

$$\begin{cases} b_i + 2\,c_i \cdot (x_{i+1} - x_i) = b_{i+1} & (1) \qquad i = 0, 1, \ldots, n-2 \\ b_i \cdot (x_{i+1} - x_i) + c_i \cdot (x_{i+1} - x_i)^2 = y_{i+1} - y_i & (2) \end{cases}$$

see condition 2) above

$$i = 0, 1, \ldots, n-1$$

We get 2n-1 equations for 2n coefficients b_i, c_i

+ additional condition $c_0 = 0$

($S(x) = a_0 + b_0 \cdot (x - x_0) \qquad$ at $\; x_0 \le x \le x_1$)

In fact, coefficients b_i, c_i can be calculated <u>sequentially</u> for $i=1,2,\ldots,n$:

b_1 , c_1 ,

then b_2 *from (1),*

then c_2 *from (2),*

then b_3 *from (1), . . .*

Theorem on the error of interpolation:

If $f'''(x)$ is continuous on $[x_0, x_n]$, then

$$|f(x)-S(x)| \le \max |f'''(x)| h^3/12,$$

where $h=\max |x_{i+1}-x_i|$.

Proof see in:
"Optimal Error Bounds for Quadratic Spline Interpolation"
Francois Dubeau
JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS,
Vol. 198, pages 49-63, 1996

We notice that the second-order derivative of

$$S(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2$$

is $\frac{d^2 S}{dx^2} = 2 c_i$

Therefore, it is constant at each interval

$$x_i \leq x \leq x_{i+1}$$

and usually jumps when x **moves from** $x_i \leq x \leq x_{i+1}$ **to next segment, since** c_i **are usually different in different segments.**

Using cubic splines

$$S_{cub}(x) = a_i + b_i \cdot (x - x_i) + c_i \cdot (x - x_i)^2 + d_i \cdot (x - x_i)^3$$

$$x_i \le x \le x_{i+1}\ , \qquad i = 0, 1, \ldots, n-1,$$

with coefficient d_i ***, we can provide the continuity of*** d^2S_{cub}/dx^2 ***at the nodes*** x_i **.**

Indeed, the derivative

$$d^2S_{cub}/dx^2 = 2c_i + 6d_i \cdot (x - x_i)$$

changes in $x_i \le x \le x_{i+1}$

and therefore its values at the ends of interval can be adjusted in a proper way.

We set 4 conditions:

1) $S_{cub}(x)$ is equal to y_i at left end of segment

$x_i \le x \le x_{i+1}$: $y_i = a_i$

2) $S_{cub}(x)$ is equal to y_{i+1} at right end of the segment

$$y_{i+1} = a_i + b_i \cdot (x_{i+1} - x_i) + c_i \cdot (x_{i+1} - x_i)^2 + d_i \cdot (x_{i+1} - x_i)^3$$

3) **First-order derivative:**

$$dS_{cub}/dx = b_i + 2c_i \cdot (x - x_i) + 3d_i \cdot (x - x_i)^2$$

the condition of equal derivatives at the right endpoint:

$$b_i + 2c_i \cdot (x_{i+1} - x_i) + 3d_i \cdot (x_{i+1} - x_i)^2 = b_{i+1}$$

$i = 1, 2, \ldots, n-1$

4) Second-order derivative :

$$d^2S_{cub}/dx^2 = 2c_i + 6d_i \cdot (x-x_i)$$

the condition of equal second derivatives at the right endpoint:

$$2c_i + 6d_i \cdot (x_{i+1}-x_i) = 2c_{i+1} \qquad i=1,2,\ldots,n-1$$

We have got 2n+2(n-1) equations with respect to 4n coefficients of spline S_{cub} .

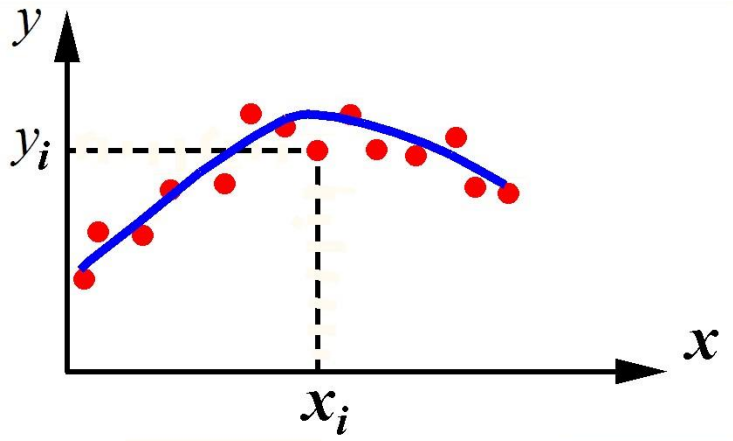
We add two conditions $d^2S_{cub}/dx^2 = 0$ at endpoints x_0, x_n :

$$c_0 = 0, \qquad 2c_{n-1} + 6d_{n-1} \cdot (x_n - x_{n-1}) = 0$$

Finally, we have 4n equations with respect to 4n coefficients.

Chapter 7. Least-Squares approximation

(Least-Squares fitting procedure)



Suppose that given values x_i , y_i show fluctuations, deviations, from some smooth curve (for example, because of influence of random factors).

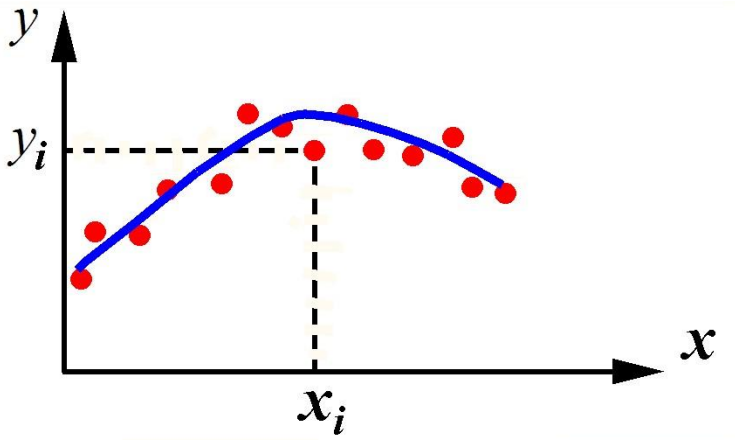
<u>Problem</u>: Find a function $p(x)$ that is smooth and well approximates x_i , y_i .

Consider a polynomial

$p(x)=a_0 + a_1x + a_2x^2 + \ldots + a_mx^m, \quad m<n$

We can use either condition

$$\sum_{i=0}^{n} | p(x_i) - y_i | \rightarrow \min$$



or

$$\sum_{i=0}^{n} [p(x_i) - y_i]^2 / (n+1) \rightarrow \min$$

(title of method)

$$\sum_{i=0}^{n} [\, p(x_i) - y_i \,]^2 \quad \rightarrow \quad \min$$

$$\sum_{i=0}^{n} [a_0 + a_1 x_i + \ldots + a_k x_i^k + \ldots + a_m x_i^m - y_i \,]^2 \rightarrow \min$$

The sum involves n+1 square brackets

This is the condition for finding a_i

Let us denote the left-hand side by

$$J(a_0, a_1, \ldots, a_k, \ldots, a_m) = \sum_{i=0}^{n} [\, p(x_i) - y_i \,]^2$$

This is a function of $m+1$ variables a_j

Necessary condition of a minimum: $\partial J/\partial a_k = 0$

$$\partial J/\partial a_k = \sum_{i=0}^{n} 2[\, p(x_i) - y_i] \cdot \partial p(x_i)/\partial a_k = 0$$

$$\sum_{i=0}^{n} [\, p(x_i) - y_i] \cdot x_i^k = 0 \qquad k=0,1,\ldots m$$

$$\sum_{i=0}^{n} [a_0 + a_1 x_i + \ldots + a_m x_i^m] \cdot x_i^k = \sum y_i x_i^k$$

$$\sum_{i=0}^{n} [a_0 x_i^k + a_1 x_i^{k+1} + \ldots + a_m x_i^{k+m}] = \sum y_i x_i^k$$

and we get the system of $m+1$ **linear equations,** $k=0,1,\ldots m$**:**

$$a_0 \sum_{i=0}^{n} x_i^k + a_1 \sum_{i=0}^{n} x_i^{k+1} + \ldots + a_m \sum_{i=0}^{n} x_i^{k+m} = \sum_{i=0}^{n} y_i x_i^k \qquad (*)$$

Theorem If $m \le n$, then there exists a unique solution of the system of equations (*) with respect to coefficients a_k of the polynomial $p(x)$ (proof is omitted).

In particular, at m=n this polynomial is equivalent to Lagrange's polynomial (which passes through each node).

We used the necessary condition of minimum

$\partial J/\partial a_k = 0$.

The fact that obtained a_k **determine indeed a minimum, not maximum, is evident from the quadratic dependence of**

$$J(a_0 , a_1, + \ldots + a_m) = \sum_{i=0}^{n} [\, p(x_i) - y_i \,]^2 \quad \text{on} \quad a_k ,$$

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$$

Notice. Polynomials

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m \qquad (**)$$

where m>10 are difficult to use in practice, as determinant of the system (*) becomes very close to 0, and the system becomes ill-conditioned.

Then, for least squares approximation, one can use **Chebyshev polynomials** $T_k(x)$ instead of (**):

$$a_0 T_0 + a_1 T_1(x) + a_2 T_2(x) + \ldots + a_m T_m(x)$$

$$\sum_{i=0}^{n} [a_0 T_0 + a_1 T_1(x_i) + \ldots + a_k T_k(x_i) + \ldots + a_m T_m(x_i) - y_i]^2 \rightarrow$$

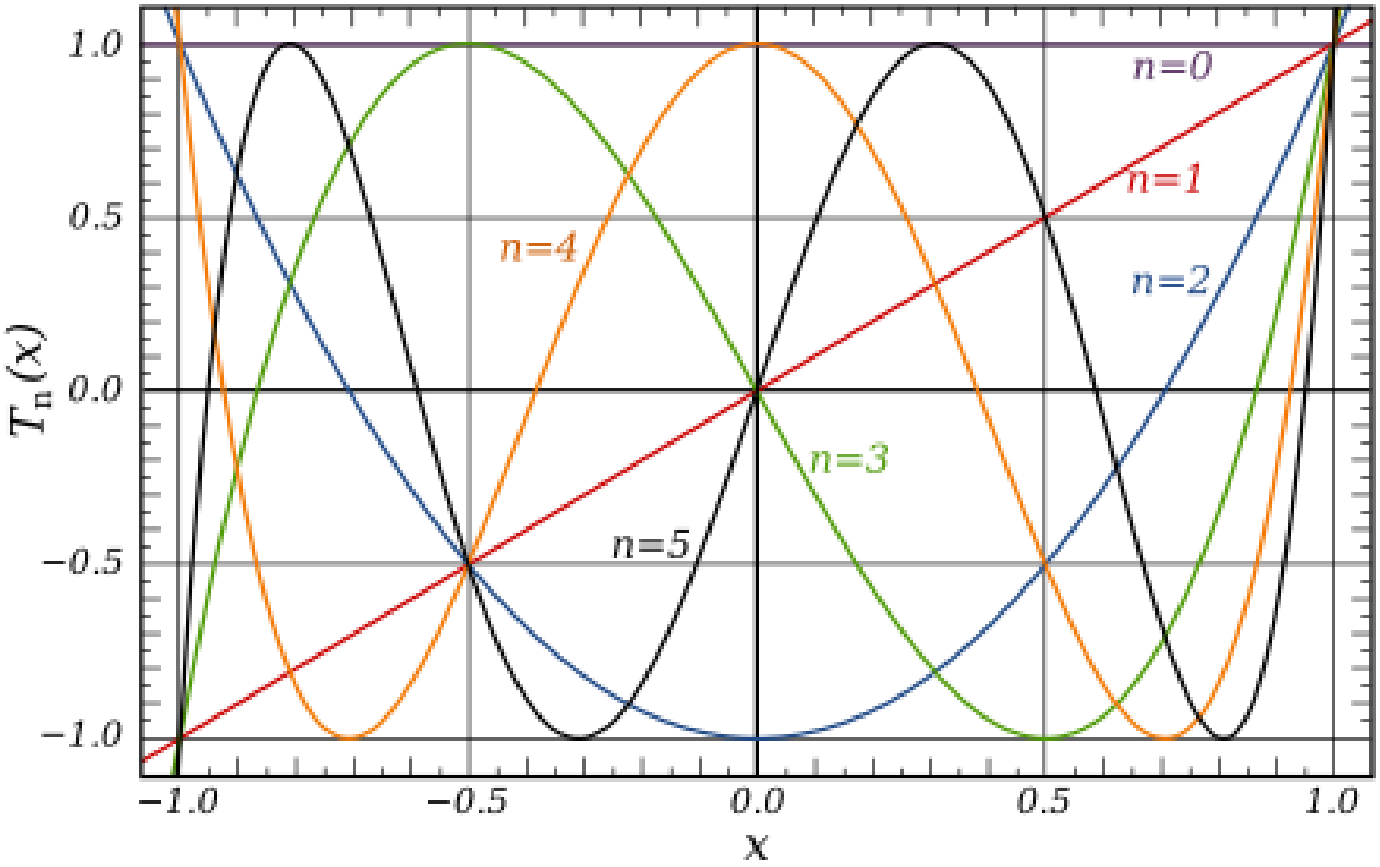
$$\rightarrow \min$$

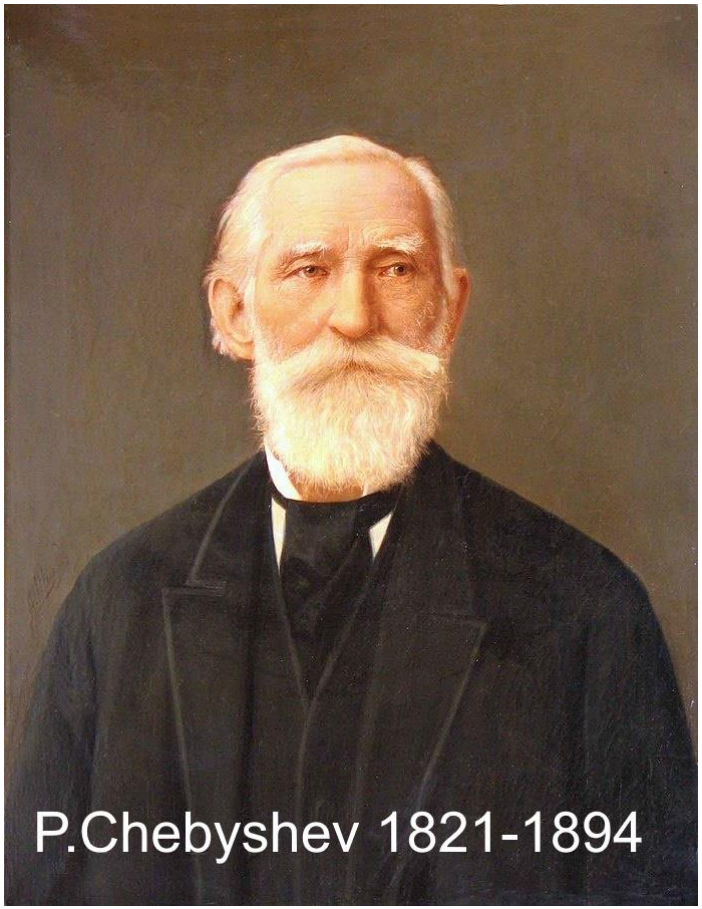
***Definition of* Chebyshev polynomials :**

$$T_0=1 \qquad T_1(x)=x \qquad T_2(x)=2x^2-1$$

$$T_k(x)=2xT_{k-1}(x)-T_{k-2}(x), \qquad k=3,4,\ldots$$

$$-1 \le x \le 1$$

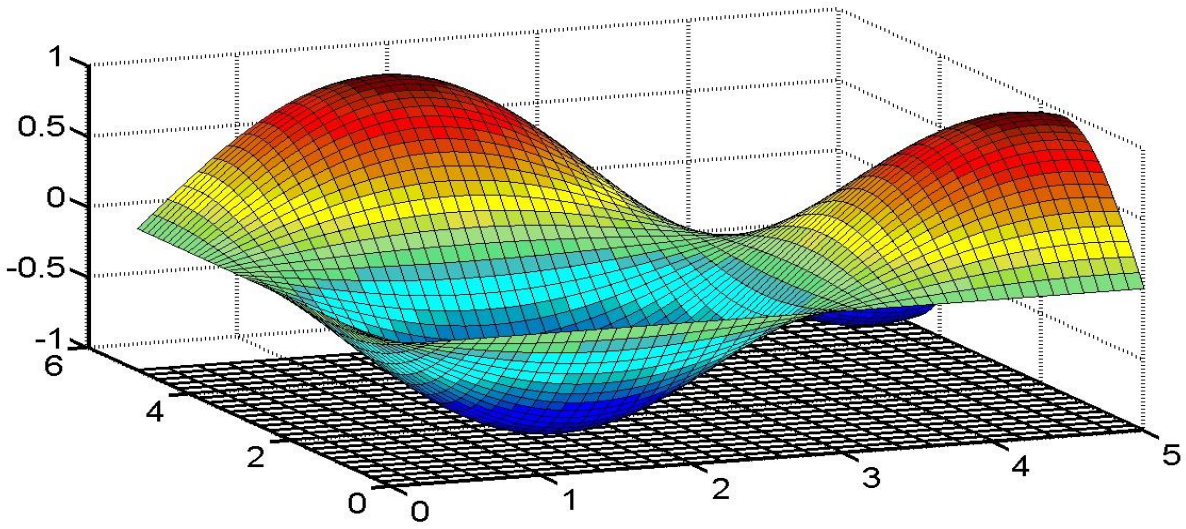




For a different interval, one can use the substitution

$$t=(2x-x_0-x_n)/(x_n-x_0)$$

Chapter 8. Method of coordinate descent for finding a minimum of a multivariable function



Let we have a function of n variables

$$F(x_1, x_2, \ldots, x_n) \qquad a_i < x_i < b_i$$

and we seek coordinates of a point at which the function attains a minimum.

(If you need to find a maximum, then consider the function $-F$ and seek a minimum.)

Method of coordinate descent:

an idea is to vary coordinates by turns for the search of a minimum.

Let us choose some initial point

$$x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$$

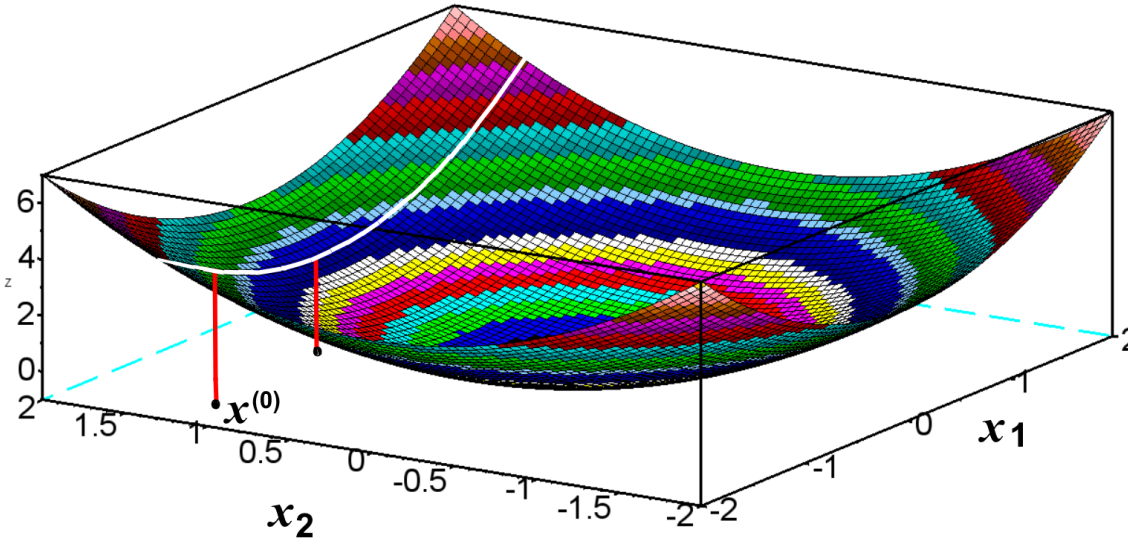
and vary the first coordinate, keeping others:

$$(x_1, x_2^{(0)}, \ldots, x_n^{(0)})$$

$$a_1 \le x_1 \le b_1$$

By calculation of F at nodes on this segment, one can find a point of minimum

$$(x_1^{(1)}, x_2^{(0)}, \ldots, x_n^{(0)}).$$



After that, we fix all coordinates except for

x_2 **and search a minimum under variation of** x_2 **:**

$(x_1^{(1)}, x_2, \ldots, x_n^{(0)})$

$(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(0)})$

. . . . . .

$(x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}) = x^{(1)}$

$x^{(2)}$

$x^{(3)}$

$x^{(k)}$

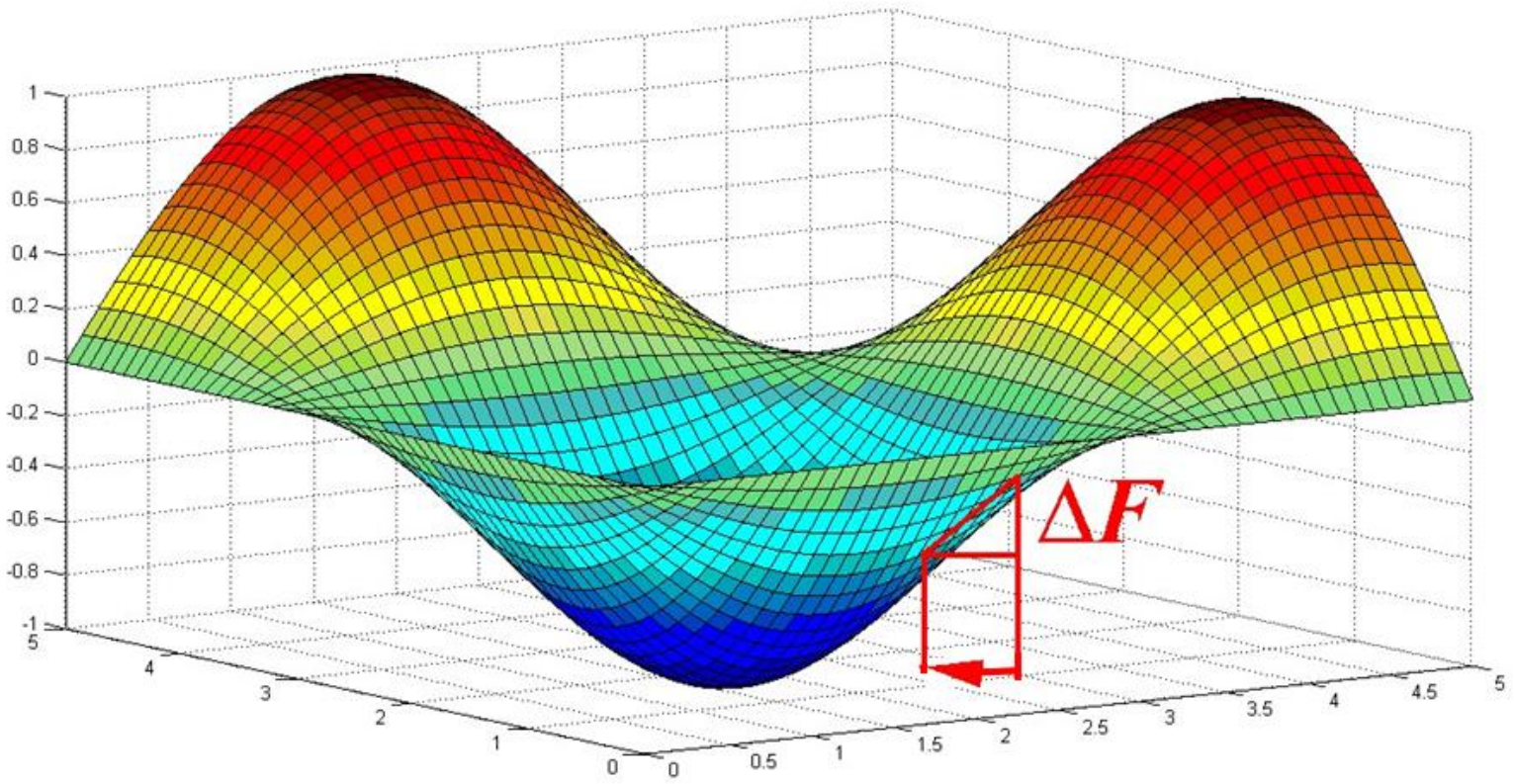
At each step, $F(x^{(k)}) \leq F(x^{(k-1)})$

Let us choose an initial point: $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$,

calculate partial derivatives, and compose the gradient

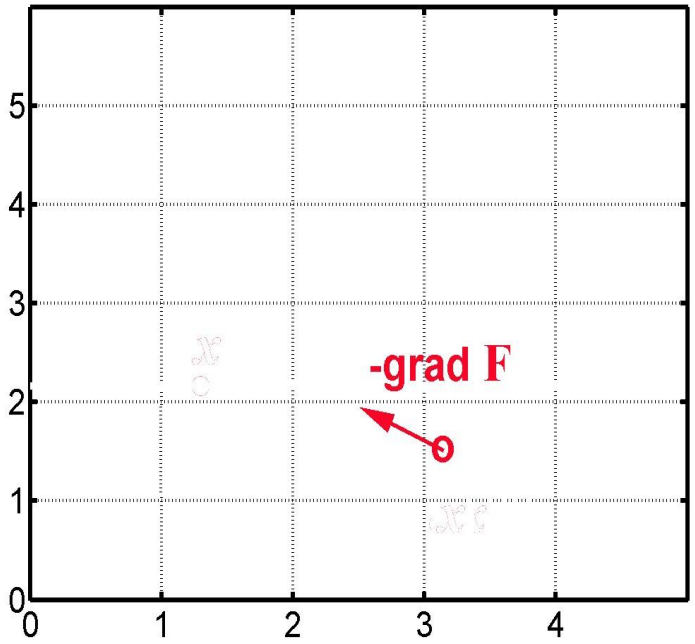
$$\mathbf{grad}\ F(x^{(0)}) = \sum_{i=1}^{n} \bar{e}_i\, \partial F(x^{(0)})/\partial x_i$$

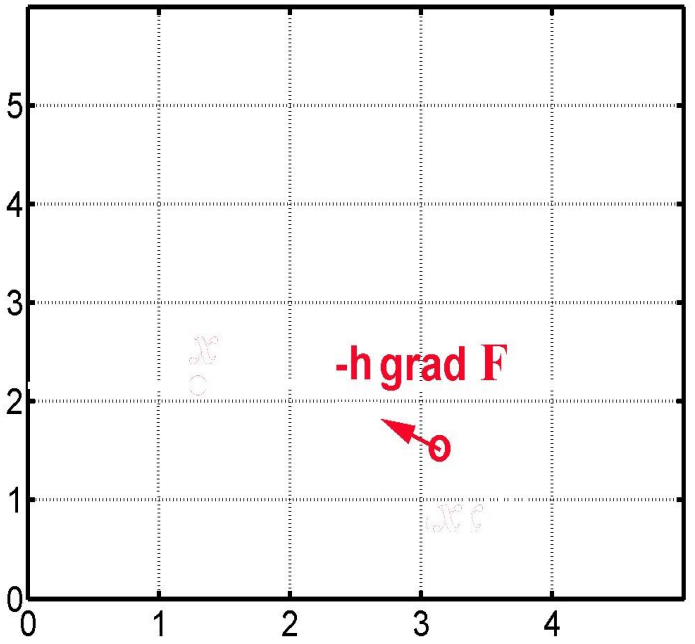
As known, vector $\mathbf{grad}\ F(x^{(0)})$ indicates the direction of most rapid rise of the function $F(x)$ at point $x^{(0)}$ in the plane (x,y).



If we make a step of length $\left|\mathrm{grad}\,F(x^{(0)})\right|$ in the direction of antigradient, then we arrive at a point where gradient changes, therefore, the direction of fastest decrease changes.

It is reasonable to consider a shorter vector $-h\cdot\mathrm{grad}F(x^{(0)})$, where h is a small parameter, in order to correct the direction of descent:

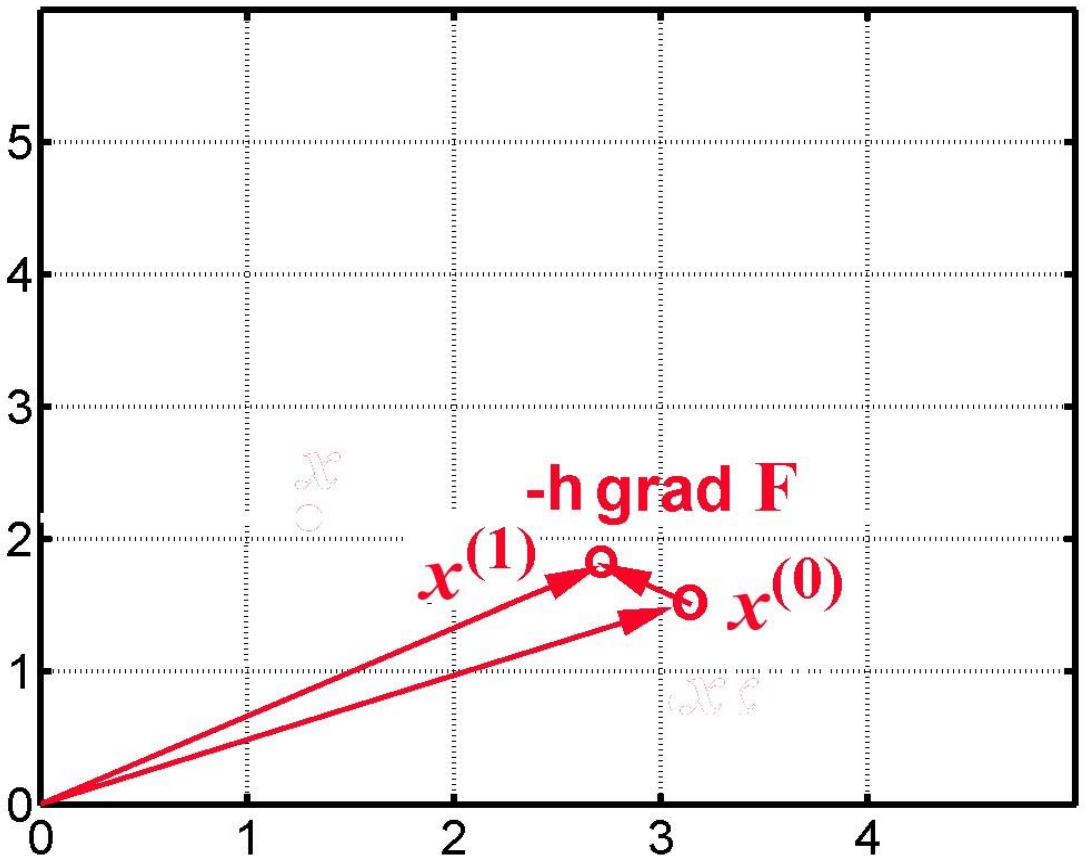




$x^{(1)} = x^{(0)} - h \cdot \mathrm{grad}\, F(x^{(0)})$ **- in vector form.**

Cartesian components:

$$x_i^{(1)} = x_i^{(0)} - h \cdot \partial F(x^{(0)}) / \partial x_i$$



Similarly, we perform further steps, which lead to a minimum of F(*x*):

$$x_i^{(k+1)} = x_i^{(k)} - h \cdot \partial F(x^{(k)}) / \partial x_i$$

$$x_i^{(k+1)} = x_i^{(k)} - h \cdot \partial F(x^{(k)})/\partial x_i$$

Notice:

As soon as we approach a minimum, the derivatives $\partial F(x^{(k)})/\partial x_i$ decrease; therefore, the difference $x^{(k+1)} - x^{(k)}$ decreases as well.

In Scilab code, it makes sense to set another condition to stop computations:

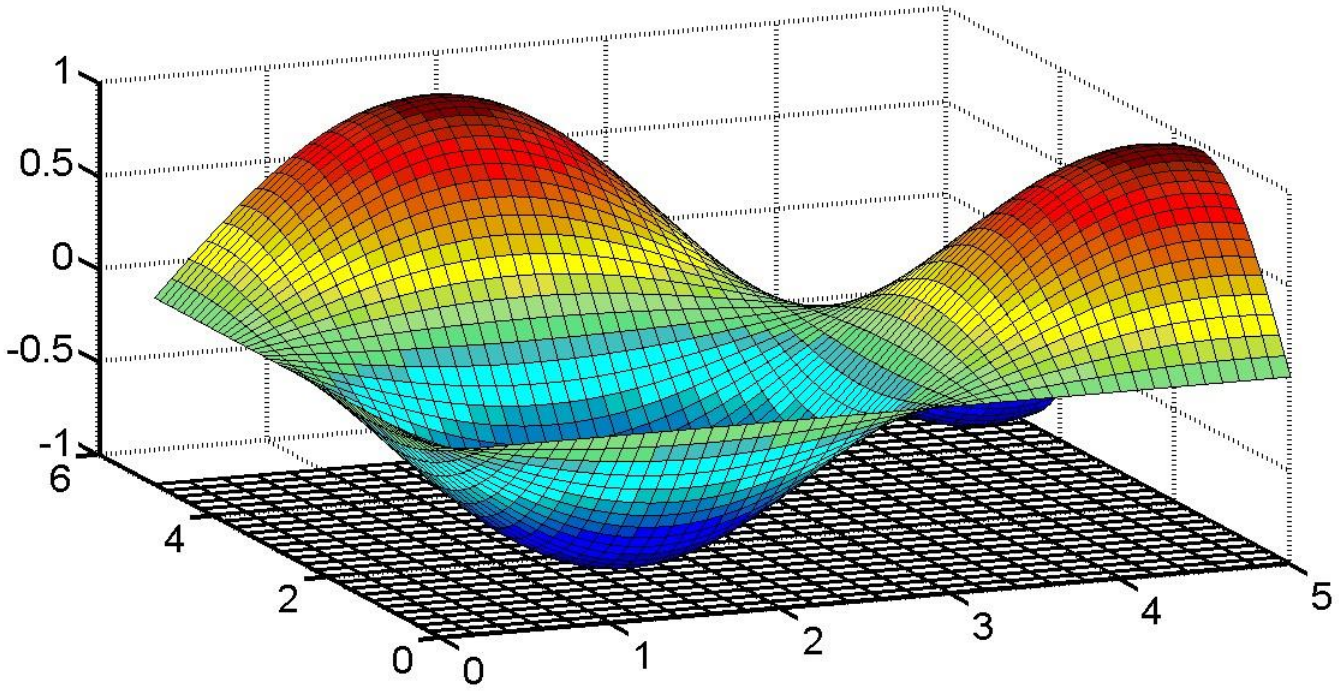
$$\left| x_i^{(k+1)} - x_i^{(k)} \right| < \varepsilon \qquad \text{for all} \quad i$$

Or $$\left| F(x^{(k+1)}) - F(x^{(k)}) \right| < \varepsilon$$

(instead of setting total number of steps k=1:300**)**

Remark 1.

Possible non-uniqueness of local minima:



Remark 2.

Often, in the method of gradient descent, one cannot find an analytical expression for partial derivatives of F.

Then some numerical methods for finding the derivatives can be used, see Chapter 11.

Method of most rapid gradient descent for finding a minimum of a function

$$z = \mathbf{F}(x_1, x_2, \ldots, x_n)$$

Recall Method of gradient descent:

Choose some $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$

Calculate

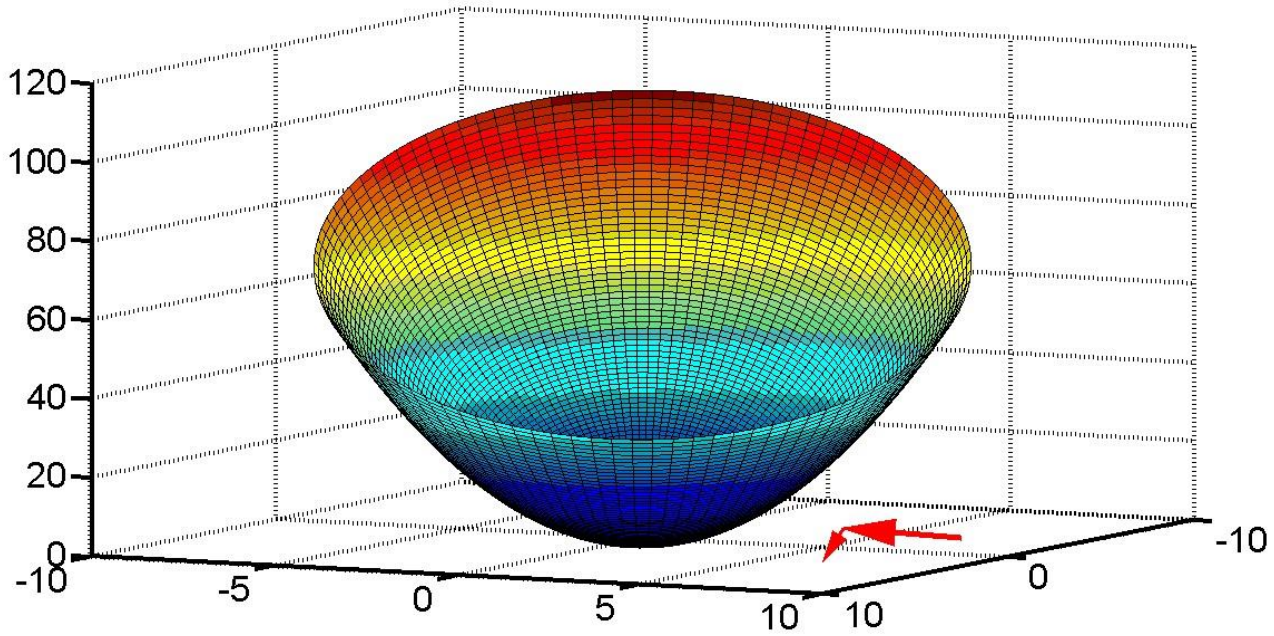
$$\mathbf{grad}\, F(x^{(0)}) = \sum_i \bar{e}_i \, \partial F(x^{(0)})/\partial x_i$$

Sequential approximations:

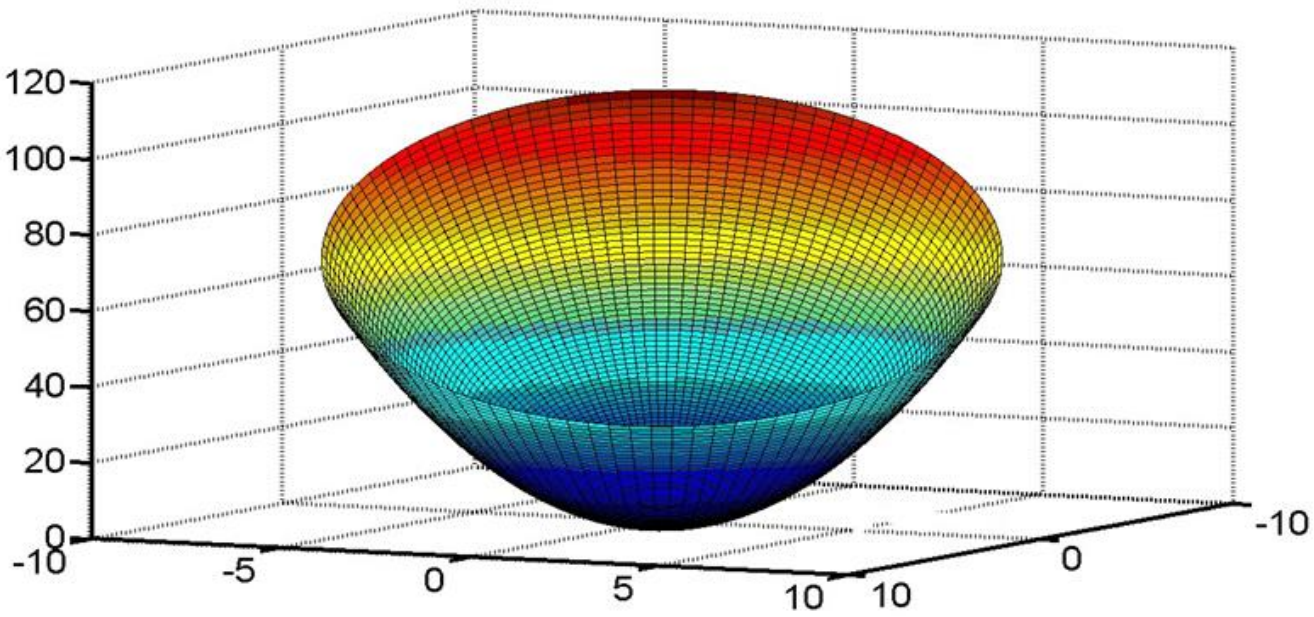
$$x_i^{(k+1)} = x_i^{(k)} - h \cdot \partial F(x^{(k)})/\partial x_i$$

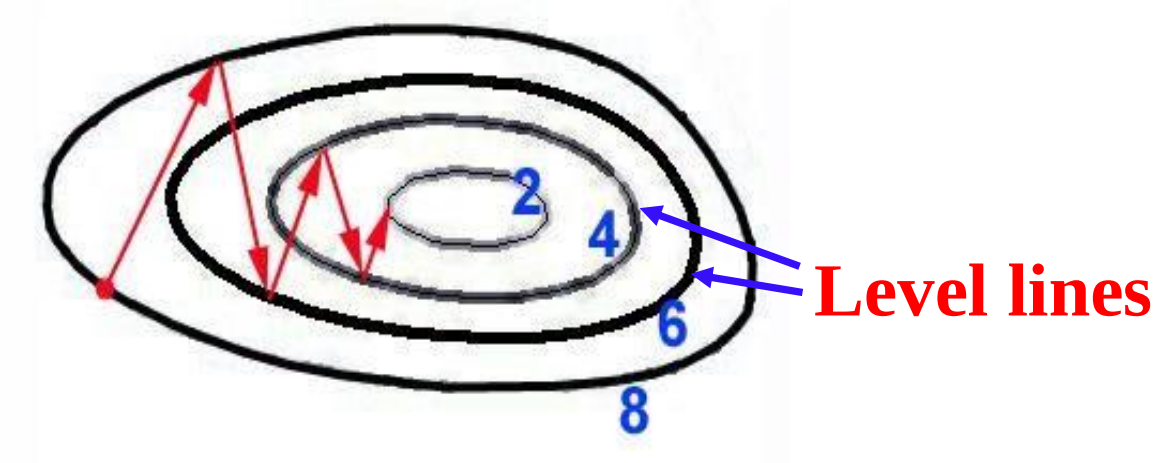
The parameter h governs the length of vector.

If h is too small, then sequence $X^{(k)}$ approaches a solution *slow*, as it takes too many steps and a lot of computing time.



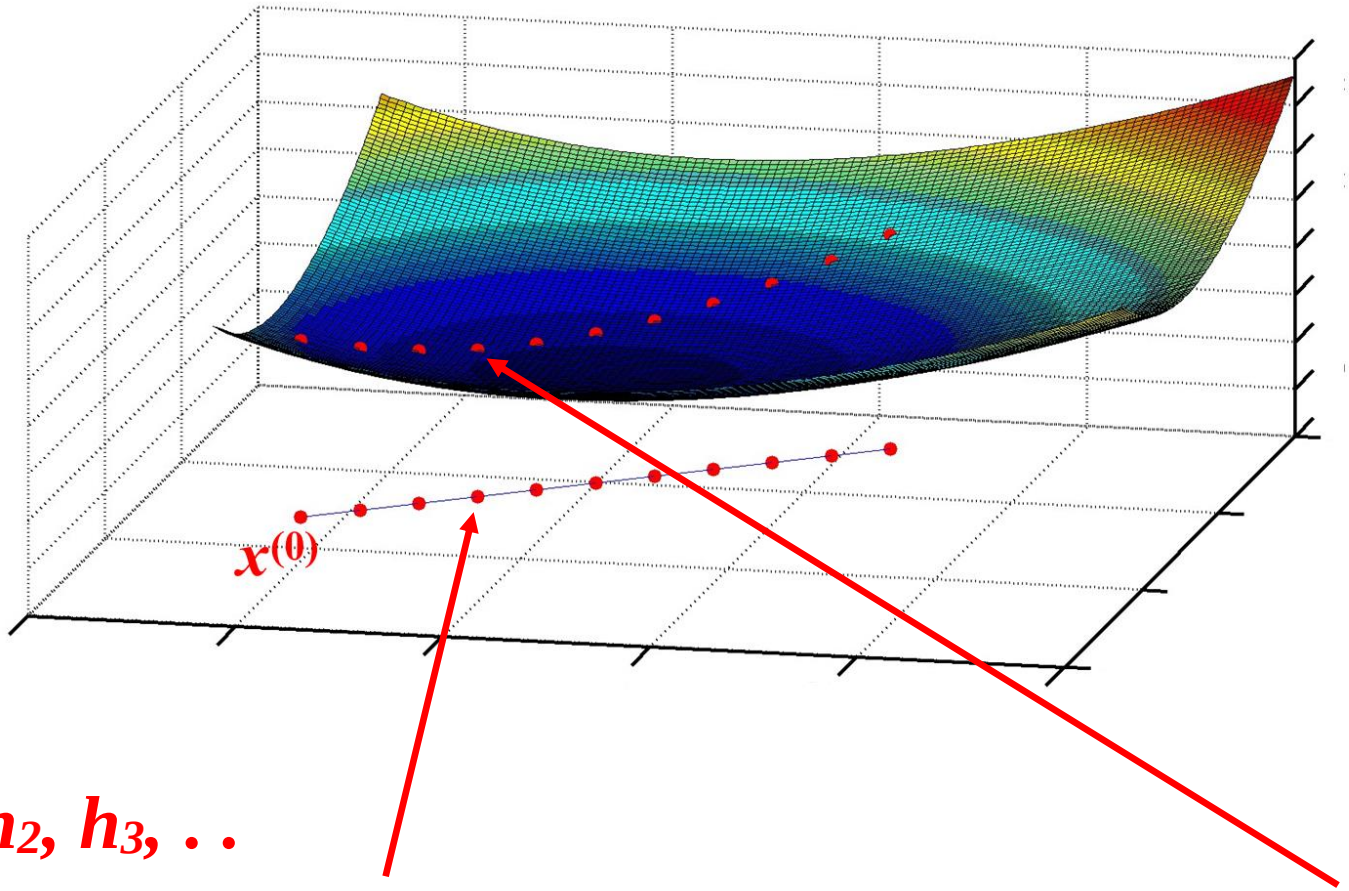
If h is too large, then sequence $x^{(k)}$ may converge slow as well, as it may pass by a minimum:





(or the sequence may happen to diverge if h is very large).

What is the optimal value of h ?

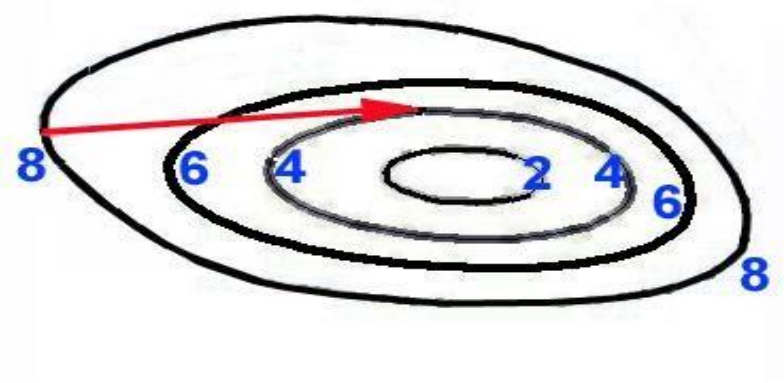


Consider various $h_1, h_2, h_3, \ldots$

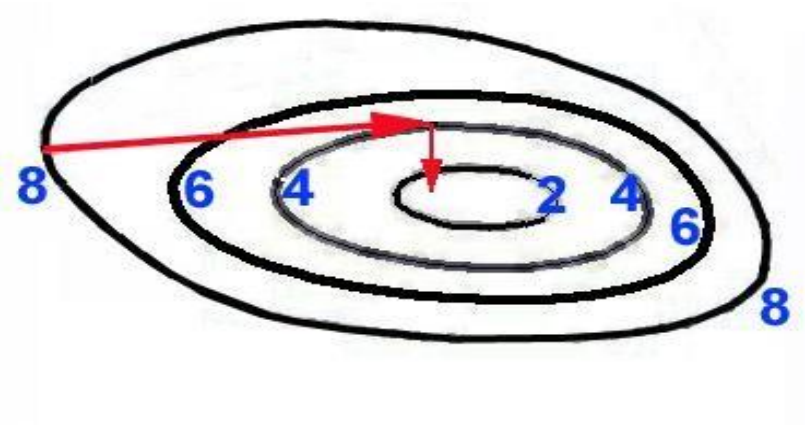
Suppose we have found such h_{opt}, that $\mathbf{F}$ attains a minimum $\mathbf{F}_{min}$

in the direction of antigradient. This can be achieved with a method of 1D optimization, for example method of golden section, see the end of this chapter.

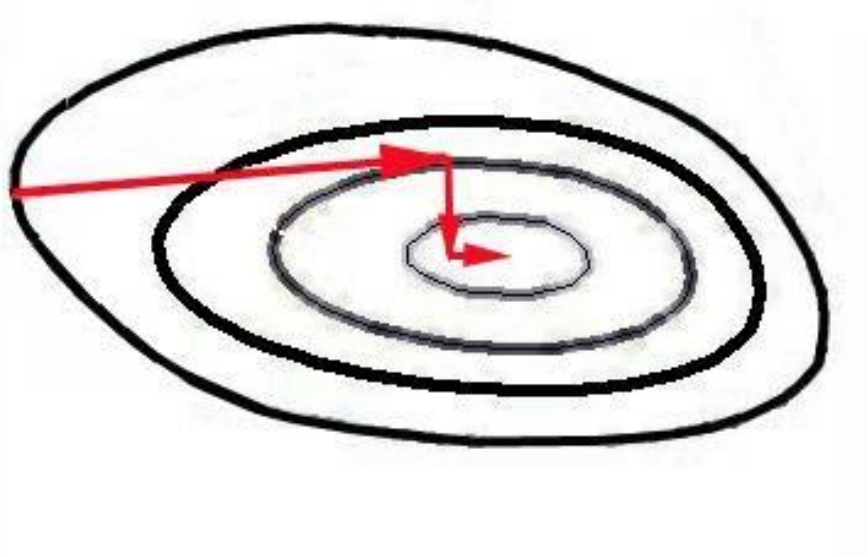
As soon as we found such optimum h_{opt}, that ensures a minimum of F, we move from $x_i^{(0)}$ to $x_i^{(1)} = x_i^{(0)} - h_{opt} \cdot \partial F(x^{(0)})/\partial x_i$



At the point of minimum $x^{(1)}$, the direction, in which the step was made, is tangent to a level line.
Antigradient calculated at this point will be normal to the level line:



Therefore next step from $x^{(1)}$ to $x^{(2)}$ will be made in the normal (orthogonal) direction, as antigradient is normal to the level line that passes through point $x^{(1)}$.

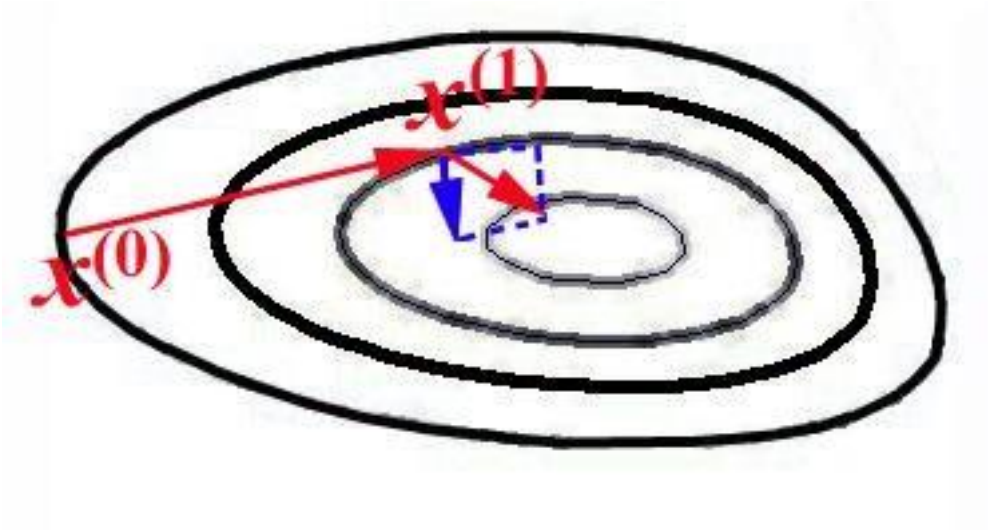


In the same way, at next steps, we seek $h^{(k)}_{opt}$ and move on:

$$x_i^{(k+1)} = x_i^{(k)} - h^{(k)}_{opt} \cdot \partial F(x^{(k)})/\partial x_i$$

Practice showed that method of <u>fastest</u> gradient descent is more efficient (from the viewpoint of necessary computing time) than method of gradient descent with a constant h .

Conjugate Gradient Method



Most rapid gradient descent:

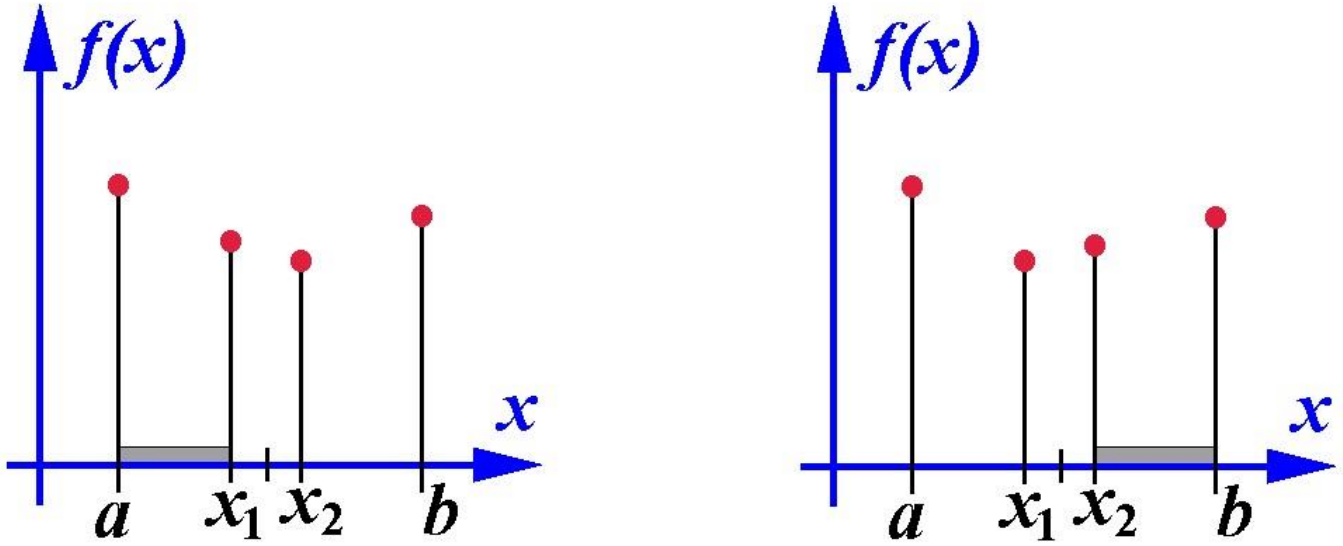
$$x^{(1)}=x^{(0)}- h_{\text{opt}}\cdot \text{grad}F(x^{(0)}) \ , \qquad x^{(2)}=x^{(1)}- h_{\text{opt}}\cdot \text{grad}F(x^{(1)})$$

Conjugate gradient method:

$$x^{(2)}=x^{(1)}- h_{\text{opt}}\left[\text{grad}F(x^{(1)}) + \beta^{(1)}\cdot \text{grad}F(x^{(0)})\right]$$

correction of the direction of step

Search of a minimum in case of a function of single independent variable. A bisection method.



Let $f(x)$ be given on $[a,b]$. We divide the segment into two equal parts and consider two points x_1 and x_2 located symmetrically about the midpoint

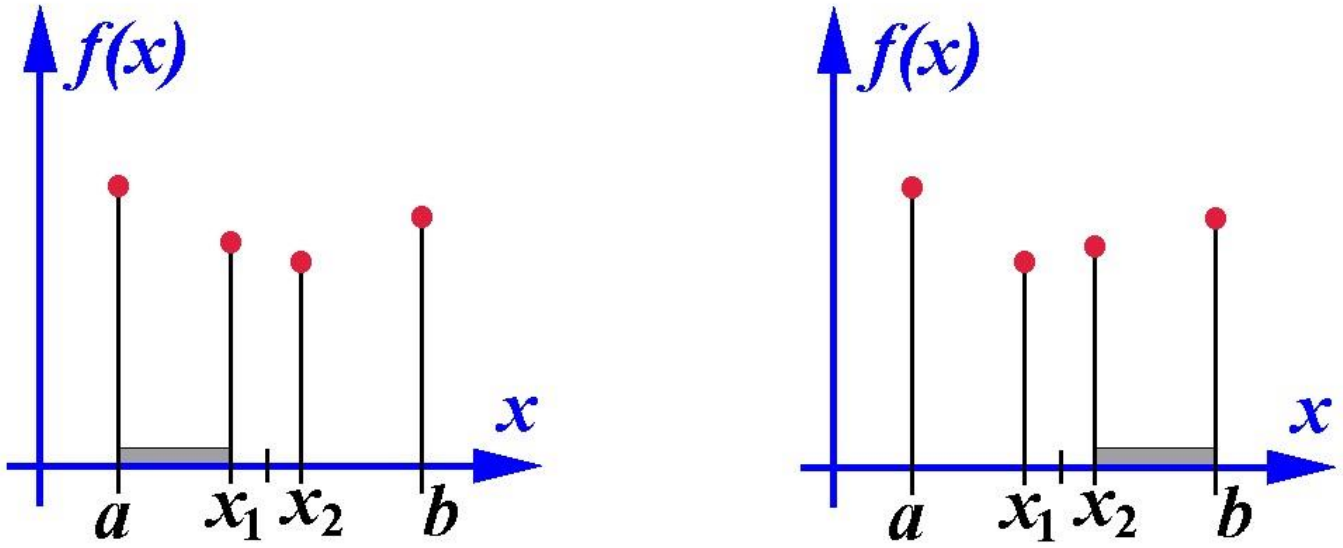
$$x_1 = 0.5\,(a+b) - \delta\,, \qquad x_2 = 0.5\,(a+b) + \delta\,,$$

where δ is small.

Let us calculate $f(x_1)$ and $f(x_2)$.

If $f(x_1) > f(x_2)$, then we drop $[a, x_1]$ and retain $[x_1, b]$

If $f(x_1) < f(x_2)$, then we drop $[x_2, b]$ and retain $[a, x_2]$

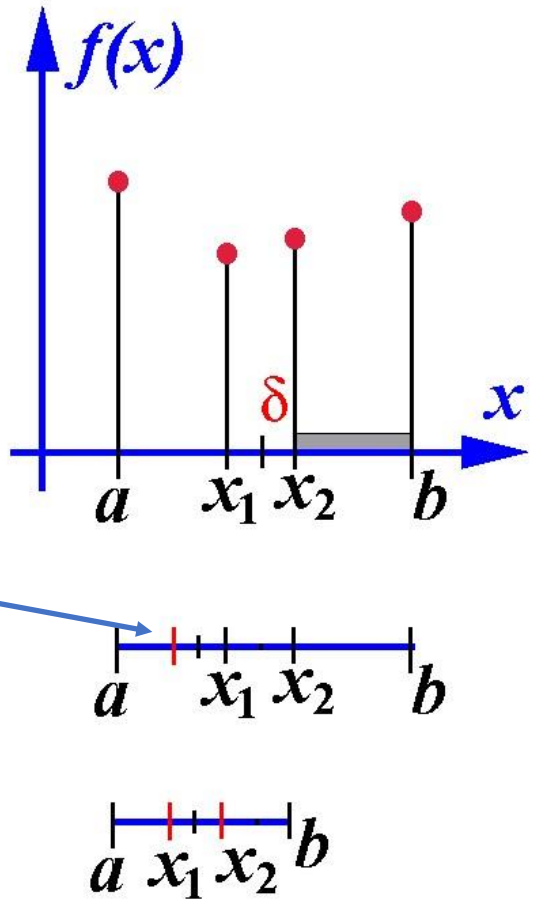


For the obtained segment $[a, x_2]$ or $[x_1, b]$ the procedure of halving the segment keeps on until the length of segment becomes small enough.

A golden section method.

In the bysection method, at each step, one needs to calculate $f(x)$ at two new points x_1, x_2 .

In a golden section method, at each step, one calculates $f(x)$ only at one new point. This can be performed by choosing



$$\delta = (\sqrt{5}/2 - 1)*(b-a)$$

In such a case, after dropping $[x_2, b]$, on the retained segment $[a, x_2]$ we already know $f(x_1)$, and x_1 is well located with respect to the middle of the retained segment $[a, x_2]$. Therefore, we only need to calculate f at the symmetric point indicated by a bar |

On the new segment $[a, x_2]$, all proportions between locations of points are the same as ones on $[a, b]$:

$$(b-a)/(x_2-a) = (x_2-a)/(x_1-a) .$$

As a consequence, further reduction of the segment can be performed in the same way.

In the method of golden section, we need two times smaller amount of computations of $f(x)$; meanwhile the length of segment is reduced at each step by the smaller factor

1.618033988… .

(not by factor 2). Therefore, the eventual gain of computational time is:

$$2 / 1.618033988$$