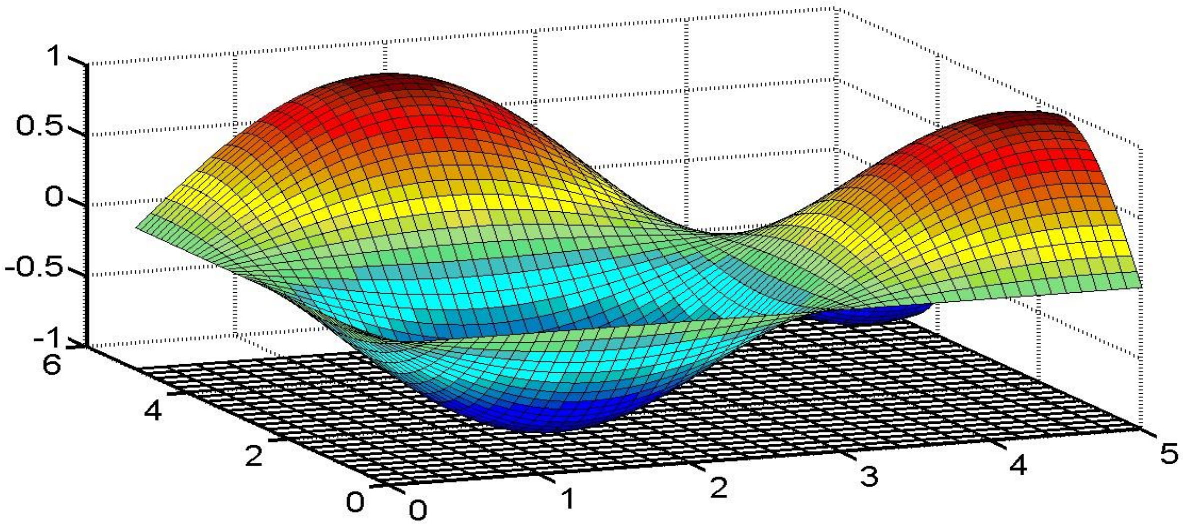
# Chapter 8. Method of coordinate descent for finding a minimum of a multivariable function




**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.)**

**The most primitive way is to introduce a lot of nodes in the domain of definition, then calculate $F$ at the nodes and use computer command "if . . . less . . . then" for identifying the node at which minimum is attained.**

**This may be very time-consuming. For example, if n=7 and each segment $[a_i , b_i]$ contains 101 node, then total number of nodes is $101^7 > 10^{14}$**

**If calculation of *F* at a single node requires, for example, 50 arithmetic operations, then total number of necessary operations is more than $5 \times 10^{15}$.**

**Intel Core i7-7700K Processor performs roughly 38 billion = $38 \cdot 10^9$ floating point operations per second. It will take $5 \times 10^{15} / 38 \cdot 10^9$ = 131600 seconds ≈ 36.5 hours to do this job.**

**Therefore, there is need in more efficient methods of finding 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

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

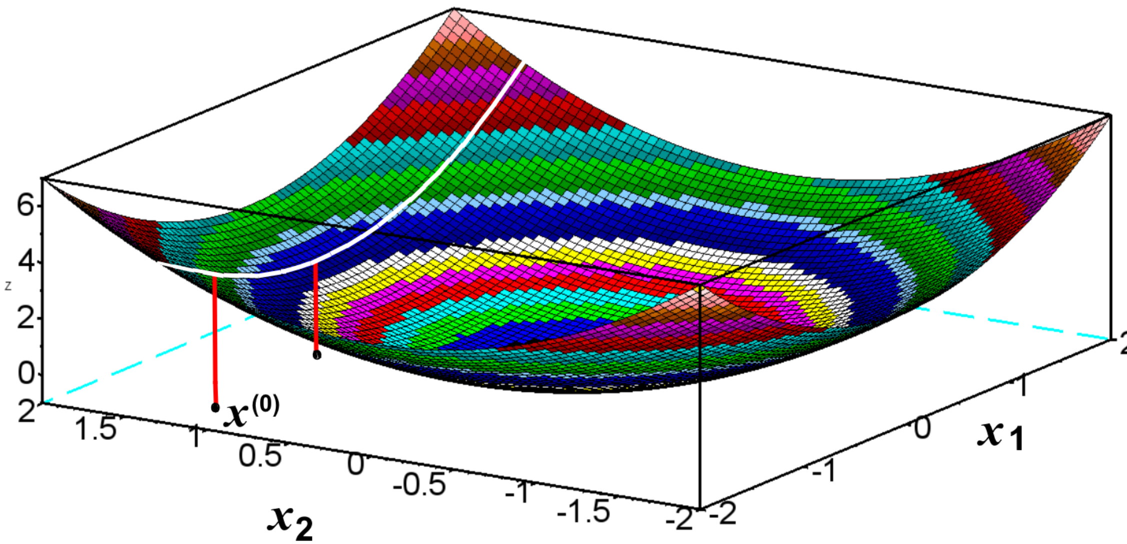
and vary the first coordinate, keeping others:

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

$$a_1 \leq x_1 \leq b_1$$

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 $x_2$ :

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

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

. . . . . . . . . . . . . . . .

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

$$x^{(2)}$$

$$x^{(3)}$$

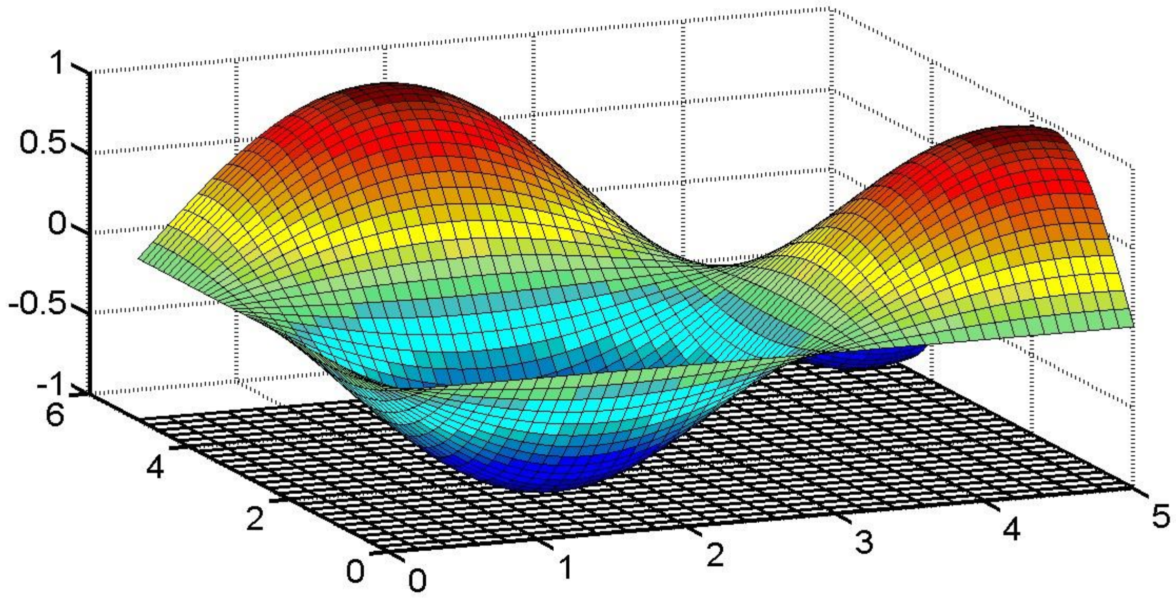
$$x^{(k)}$$

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

# Method of gradient descent

## for finding a minimum of a multivariable function

**https://en.wikipedia.org/wiki/Gradient_descent**




**Again, we have a function of $n$ variables**

$$F(x_1, x_2, \ldots, x_n)$$

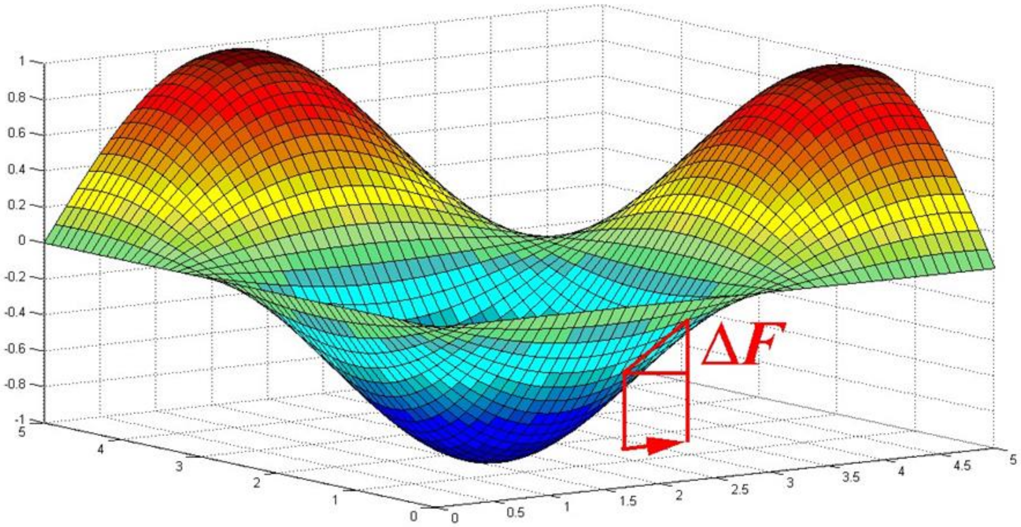
**and we seek coordinates of a point at which the function attains a local minimum.**

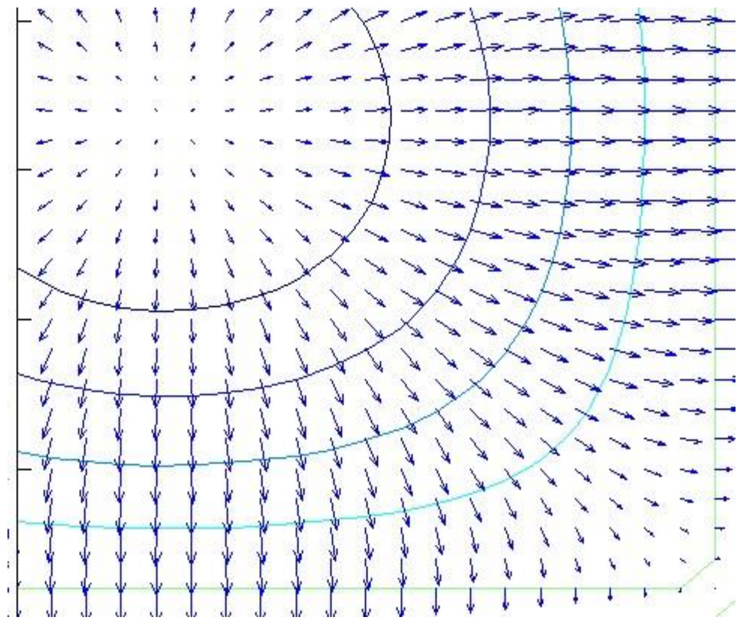
Let us choose an initial point: $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$ ,

calculate partial derivatives, and compose the gradient

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

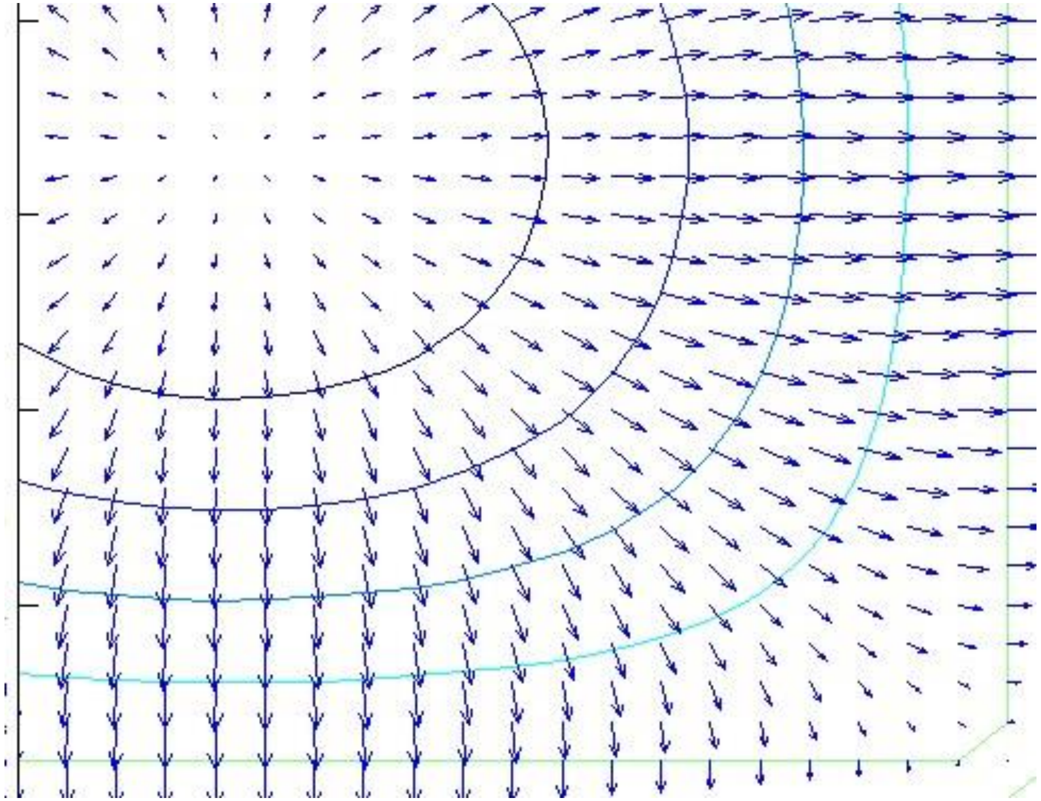
As known, vector $\mathrm{grad}\ F(x^{(0)})$ indicates the direction of most rapid <u>rise</u> of the function $F(x)$ at point $x^{(0)}$ in the plane $(x,y)$.



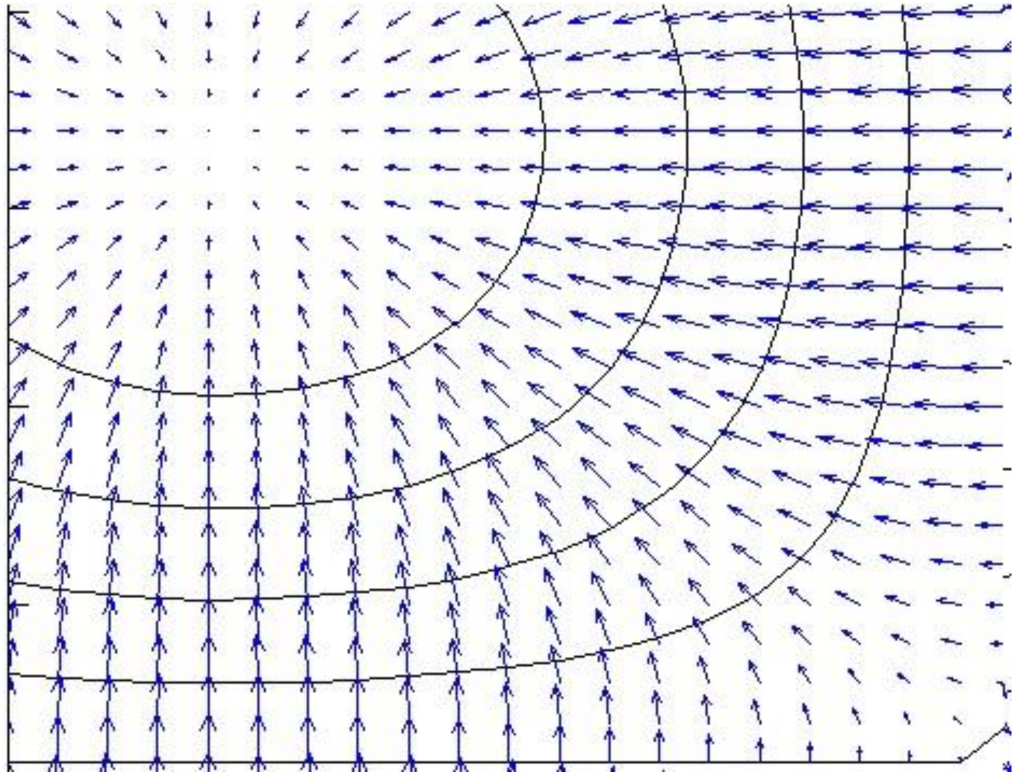



**The gradient grad F is normal to level lines**

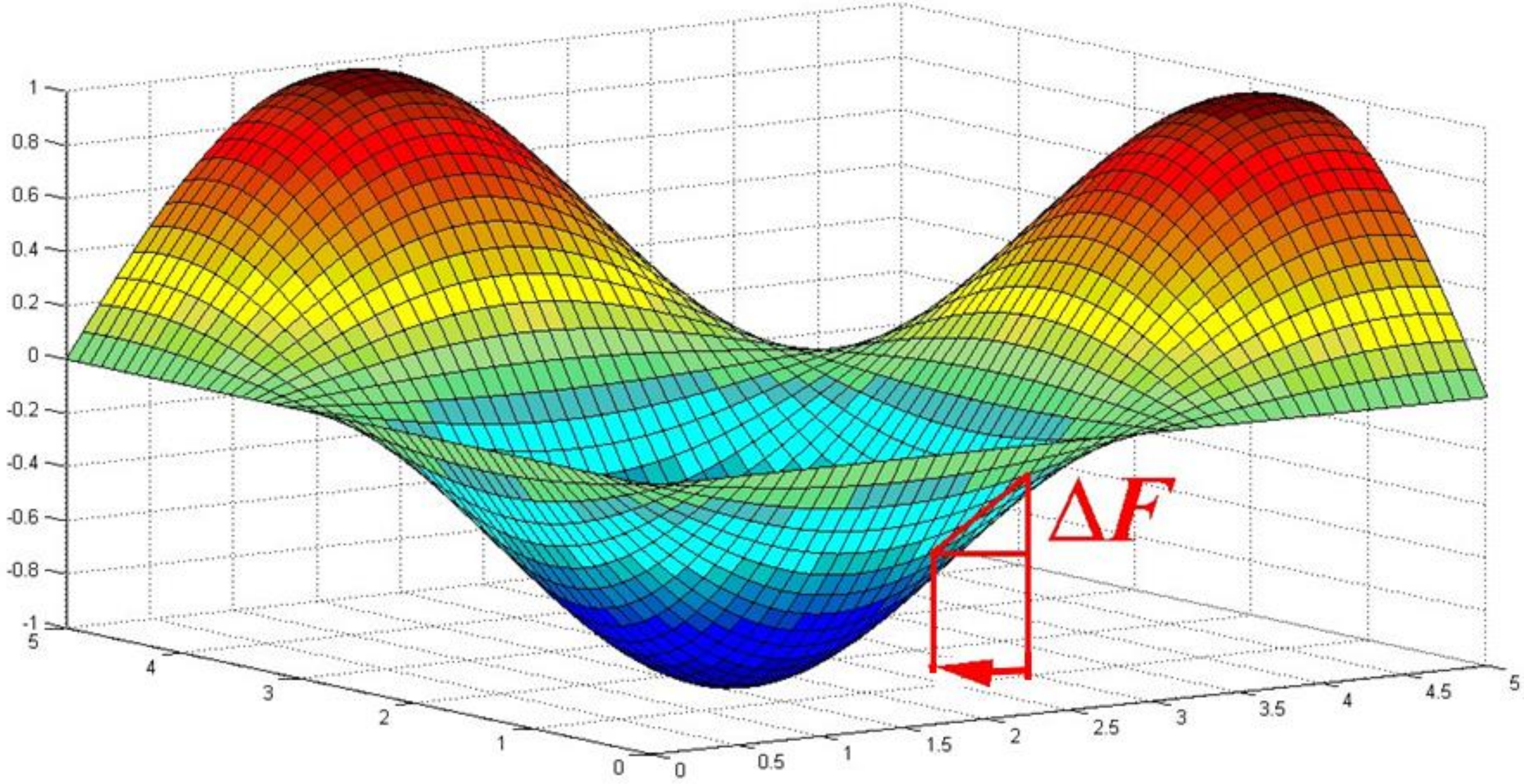
## grad F



## -grad F

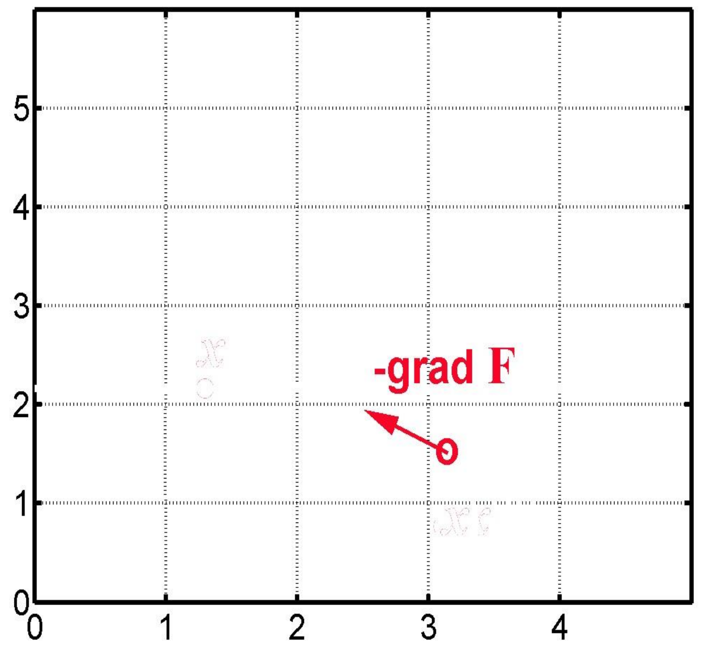


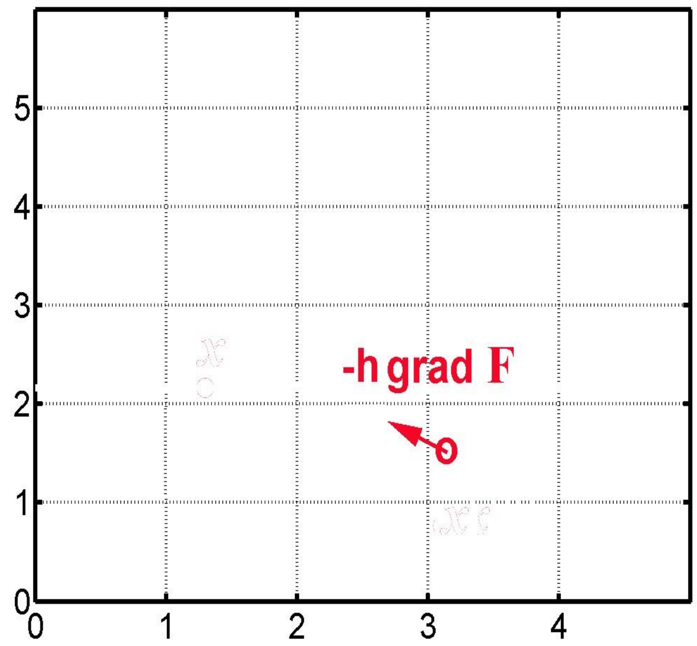
**Antigradient** $-\mathrm{grad}\ \mathrm{F}(x^{(0)})$ **indicates the direction of fastest** **<u>decrease</u> (descent) of the function** $\mathrm{F}(x)$ **at this point**




**If we make a step of length $|\mathrm{grad}\ F(x^{(0)})|$ 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:**

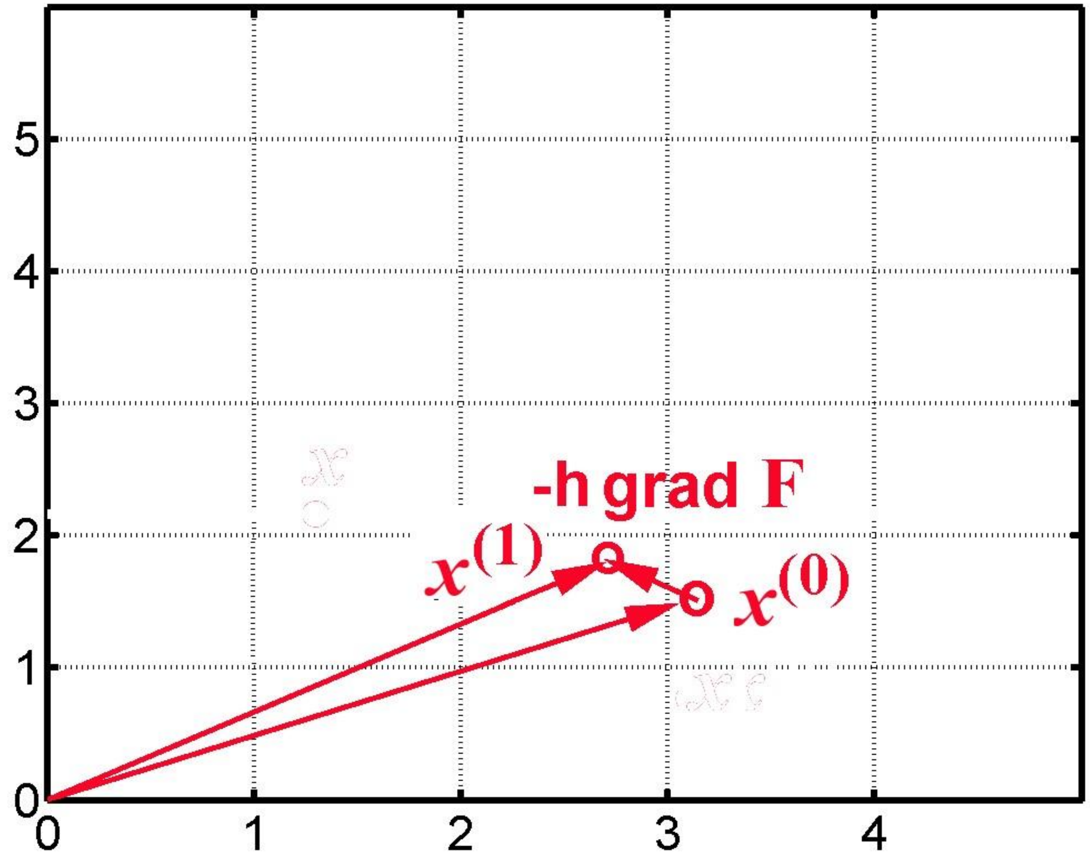



$$x^{(1)} = x^{(0)} - h\cdot\text{grad } F(x^{(0)})$$ - in vector form.

**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:

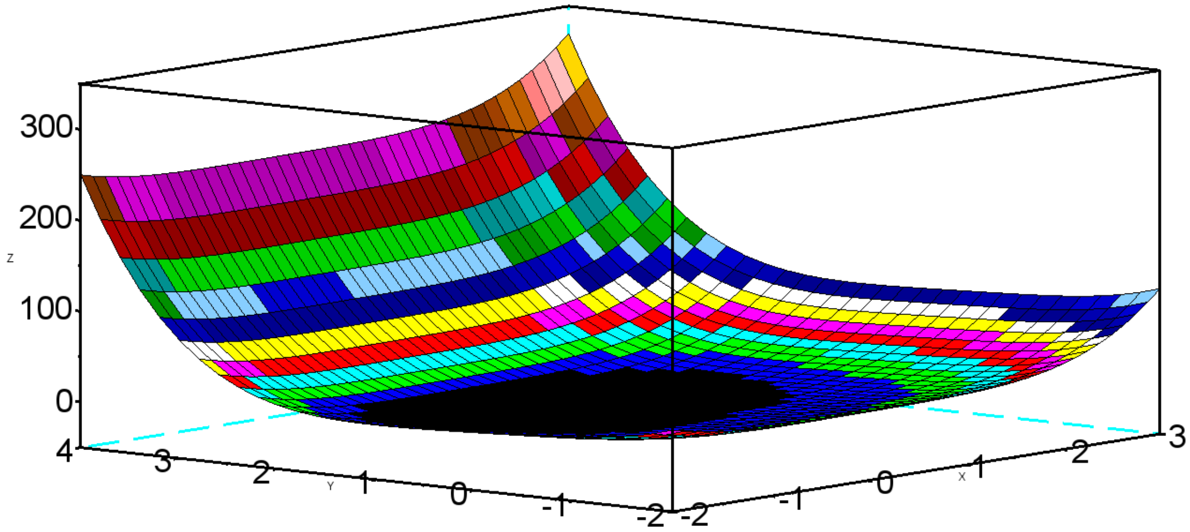
As soon as we approach a minimum, the derivatives $\partial F(x^{(k)})/\partial x_i$ decrease; therefore, the difference $x^{(k+1)} - x^{(k)}$ decreases as well.

Example 1. Find a minimum

z=F(x,y)= x^4 + y^4 - 5y +x          -2<x<3,   -2<y<4

Scilab :

```
for i=1:41
for j=1:31
   x(i,j)=0.125*(i-1)-2
   y(i,j)=0.2*(j-1)-2 end
end
F=x.^4+y.^4 -5*y+x
surf(x,y,F)
```



```
h=0.01
xx=2.5
yy=3.5
plot(4,5)
for k=1:300
dFdx=4*xx^3 +1
dFdy= 4*yy^3-5
xx=xx-h*dFdx
yy=yy-h*dFdy
plot(xx, yy,'or','LineWidth',2)
end
disp('xx=',xx,'yy=',yy,'k=',k)
```

answer: xx= -0.6299189
yy= 1.0772173

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

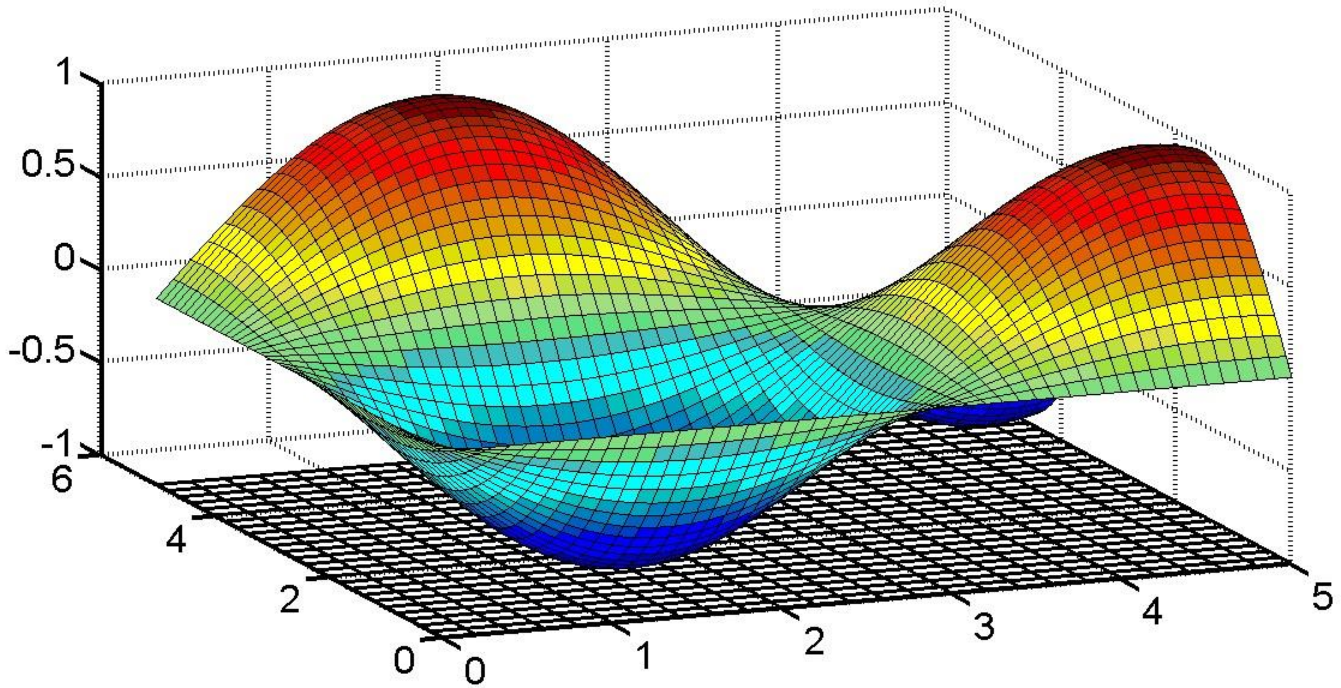
$$\left| x_i^{(k+1)} - x_i^{(k)} \right| < \varepsilon \qquad \textbf{for all} \quad i$$

**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

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

# Recall Method of gradient descent:

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

**Calculate**

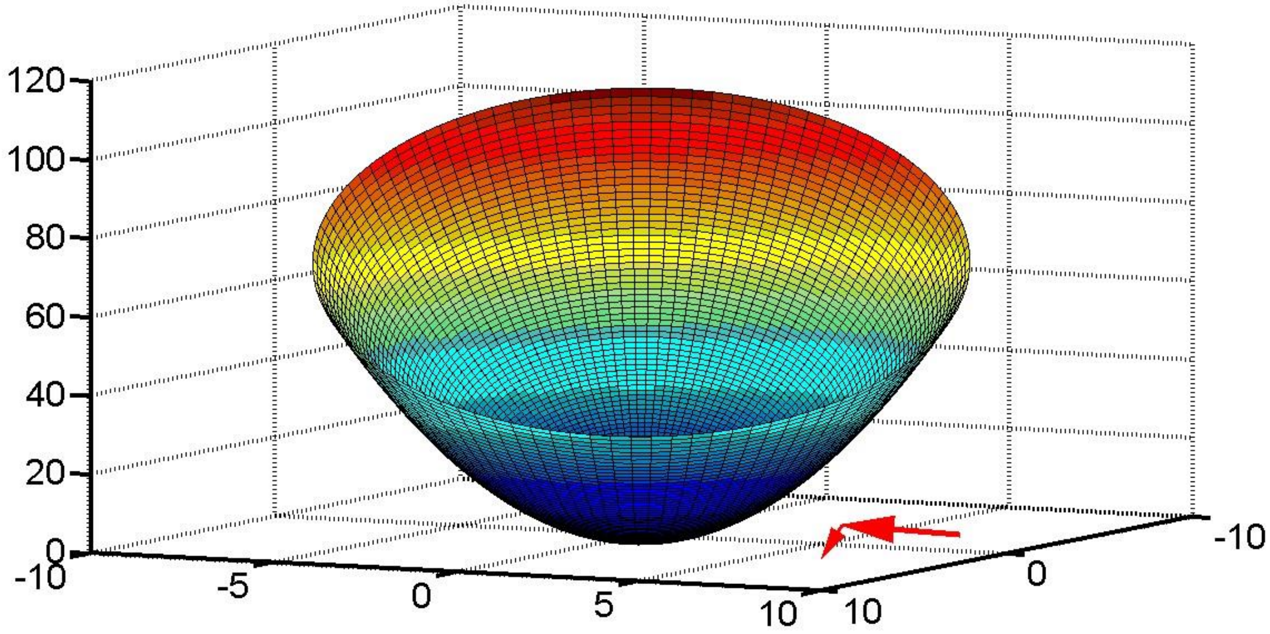
$$\text{grad } F(x^{(0)}) = \sum_i \bar{e}_i \, \partial F(x^{(0)})/\partial x_i$$

**Sequential approximations:**

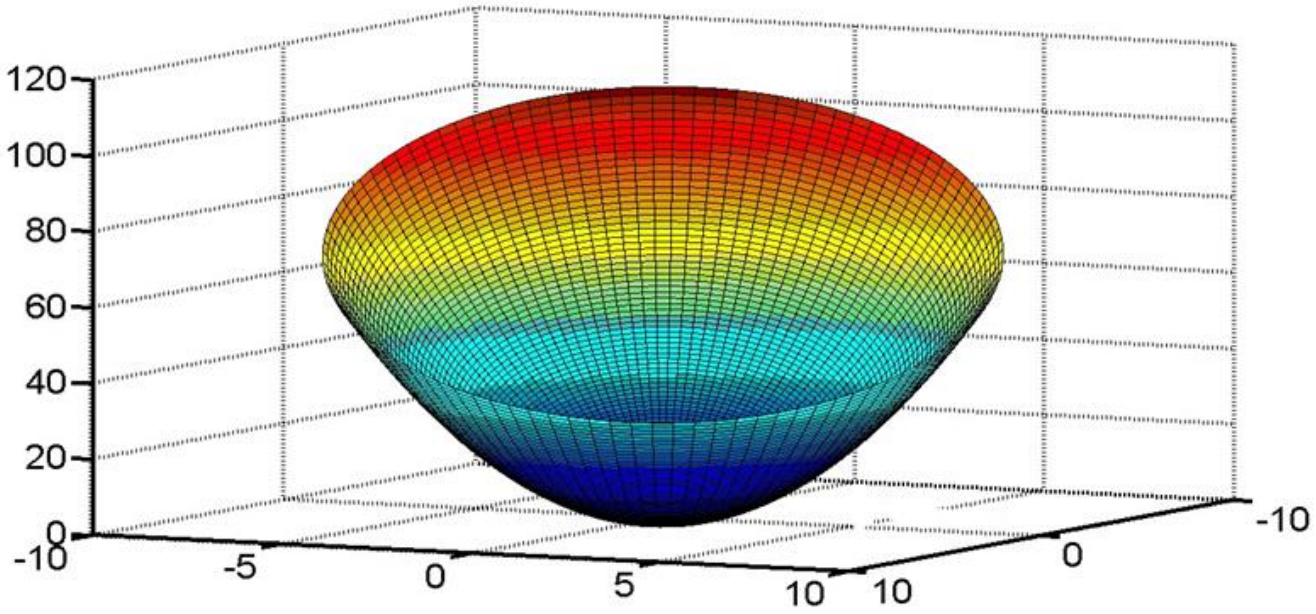
$$x_i^{(k+1)} = x_i^{(k)} - h \cdot \partial F(x^{(k)})/\partial x_i$$

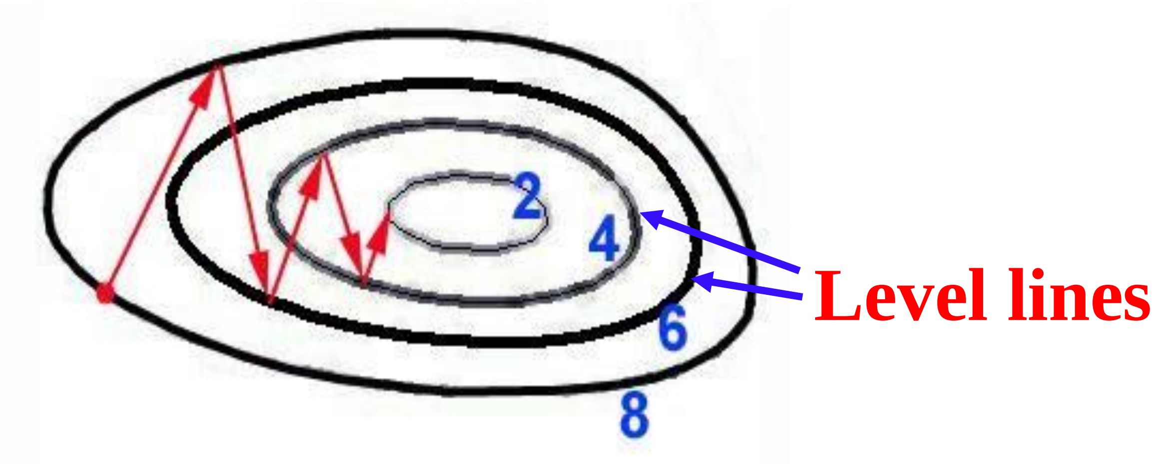
**The parameter *h* governs the length of vector.**

**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.**



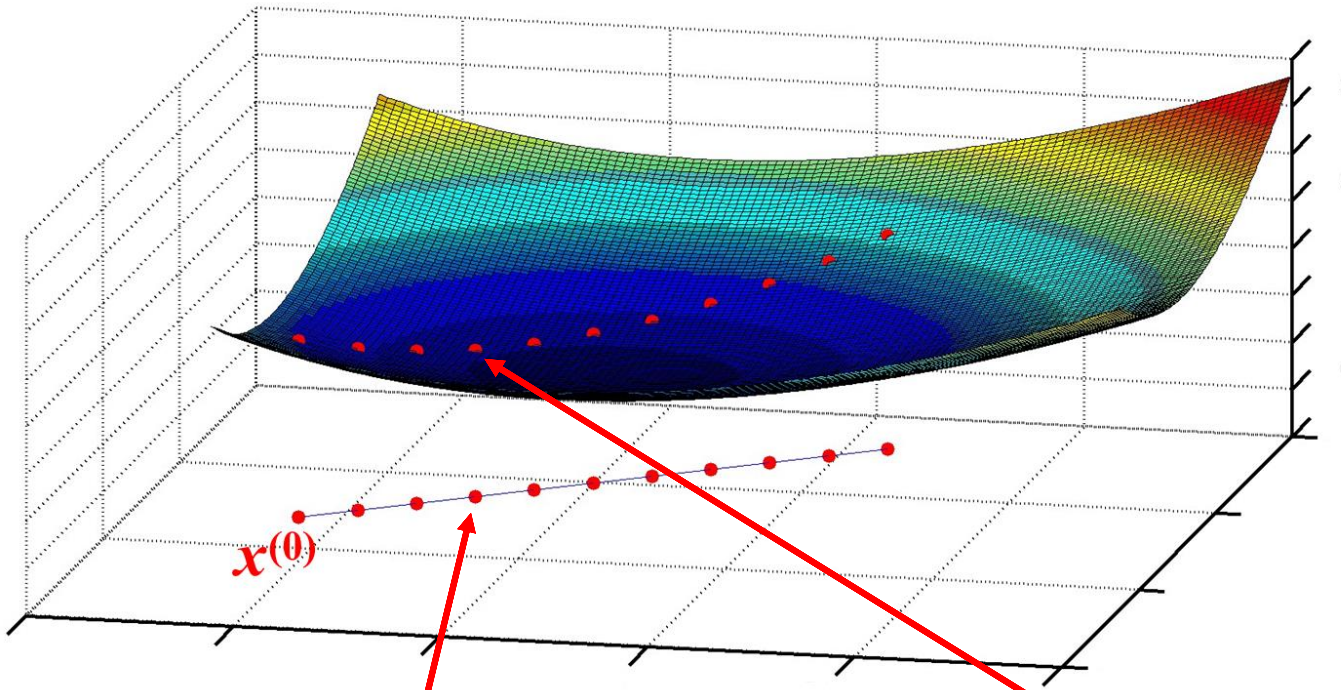

If $h$ is too large, then sequence $x^{(k)}$ may converge <u>slow</u> as well, as it may pass by a minimum:




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

**What is the optimal value of $h$ ?**

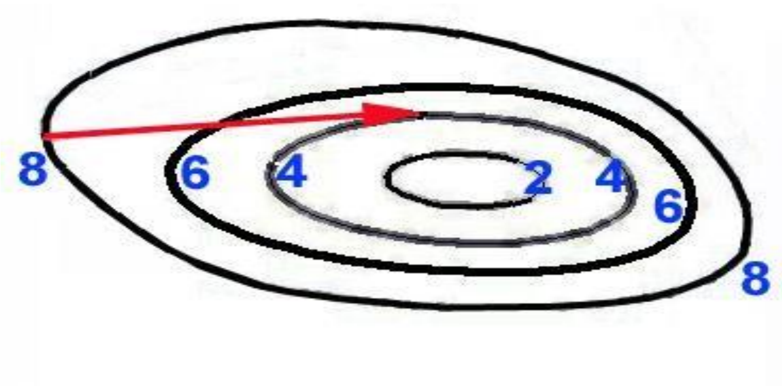



Consider various $h_1, h_2, h_3, \ldots$

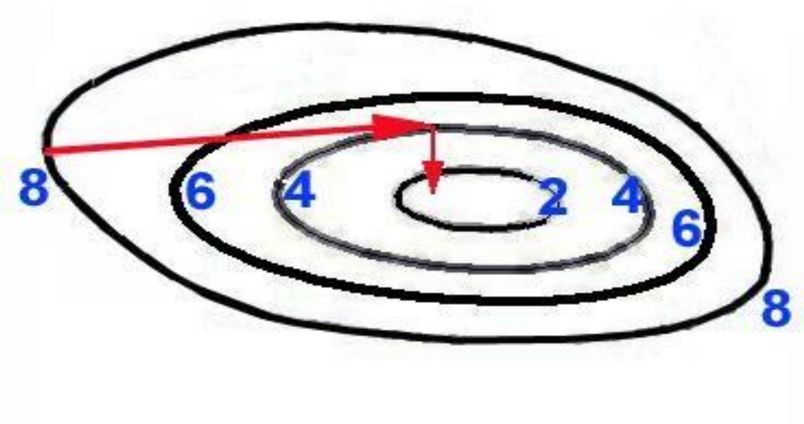
Suppose we have found such $h_{opt}$, that $\mathbf{F}$ attains a minimum $\mathbf{F}_{min}$

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.

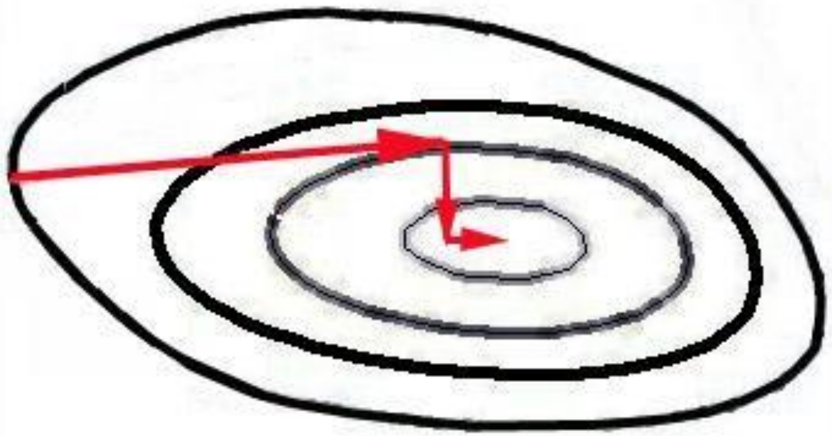
As soon as we found such optimum $h_{opt}$, that ensures a minimum of F, we move from $x_i^{(0)}$ to $x_i^{(1)} = x_i^{(0)} - h_{opt} \cdot \partial F(x^{(0)})/\partial x_i$




**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 $x^{(1)}$ .**

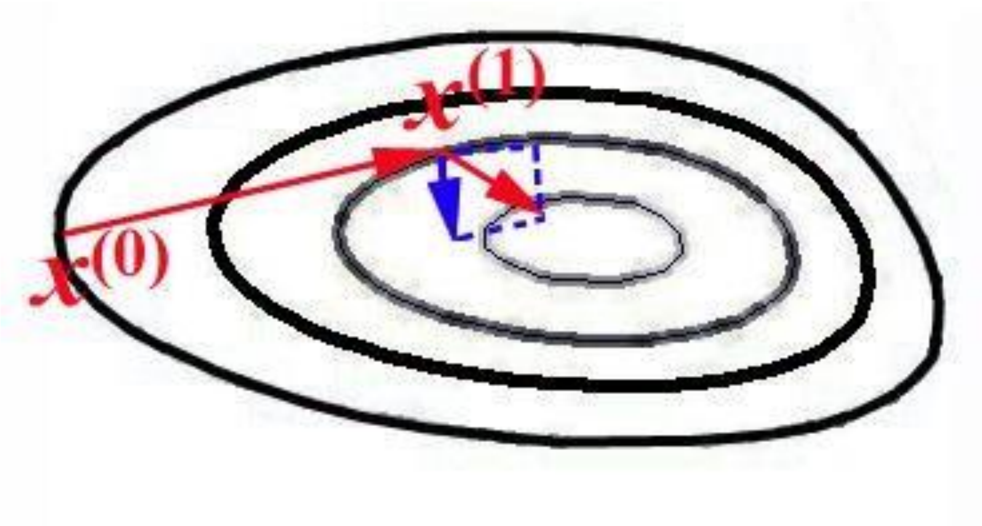


**In the same way, at next steps, we seek** $\boldsymbol{h^{(k)}_{opt}}$ and move on:

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

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:**

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

## Conjugate gradient method:

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

**correction of the direction of step**

Scilab built-in command: fminsearch

```
function y=HIT(x)
  y = x(2)^2 + (1-x(1))^2;
endfunction
[x, fval] = fminsearch ( HIT, [1 -1] )
```

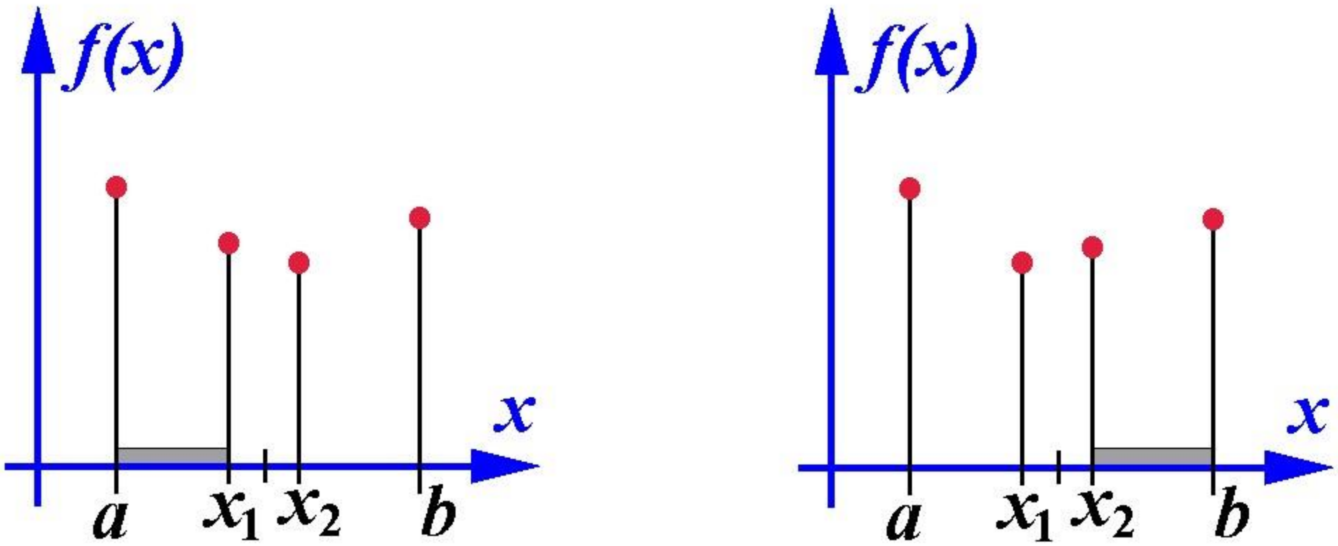
**Computes the unconstrained minimum of given function HIT with the Nelder-Mead algorithm.**

**Same command "fminsearch" in Matlab. In addition, "**`gradient`**"**

```
[FX,FY] = gradient(F)
```

returns the *x* and *y* components of the two-dimensional numerical gradient of matrix `F`.

# 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

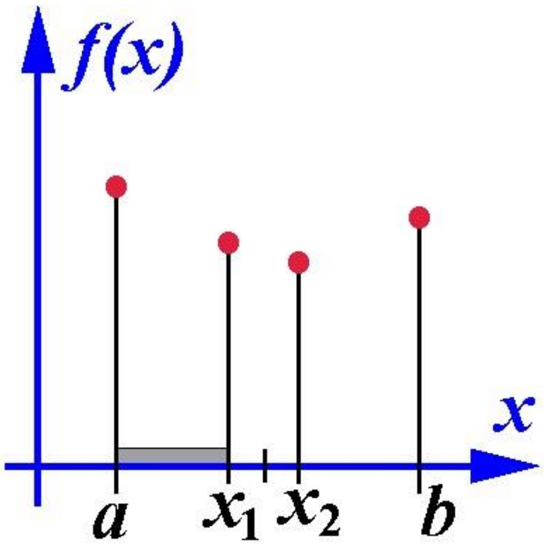
$$x_1 = 0.5\,(a+b) - \delta\,, \qquad\qquad x_2 = 0.5\,(a+b) + \delta\,,$$

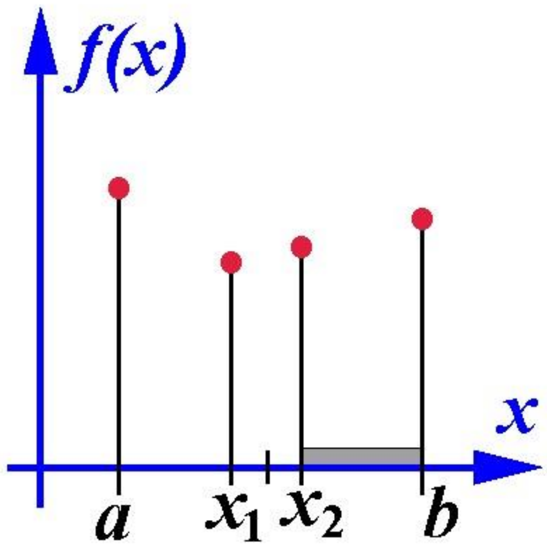
where $\delta$ is small.

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

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

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



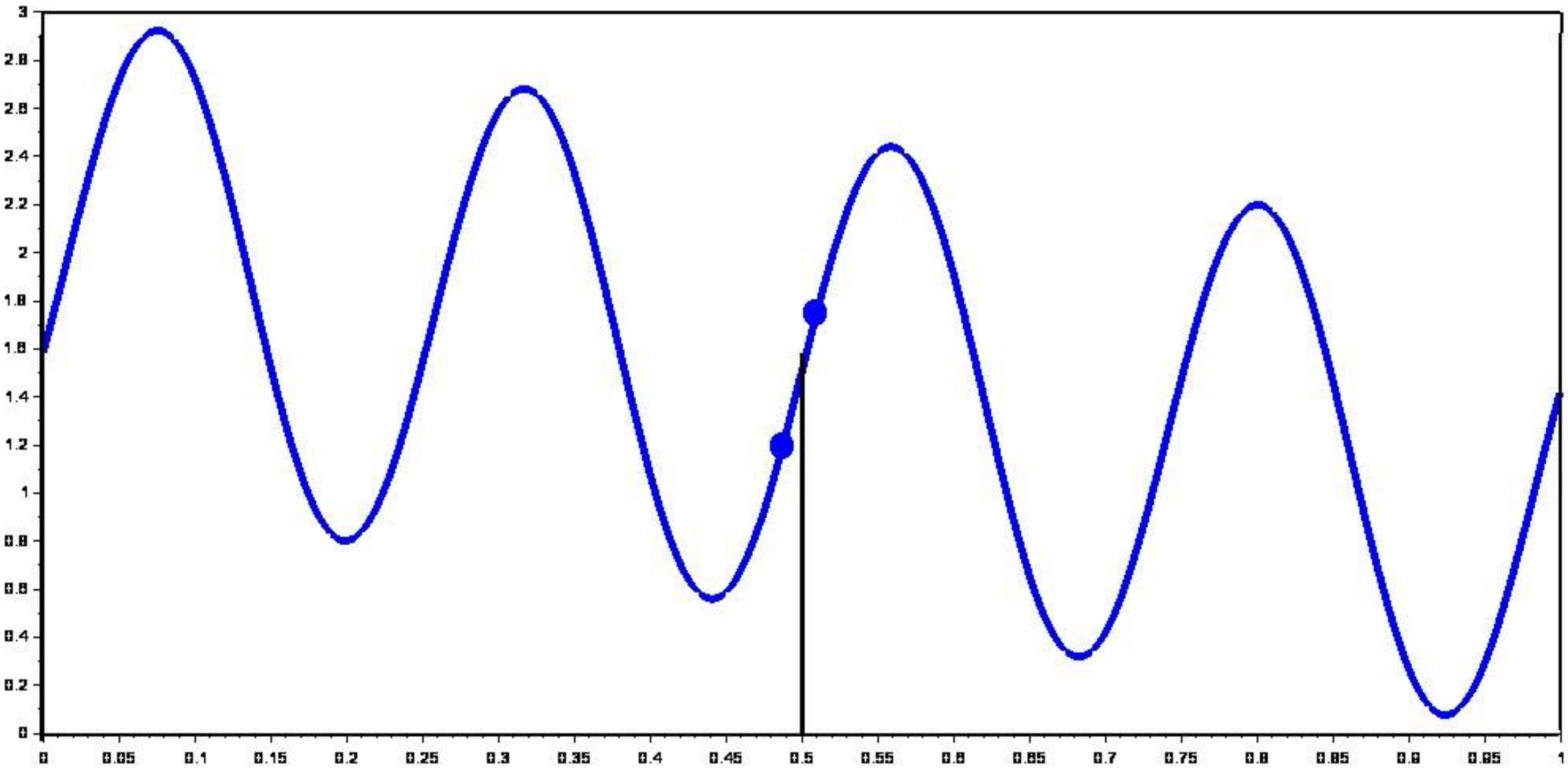




**For the obtained segment $[a, x_2]$ or $[x_1, b]$ the procedure of halving the segment keeps on until the length of segment becomes small enough.**

```
x=0:0.0002:1
y=x.*exp(x)+50*x.*sin(2*x).^2 -30*x
      plot(x,y)
      xgrid
 x1=0
 x2=1
 delta=0.0000005
 i=1
 while x2-x1> .000002
     x3=0.5*(x1+x2)-delta  ;
     x4=0.5*(x1+x2)+delta ;
 i=i+1 ;
 y3= x3*(exp(x3)+50*sin(2*x3)^2 -30)  ;
 y4= x4*(exp(x4)+50*sin(2*x4)^2 -30) ;
 if y3<y4 then       x2=x4   ;
 else x1=x3  ;
```

```
    end
disp( y3,i)
end
xmin=0.5*(x3+x4)
ymin= xmin*(exp(xmin)+50*sin(2*xmin)^2 -30)
disp("ymin=",ymin,"xmin=", xmin)
```

**It is recommended to plot a graph of the function $f(x)$ in order to avoid a loss of a minimum.**



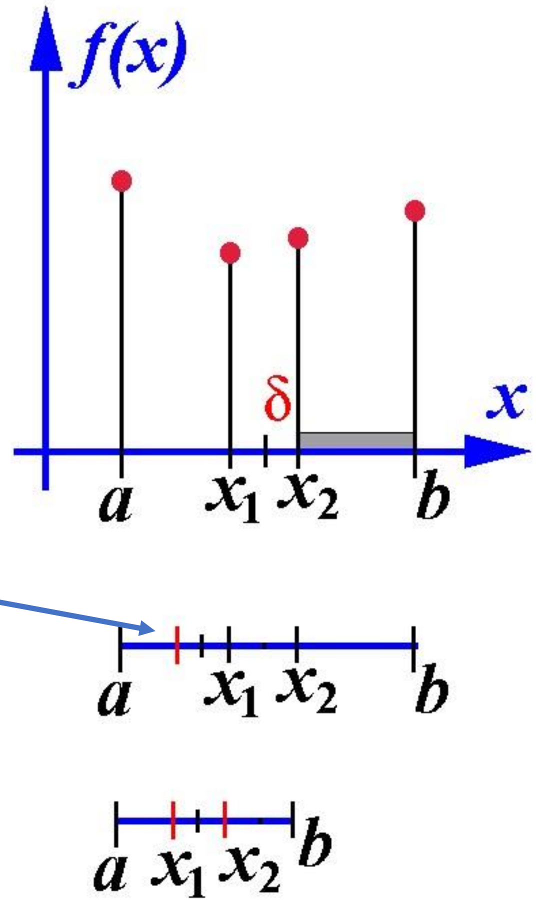
3
2.8
2.6
2.4
2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0 0.05 0.1 0.15 0.2 0.25 0.3 0.35 0.4 0.45 0.5 0.55 0.6 0.65 0.7 0.75 0.8 0.85 0.9 0.95 1

# 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 <u>middle</u> 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**