OPTIMIZATION ALGORITHM

Sebastien Charnoz & Andrea Ciardi

https://sites.google.com/view/paris-physics-master/first-semester-courses/numerical-methods-for-physics

Paris Physics Master

Many problems need to minimize a function or find an optimal solution;

-Minimizing the distance (XHI2) between data points and a model

- Find the equilibrium state of a mechanical system (Minimize E_{pot})

- Find the equilibrium state of a gas, a chemical composittion (Maximize Entropy)

- All optimization problems :minimize cost in engineering, minimize time …

- Data Processing : machine learning, image processing ….

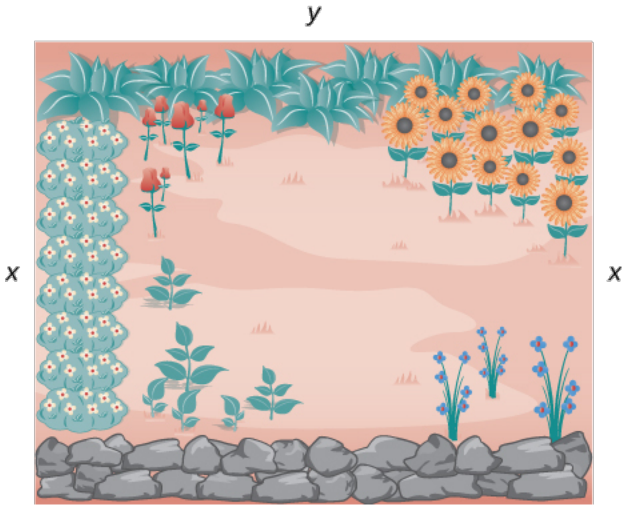
All these problems come in a wide category: optimization

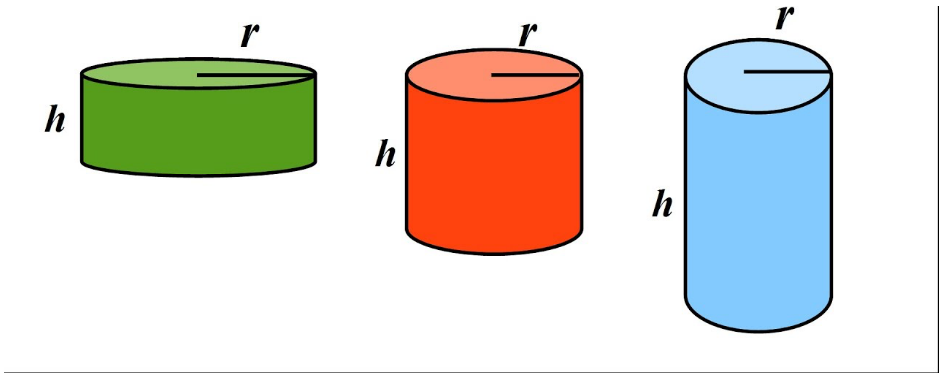
Note: Maximize F (x) = Minimize -F (x)
Minimize and maximize are actually the same problem !

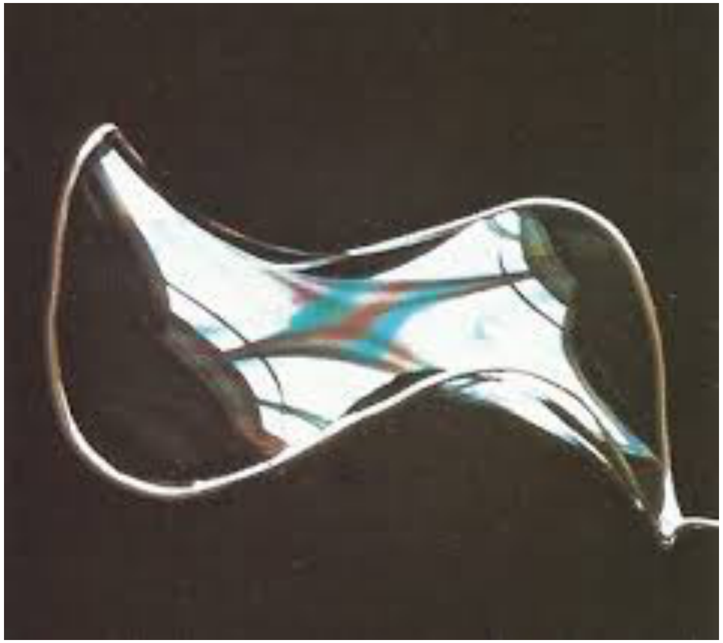
Examples :

Finding the optimal size of a rectangular garden with a given perimeter

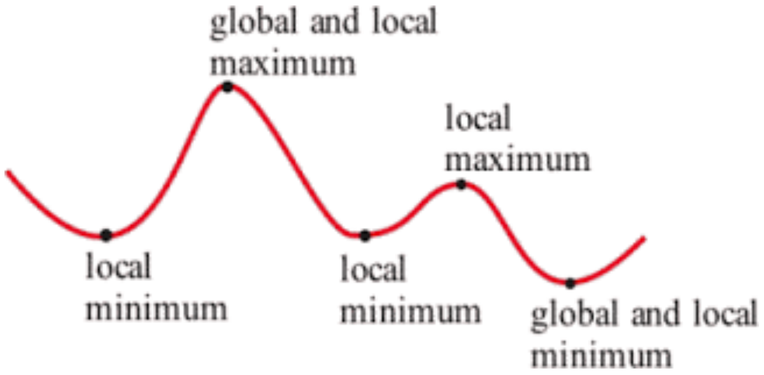
Finding the cylinder with the smallest surface , with constant volume to Minimize cost.

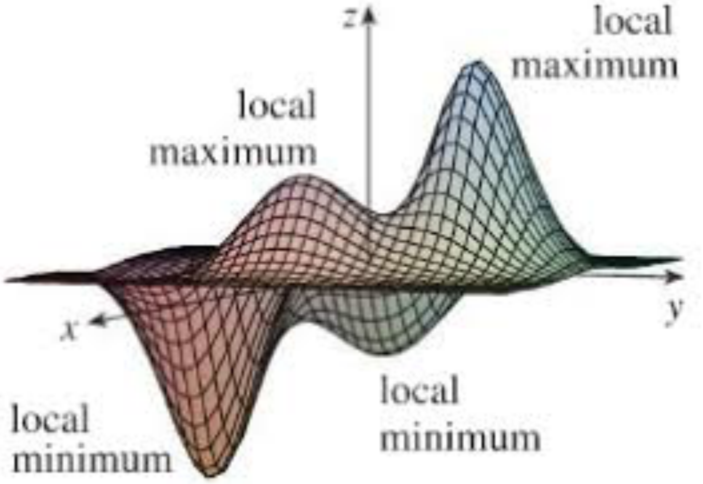




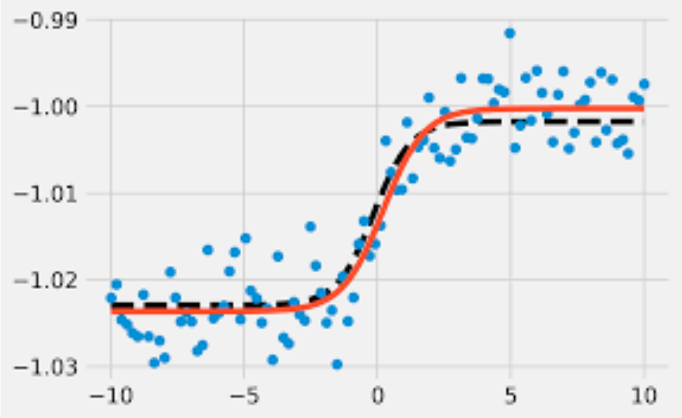


Find the minimum point of a given function with complex shape





Find the best model that fits some data points.

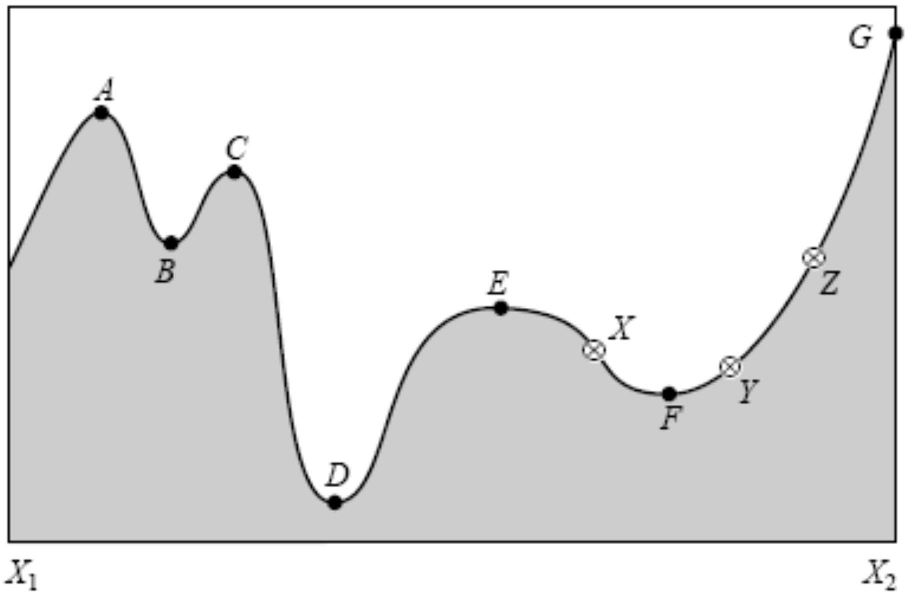


Once again, it is an ill-posed problem:

There is NO ideal method.

It is quite easy (i.e. fast) to find a local minimum.

It is very difficult (i.e. long) to find a global minimum



B, D, F are local minima

D is the global minimum

This research is particularly long and difficult if the problem is N-dimensional (N>2)

Let n, m, and p be positive integers. Let X be a subset of R^n, let f, g_i, and h_j be real-valued functions on X for each i in $\{1, \ldots, m\}$ and each j in $\{1, \ldots, p\}$, with at le
A nonlinear minimization problem is an optimization problem of the form

There are different categories of problem :

Unconstrained optimization :

We try to find the minimum of a function F (or of a vector of functions) for some variable (x) with NO constrain on x (basic case)

Constrained optimizaion :

We try to find th minimum of a function F WITH constrained on the variable X.

Linear optimization :
Minimize C^T ***X****, Subject to A* ***X*** *<b,* ***x*** *>0 (the « garden » problem)*

Non linear optimization :
(Cylinder problem)

$$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g_i(x) \leq 0 \text{ for each } i \in \{1, \ldots, m\} \\ &\qquad\qquad h_j(x) = 0 \text{ for each } j \in \{1, \ldots, p\} \\ &\qquad\qquad x \in X. \end{aligned}$$

Chemical reaction at equilibrium:
To find the composition at equilibrium the chemical potential must be minimized of the system but must be respected that the total mass of the system is preserved ...

There are also specific methods for these cases.

Most method are first built for 1D problem (x is a scalar)

For Multi-dimensional problems, numerous methods try extend the 1D case.

Note :
All methods start from an « initial guess » and then try to find the closest minimum => IMPORTANCE OF THE FIRST GUESS.

For REAL TOUGH PROBLEMS

Multidimensional optimization, global minimum finding, data mining

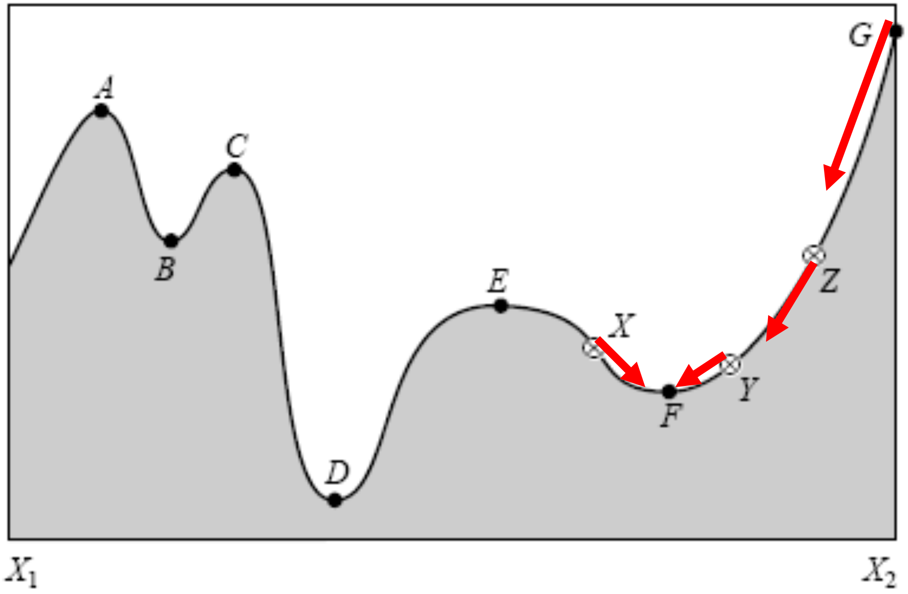
New families of algorithm :
Genetic Algorithm, simulated annealing, machine learning..

1D unconstrained mimization

The most classic method is the famous gradient descent:

We want to find A minimum of F(x), starting from a point X_0 given by the user

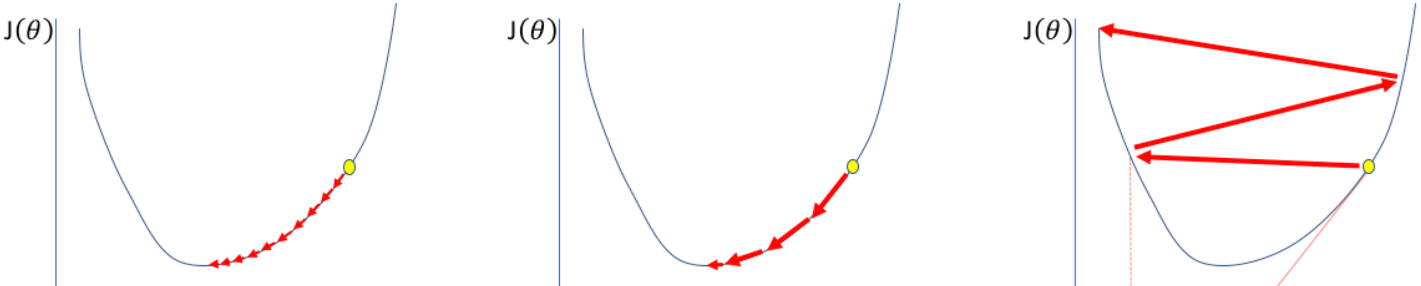
We **explore** F(x) always « Downslope » => in the direction of the local gradient.



G=starting point
F =minimum found (closest)

This is why most algorithms find
of the *local minima*
not the *absolute minimum*

A big part of the computation
Time and difficulty is
is to compute the derivative H '(x)



IMPORTANCE OF THE FIRST AND SECOND DERIVATIVES F'(X) AND F''(X)

Reminder : if x is a minimum of F(x) then F'(x)=0

So :If you KNOW F'(X) (gradient) this will make your life MUCH EASIER

If you use a minimization algorithm

IT MEANS you CANNOT analytically calculate the zeros of F '(X)

-Either because you do not know F '(x)

-Either because solve F '(X) = 0 is too complicated.

- Or because F(X) is a complex function.

BUT IF YOU KNOW F'(X) THIS WILL BE VERY USEFUL AND WILL SAVE A LOT OF TIME

MINIMIZE WITH DOWN GRADIENT known derivative

We know F and we assume we know F' (X)

Starting from a starting point X_0 ('first guess'), we compute a serie of points as follows :

$$X_{k+1} = X_k + d_k g_k$$

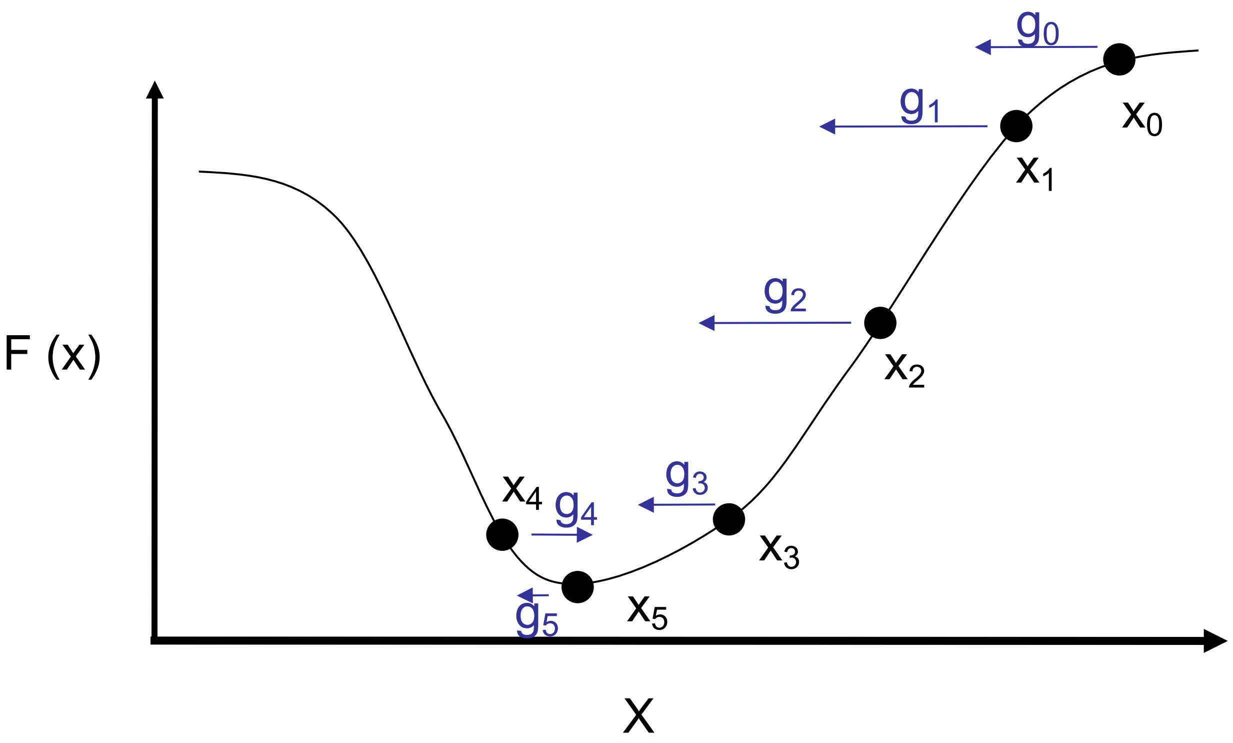
where g_k is the direction of descent (« left or right » in 1D) an d_k is the length of The step

The choice of g_k is such that f (X_{k+1}) <F (X_k). A natural choice is :

$$g_k = -\left.\frac{df}{dx}\right|_{X_k}$$

In the simplest method of descent gradient, g is simply « minus the gradient of F »

Note : here g_k is not normalized to 1



Example 1D

g_i is less gradient of point i

$$X_{k+1} = X_k + d_k g_k$$

2 questions:

- How to choose the step-size d_k ? (it must be smart)

- When does the calculation stops .

The « Fixed » descent

If one does not know the second derivative descend the fixed pitch gradient. we impose d_k= cst **at the beginning of the calculation.**

What value do to d? **Hard to say a priori. In general it takes**
d <the characteristic size of the research area**.**
For example the width of the valley.

The descent fixed not only on if the function is soft, and if the point Start is close to the solution.

The test will stop when

|| $f '(x_k)$ || <epsilon

or $X ||_k-X_{k+1}|| / || X_k||$ <epsilon

Warning: "fixed pitch" is an abusive name.
The REAL no progress is: dxg_k where g_k is -F '(X)

This is the multiplicative factor that is constant.

Choice of the Step Size d$_k$

requires knowing the second derivative !! :
One makes a locally parabolic approximation of the function

It is based on the Taylor Expansion (T.E.) of F (x) at point X$_k$

$$f(X_k + dx_k) = f(X_k) + \left.\frac{df}{dX}\right|_{x_k} dx_k + \frac{1}{2}\left.\frac{d^2 f}{dX^2}\right|_{x_k} dx_k^2 + o(d_k^2)$$

$$\Rightarrow$$

$$f'(X_k + dx_k) = f'(X_k) + \left.\frac{d^2 f}{dX^2}\right|_{x_k} dx_k$$

T.E. of the derivative.

We want to cancel the derivative So, we choose to dx$_k$ so that F '(X$_k$+ dx$_k$) = 0

$$dx_k = \frac{-f'(X_k)}{\left(\frac{\partial^2 f}{\partial x^2}\right)_{X_k}}$$

Now with our notation:

$$X_{k+1} = X_k + d_k g_k$$

And g_k= -F (X_k) from where the "no" of progress$_k$ dx =$_k$/ F (X_k)

$$d_k = \frac{1}{\left(\frac{d^2 f}{dx^2}\right)_{X_k}}$$

This method is quadratic.

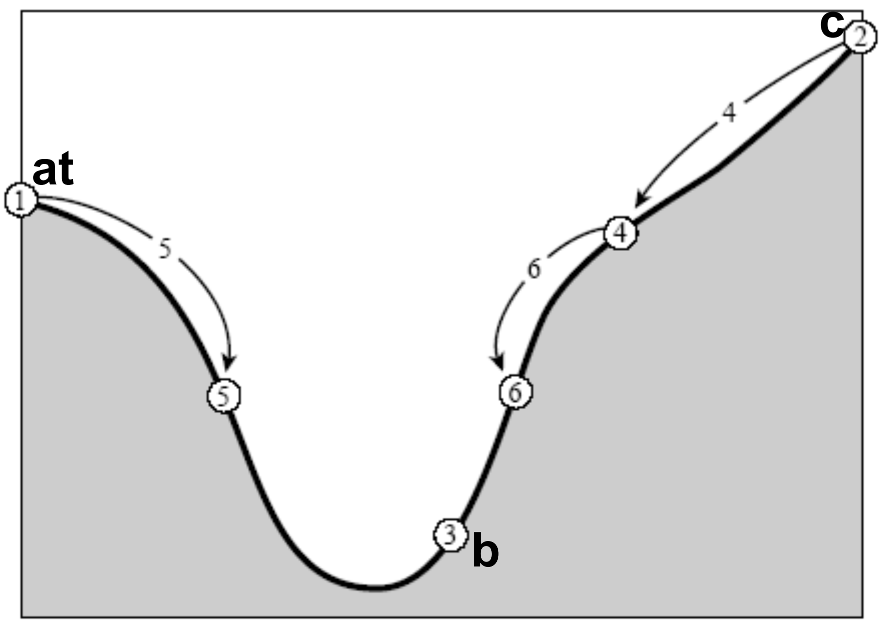
The advantage of this method is that it generalizes well N-dimensional problems

But Often it can be difficult to compute F’(X) …and F’’(X) !!

Example : the XHI2 test :

MINIMIZATION 1D WITHOUT GRADIENT (Bisection)

This is the easiest method, very similar to the method of bisection to find a zero



Basic Method

General method

Framing the minimum of three points:
a <b <c

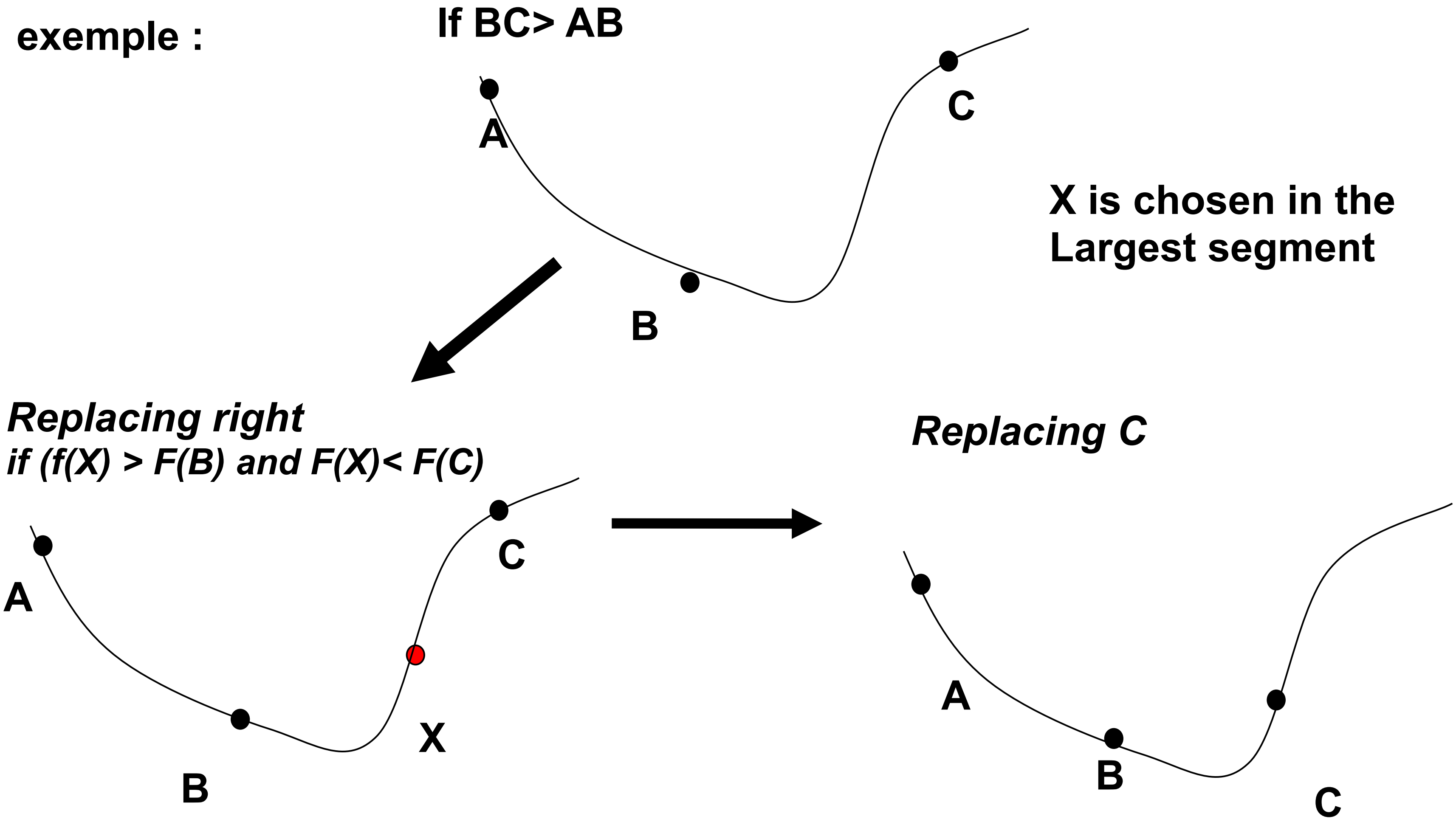
such that f (b) <f (a)
f (b) <f (c)

The$_{ab}$= ba and L$_{bc}$cb =

1. We start from a and b and c
2. calculates f (a), f (b), f (c)
3. If L$_{ab}$> L$_{bc}$ We take X = medium (a, b)
If f (x) <f (b) then c = b and b = x
If a = x, b and c are kept
5. If L$_{bc}$> L$_{ab}$ We take X = medium (b, c)
If f (x) <f (b) then a = b and b = x
If x = c, a and b are kept

6. Returns 2

exemple :



If BC> AB

X is chosen in the Largest segment

Replacing right
if (f(X) > F(B) and F(X)< F(C)

Replacing C

At the end of the stage:

F (b) <F (a) and F (b) <F (c)

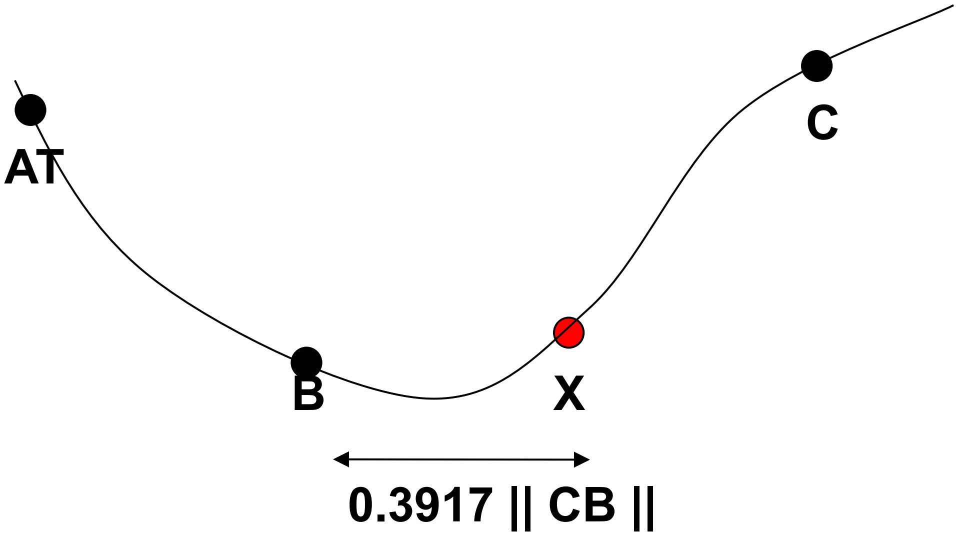
Method of the "Golden rule" to find X

In fact it can be shown that there is an optimal way to choose again point X in a single step.

the larger of the two right left segments are determined:
|| AB || and CB || || . Let D be the segment length

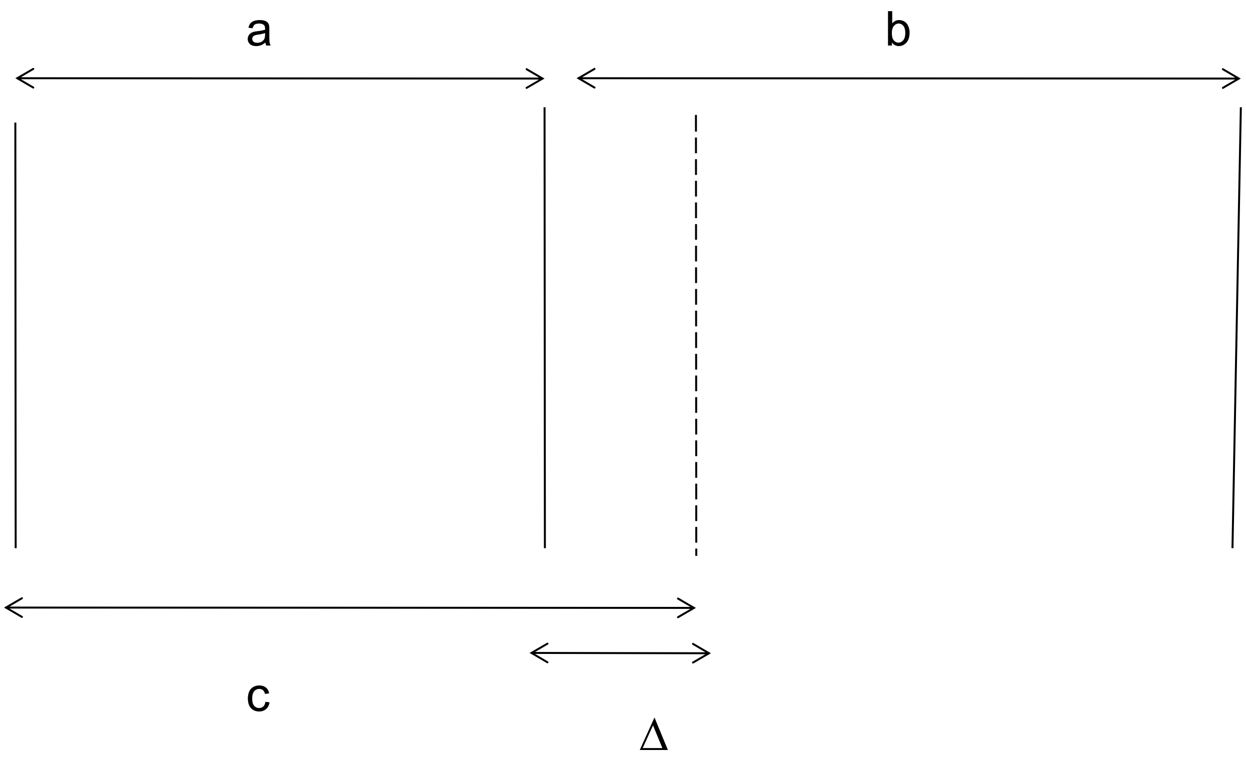
And we take the new point X in this segment in the distance:
0.39197 x D (golden ratio) the point B

If F (x) <F (B) then B = X or if F (x) <F (c) then C = X



So that (aX) / (C) = 1.618

The Golden ratio



How to choose a and b ?

Keep the proportions for each iteration:
To the left $\Delta / A = a / b$ and to the right $\Delta / (B \Delta) = A / b$

is eliminated Δ and include: $a / b \sim 1.618$ .. The golden number

However, whereas you are sur not to spend too much time fo find the minimum,
This is not ALWAYS faster than the simple bissection

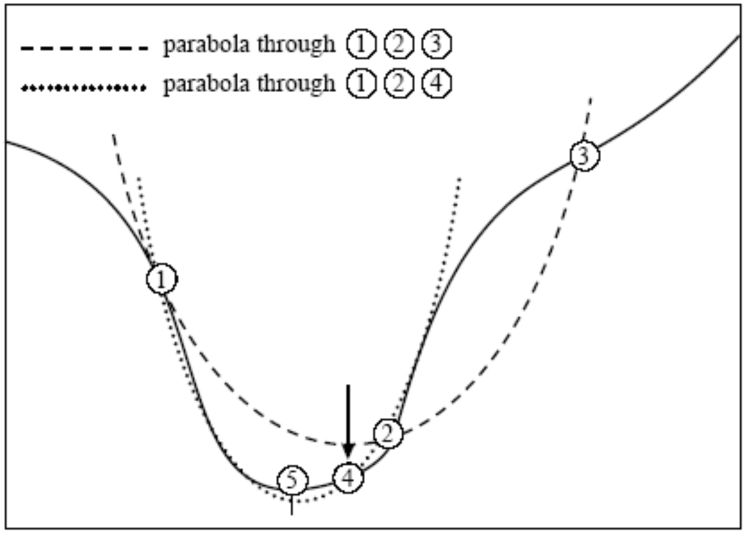
parabolic method for finding X

Now the rule of we can do better if F is nicely parabolic meadows the minimum.
If F is *effectively* parabolic on the interval [a, c], then we can analytically calculate the minimum

$$x = b - \frac{1}{2}\frac{(b-a)^2[f(b)-f(c)]-(b-c)^2[f(b)-f(a)]}{(b-a)[f(b)-f(c)]-(b-c)[f(b)-f(a)]}$$

Note: only works if F is not linear

In practice, F is rarely *exactly* parabolic. However, one can use
It calculates X to find the new minimum supervision



For the next step

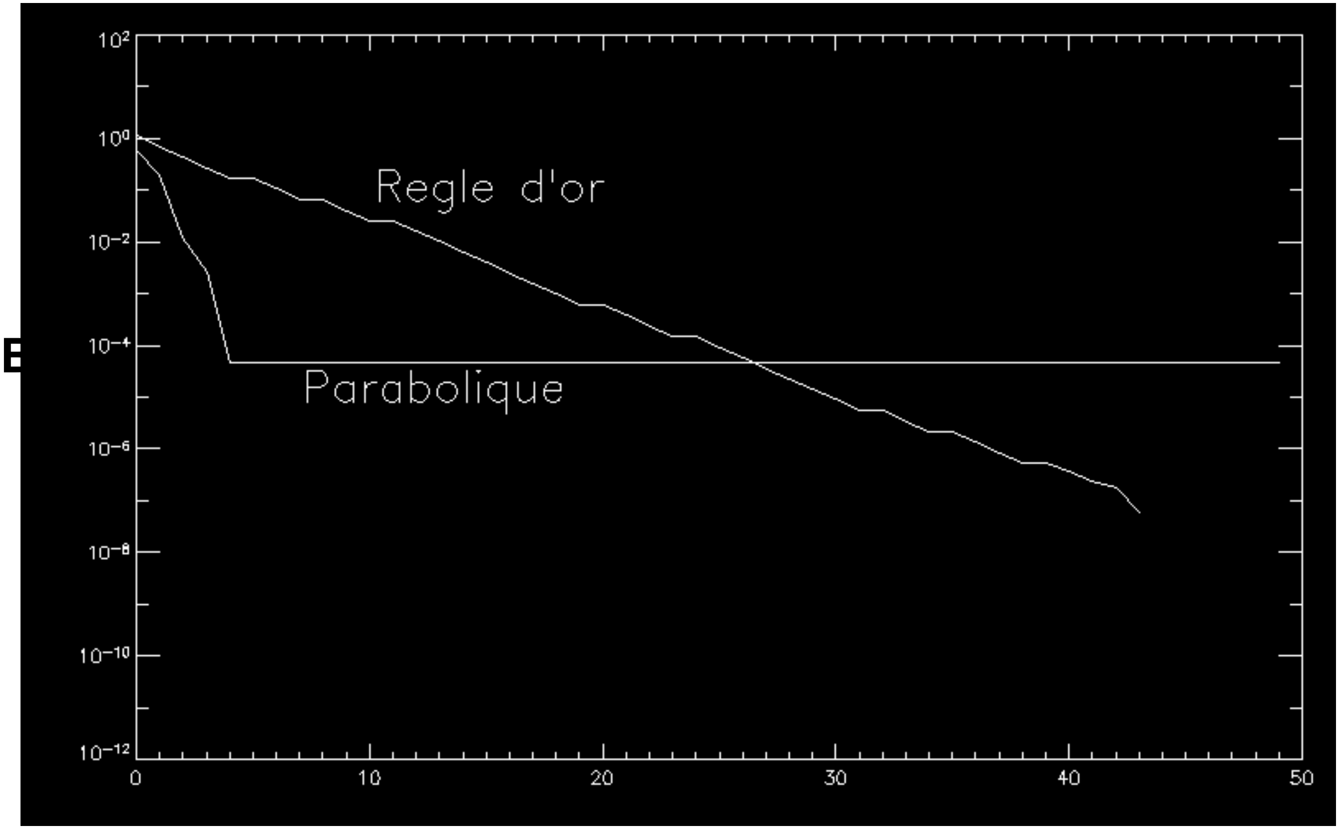
Yes x <b then ABC = [a, x, b] (left)

If x> b then ABC = [x, b, c] (right)

Comparison of two methods: minimization of Sin (x) ^ 2

Fault:

|| || AC / E



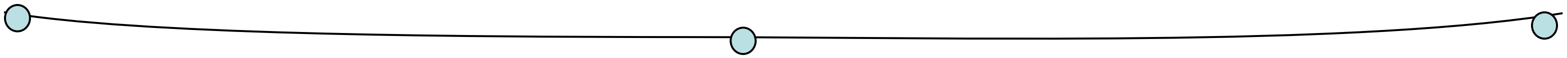
N iterations

We see that the parabolic method converges much faster that the method of the Golden Rule.

However its convergence is limited (> 10^{-5}) .... Why ?

When the function is "flat" parabolic method becomes unstable

In practice what happens when the function is locally "flat"
An error digital precision can "go out" on the miminum interval



minimization Multi-Dimensional

Minimize F (X) wherein X = (x_1.... x_{not})

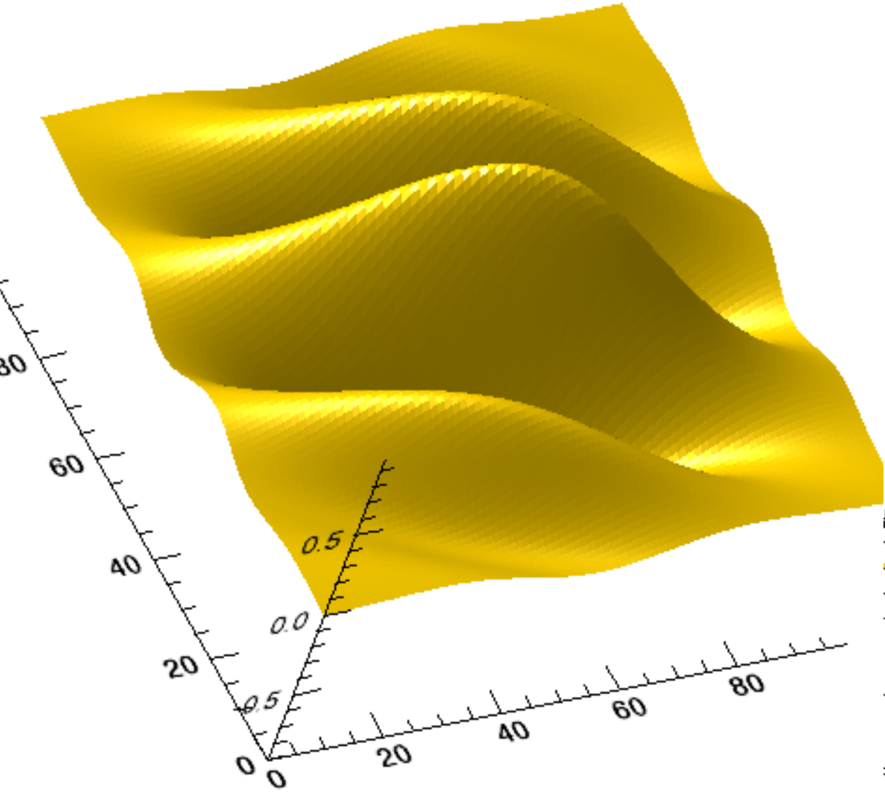
The problem is much more twisted ... and much longer to solve

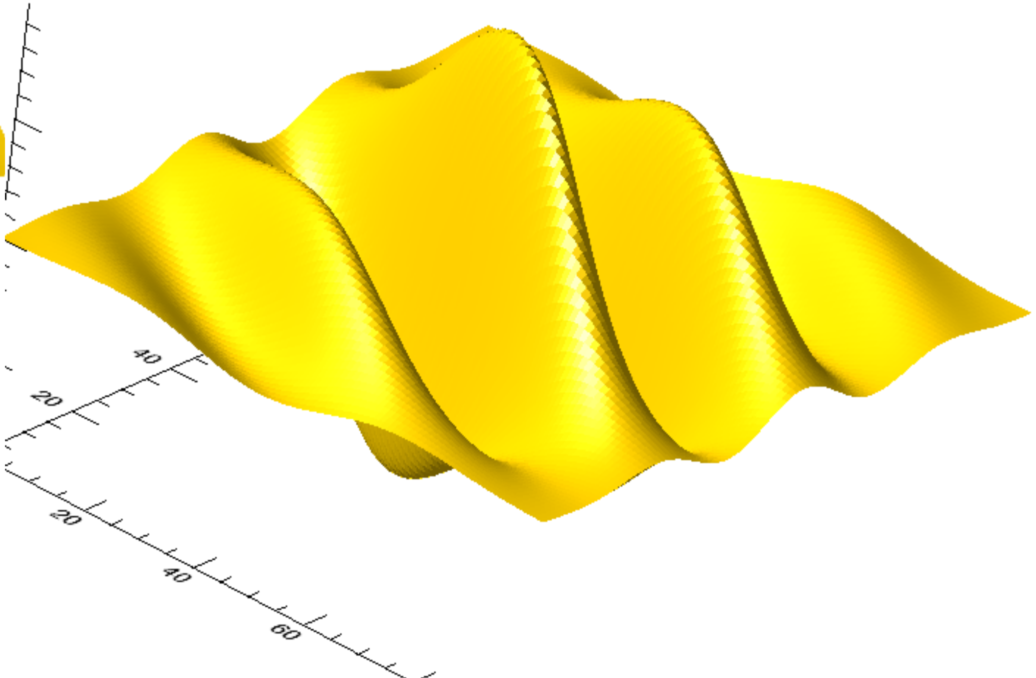
The big problem is often the CHOICE of the descent direction

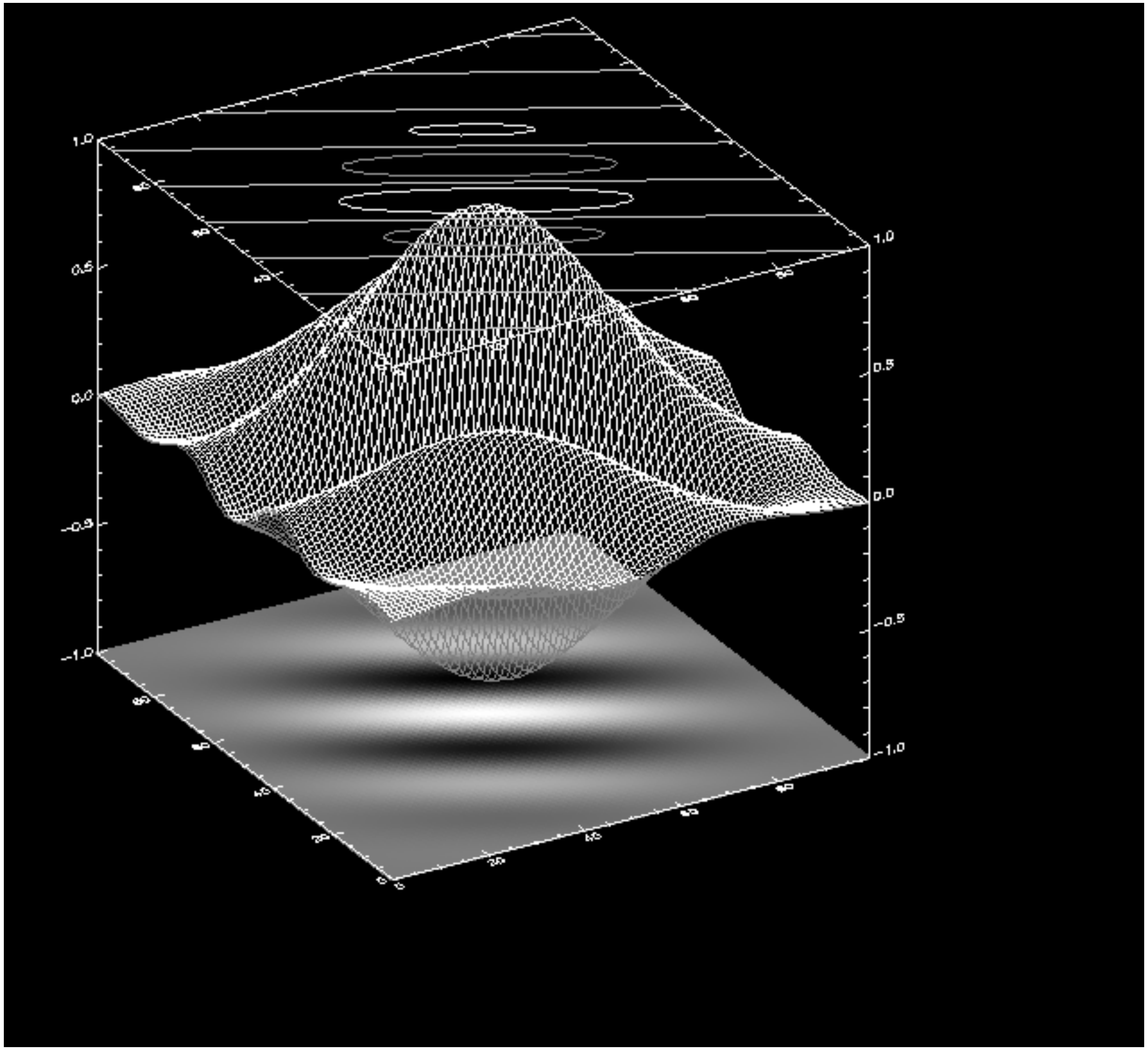
Once we know which direction we go down, the other problem is how the *length* the descent.

In N dimensions, these choices are crucial.

sample function to minimize, in 2D only F (x, y)







Here: several minimum,

Existence of "valleys" narrow,

Maximums and local minimums
etc ...

Several techniques to tackle such a problem. Any is universally effective.

ALL require a starting point chosen by the user X_0
=> Need from a point X_0 not very far from the minimum

The greater part use the DL function of several variables (N variables)
X = P + [x_1, ..., x_{not}] where P is a vector

$$f(\mathbf{x}) = f(\mathbf{P}) + \sum_i \frac{\partial f}{\partial x_i} x_i + \frac{1}{2} \sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j + \cdots$$
$$\approx c - \mathbf{b} \cdot \mathbf{x} + \frac{1}{2} \mathbf{x} \cdot \mathbf{A} \cdot \mathbf{x}$$

$$c \equiv f(\mathbf{P}) \qquad \mathbf{b} \equiv -\nabla f|_{\mathbf{P}} \qquad [\mathbf{A}]_{ij} \equiv \left. \frac{\partial^2 f}{\partial x_i \partial x_j} \right|_{\mathbf{P}}$$

b is the gradient
A is called the "Hessian" f (x). It is simply the derivative matrix seconds at P.

Example Hessian: Here the function is f (x, y, z)

Hessian: A =

$$\left.\begin{bmatrix} \frac{\partial^2 f}{\partial x \partial x} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y \partial y} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z \partial z} \end{bmatrix}\right|_P$$

Calculated at the point P

While the term:

$$\sum_{i,j} \frac{\partial^2 f}{\partial x_i \partial x_j} x_i x_j = \vec{X} \cdot \left(A \vec{X} \right)$$

Is "Scalar Product X **with** AX **"**

Methods CLASSIC down the slopes.
They fall into two main categories:

Methods that use several points: no gradient or Hessian

*** Simplex Method (also called "Amoeba", "Amoeba" in English)**

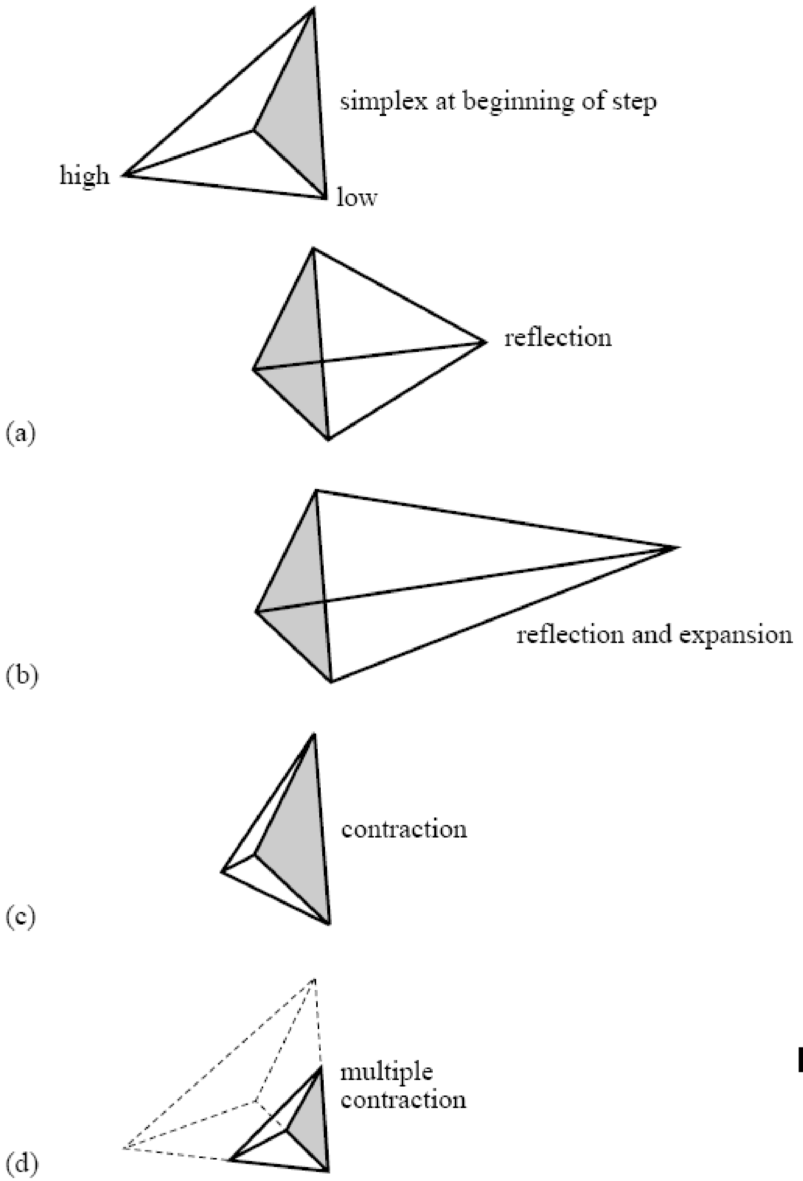
Mixed methods (multiple 1D minimizations):
*** Relaxation**

Methods using the gradient and the Hessian (if known)

- **fixed gradient descent (Gradient)**
- **optimal gradient descent (Gradient Hessian +)**
- **or conjugate gradient(Gradient Hessian +)**

All the methods above converge to a local minimum

the highest and lowest point point were located.
Then you have to tighten the supervision step with some evolution rules



Reflection: one reflects the most points high in the opposite direction

Reflection + expansion

Contraction

multiple contraction

Mixed methods: Relaxation

The space in which we operate is orthonormal $\{e_1 \ldots e_{not}\}$

The idea is to break down a problem N-dimensional minimization in N problems 1D

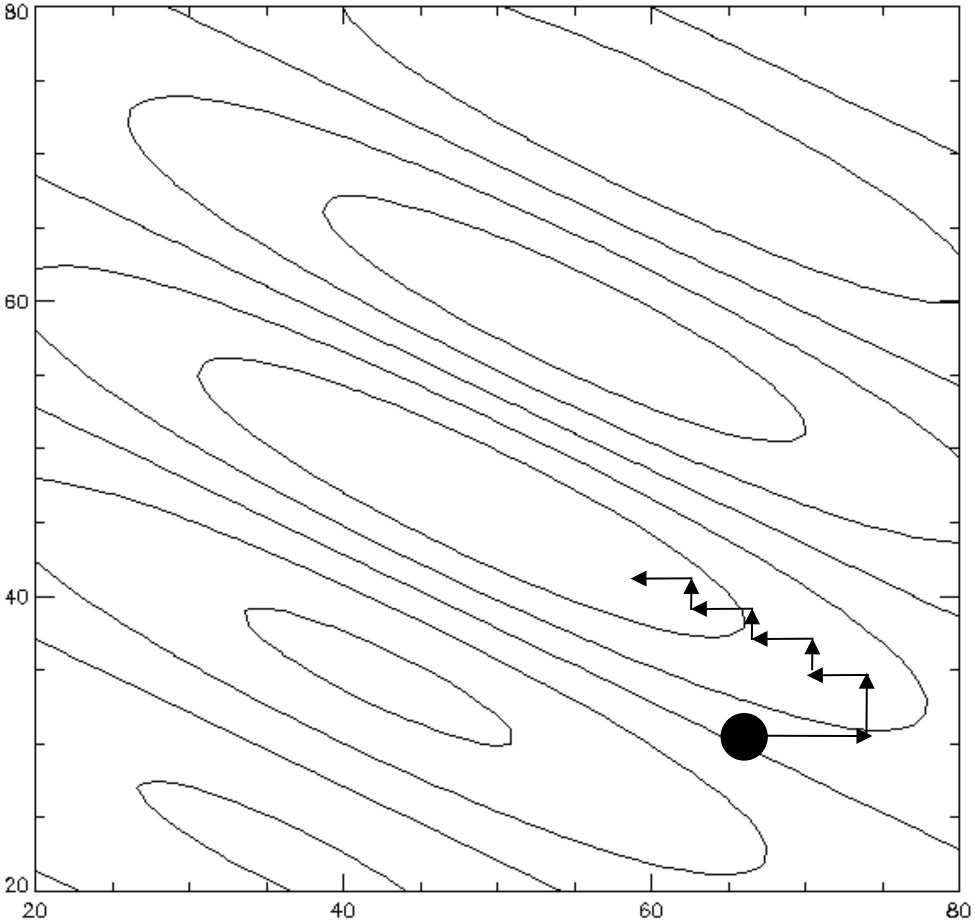
The idea is to cycle through dimensions. On each dimension we perform 1D minimization.

1- **Select an axis in** $\{e_1 \ldots e_{not}\}$, **beginning with** e_1

2- **Minimize function along this axis (any method) F** (x_i)

3- **If the method has converged, redo same operation on the next axis.**

4. **If all the axes have been tested and that the method has not converged to the minimum, go back to e1**



A 2D:
1- Get into the axis X
2- minimize along X
3. Stand on the axis Y
4 minimize along Y
5- Start in 1
if we have not reached the mini

Note: "minimize along the axis e_i "means

When we develop P : P = (p_1..., p_{not})

minimizes the function G (x) = F [(p1, ..., p_i+ X, ip ...$_{not}$)]

Only i^{th} variable is evolving while p_1..., p_{not} are fixed.
So G (x) is a function of one single variable.

To conduct the minimization of G (x) one can use any method already seen in 1D.

GRADIENT DESECENT (also Steepest Descent)

the known gradient. Same method as 1D

We know F and F (X)

X is multidimensional X = (x_1 , ..., x_{not})

Starting from a starting point X_0

Is calculated following

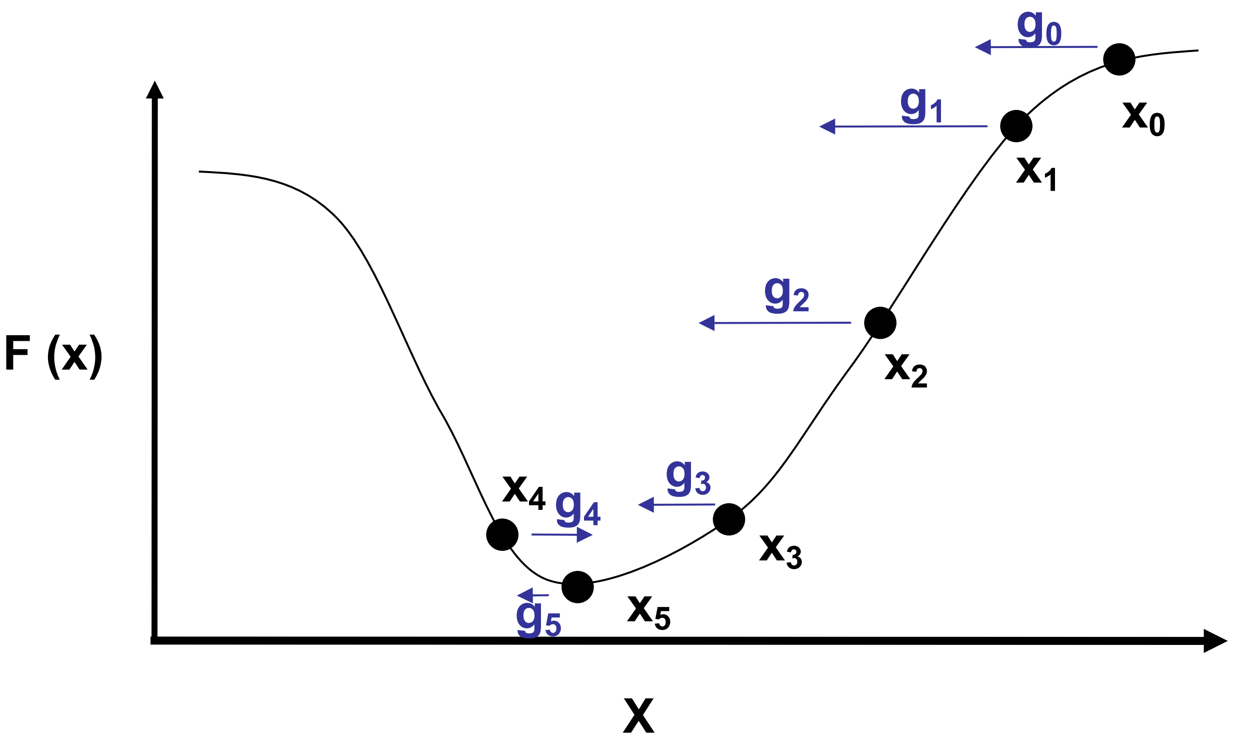
$$\vec{X}_{k+1} = \vec{X}_k + \vec{d}x_k = \vec{X}_k + d_k \vec{g}_k$$

where g_k is the direction of descent and$_k$ is not downhill
g_k = (G_{k1} , ..., g_{kn}) gradient vector
d_k a real number

The choice of g_k is D_k will be such that F (X_{k+1}) <F (X_k)

$$\vec{g}_k = -\vec{\nabla} f(X_k)$$

In the simplest method of gradient descent , g is simply - the gradient of F



Example 1D

g_i is less gradient of point i

$$X_{k+1} = X_k + d_k g_k$$

Either one chooses$_k$ as constant (d !!$_k$ is a vector)

Steepest descent, either with fixed step, or with optimal step (using the HESSIAN)

$$\overrightarrow{\boldsymbol{X_{k+1}}} = \overrightarrow{\boldsymbol{X_k} +} \boldsymbol{dX_k} = \overrightarrow{\boldsymbol{X_{k+1}}} + \boldsymbol{d_k} \cdot \overrightarrow{\boldsymbol{g_k}}$$

With $\overrightarrow{\boldsymbol{g_k}} = -\overrightarrow{\boldsymbol{\nabla f}}$

g$_k$ is the direction : - gradient (vector)
d$_k$ is the lentgh : scalar

$$f(X_k + dX_k) = f(X_k) + \vec{\nabla} f(X_k) \cdot \vec{d}X_k + \frac{1}{2} \sum_{i,j} \frac{d^2 f}{dx_i dx_j} dX_k^2 + o(dX_k^2)$$

$$\Rightarrow$$

$$\vec{\nabla} f(X_k + dX_k) = \vec{\nabla} f(X_k) + \sum_{i,j} \frac{d^2 f}{dx_i dx_j} \vec{d}X_k + o(dX_k)$$

The 1st and 2nd derivatives can be written as follows :.

$$\vec{\nabla} f(X_k + dX_k) = \vec{\nabla} f(X_k) + \sum_{i,j} \frac{d^2 f}{dx_i dx_j} \vec{d}X_k + o(dX_k)$$

$$\left.\begin{pmatrix} \dfrac{\partial f}{dx} \\ \dfrac{\partial f}{dy} \\ \dfrac{\partial f}{dz} \end{pmatrix}\right._{X_k} = \vec{g}$$

$$\left.\left.\begin{bmatrix} \dfrac{\partial^2 f}{\partial x \partial x} & \dfrac{\partial^2 f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial x \partial z} \\ \dfrac{\partial^2 f}{\partial y \partial x} & \dfrac{\partial^2 f}{\partial y \partial y} & \dfrac{\partial^2 f}{\partial y \partial z} \\ \dfrac{\partial^2 f}{\partial z \partial x} & \dfrac{\partial^2 f}{\partial z \partial y} & \dfrac{\partial^2 f}{\partial z \partial z} \end{bmatrix}\right|\right._{X_k} \begin{pmatrix} dx \\ dy \\ dz \end{pmatrix} = A\vec{d}x$$

$$\boldsymbol{\vec{\nabla} f(X_k + dX_k) \simeq \vec{\nabla} f(X_k) + A\overrightarrow{dX_k} = \vec{g} + A\overrightarrow{dX_k}}$$

The idea is that we only go in the direction of the steepest descent (-gradient)
So $\overrightarrow{dX}$ is always proportional to the gradient.

Now we want to find the minimum, so cancel the gradient keeping $\overrightarrow{dX}$ is always proportional to the gradient

$$0 = \vec{g} + A d\vec{x} \Rightarrow$$

$$or \;\; d\vec{x} = -d\vec{g} => $$

$$0 = \vec{g} - (A\vec{g})d \Leftrightarrow$$

$$0 = \vec{g}^{\,2} - \vec{g}\cdot(A\vec{g})d =>$$

$$d = \frac{g^2}{\vec{g}\cdot(A\vec{g})}$$

d is the optimum distance

Summary:

At each step of calculation:

- Calculate the descent direction: $\vec{g}_k = -\vec{\nabla} f(X_k)$

-**Calculate the length of descent:** $d_k = \dfrac{{g_k}^2}{\vec{g}_k \cdot (A\vec{g}_k)}$

- **Compute the new point X_k** $X_{k+1} = X_k + d_k \times \vec{g}_k$

REPEAT down to desired accuracy

But this method has its limitations:

Sometimes the steepest slope is not always optimal !!!

EXAMPLE:

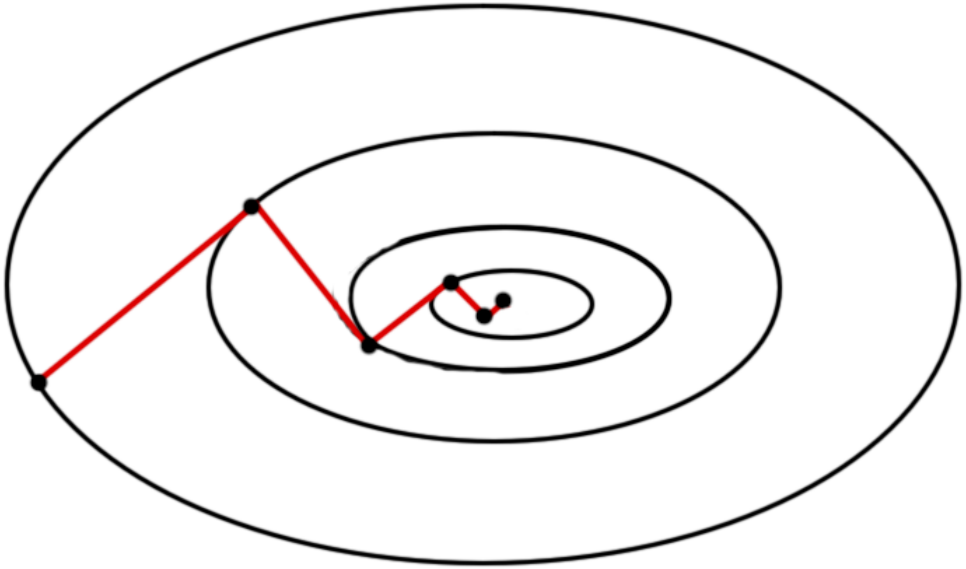
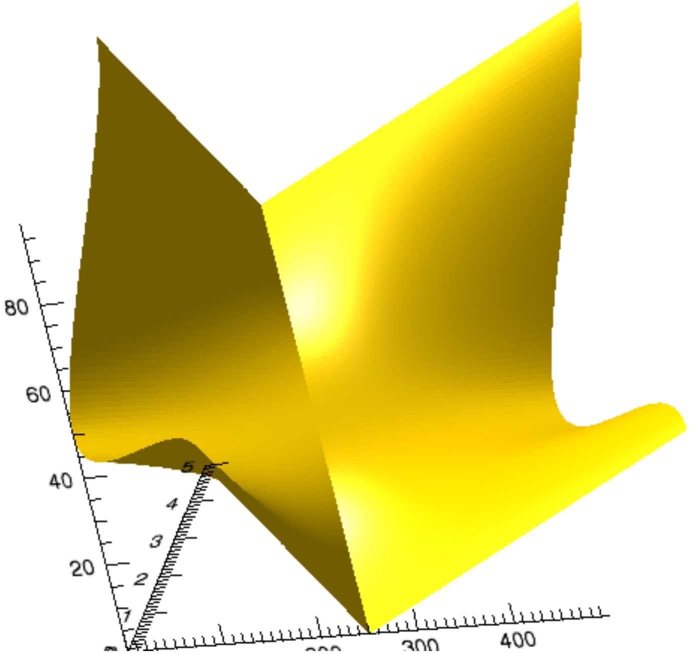


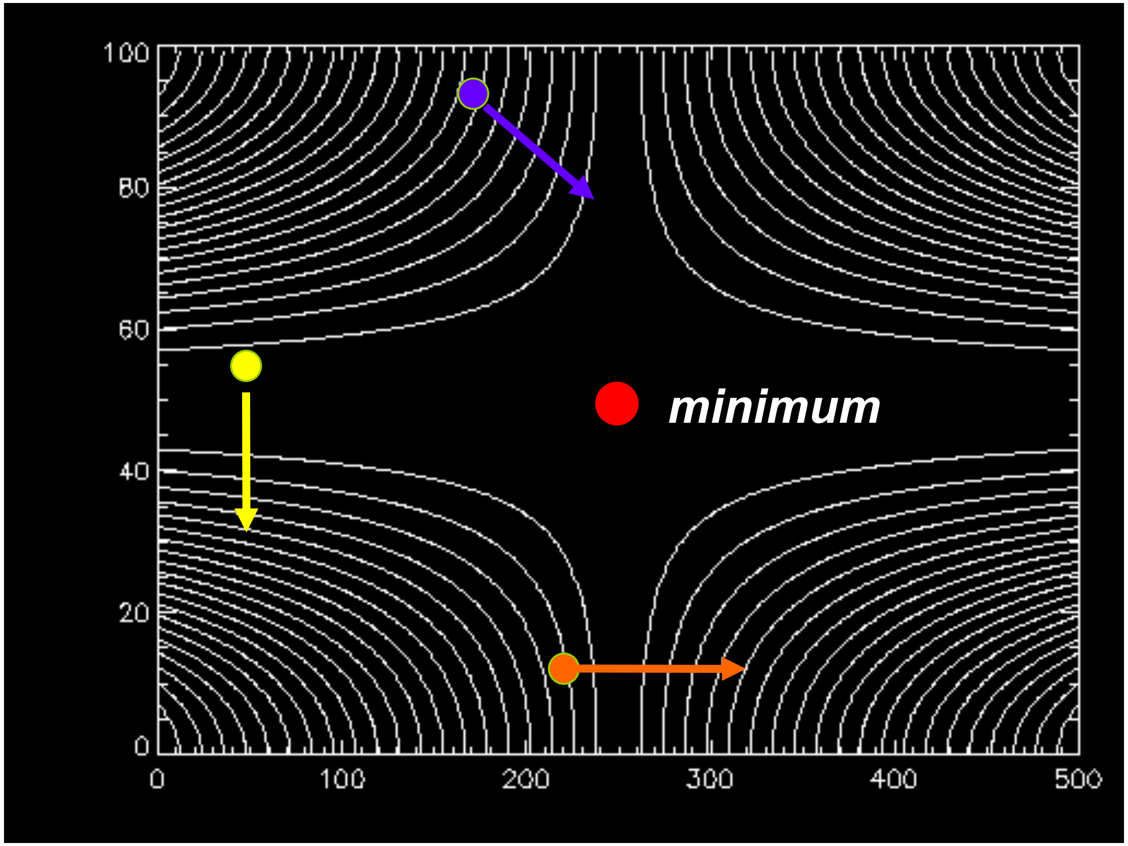
Figure 1: The contour plot of a function, with the steps of the steepest descent method in red

Either minimize F (x, y) (picture below)

Imagine that we are in a "valley" close.



The arrow is the direction of - gradient.
For yellow and orange dots gradient is not the best direction!



=> conjugate gradient

The algorithm will be like this:

$$\vec{X}_{k+1} = \vec{X}_k + d_k \vec{u}_k$$

where d_k is the distance of descent and u_k is the direction of descent. BUT in the conjugate gradient method, u_k is not the gradient (a priori) It will be something more complicated ..

In fact the descent direction u_k vary at each stage. In particular, it must have

$$\vec{u}_k \perp \vec{u}_{k+1}$$

So 2 questions:

1) How to calculate the descent direction u_k ?

2) Knowing u_k what is the best value of the length of descent_k ?

How to build the U_k vectors ?

We turn to a theorem of algebra. If A is symetrical, definite and positive (which is the case around a minimum, i.e. all eigen values are > 0) : then There exist a base of space {U_k} with N linearly independant vectors so That:

$$\boldsymbol{U_i^T}\mathbf{A}\mathbf{U_j} = \overrightarrow{\mathbf{U_i}} \cdot \left(\mathbf{A}\overrightarrow{\mathbf{U_j}}\right) = \mathbf{0}$$

i.e. all U_k are *conjugate* to each others with respect to Matrix A.

There are N of such vectors (N= dimension of space).

So starting from a point : $\overrightarrow{X_0}$ **and assuming that the minimum is** $\overrightarrow{X_M}$ **the Displacement toward the minimum can always be decomposed like this :**

$$\overrightarrow{\boldsymbol{X_M}} - \overrightarrow{\boldsymbol{X_0}} = \boldsymbol{\alpha_0}\overrightarrow{\boldsymbol{U_0}} + \boldsymbol{\alpha_1}\,\overrightarrow{\boldsymbol{U_1}} + \cdots + \boldsymbol{\alpha_N}\overrightarrow{\boldsymbol{X_N}}$$

it simply states that any vector can be decomposed on the base of Conjugate vectors with respect to A.

So far we have just assumed that the set of A-conjugate search directions exists. In practice we need a way to create it. There are several ways to choose such a set. The eigenvectors of A form a A-conjugate set, but finding the eigenvectors is a task requiring a lot of computations, so we better and another strategy.

There exist an alternative strategy that is iterative and does not need to compute the eigen vectors.

Let start from point $\overrightarrow{X_0}$. Since we want to nullify the derivative of f

$$\vec{\nabla} f(X_k + dX_k) \simeq \vec{\nabla} f(X_k) + A\overrightarrow{dX_k} = \vec{g} + A\overrightarrow{dX_k}$$

So $\overrightarrow{U_0} = \vec{g}_0 = -\left(\vec{g} + A\overrightarrow{dX_k}\right)(\textbf{ first step is a simple steepest desent})$

Then we go like this (we do not demonstrate, see'Numerical Recipes')

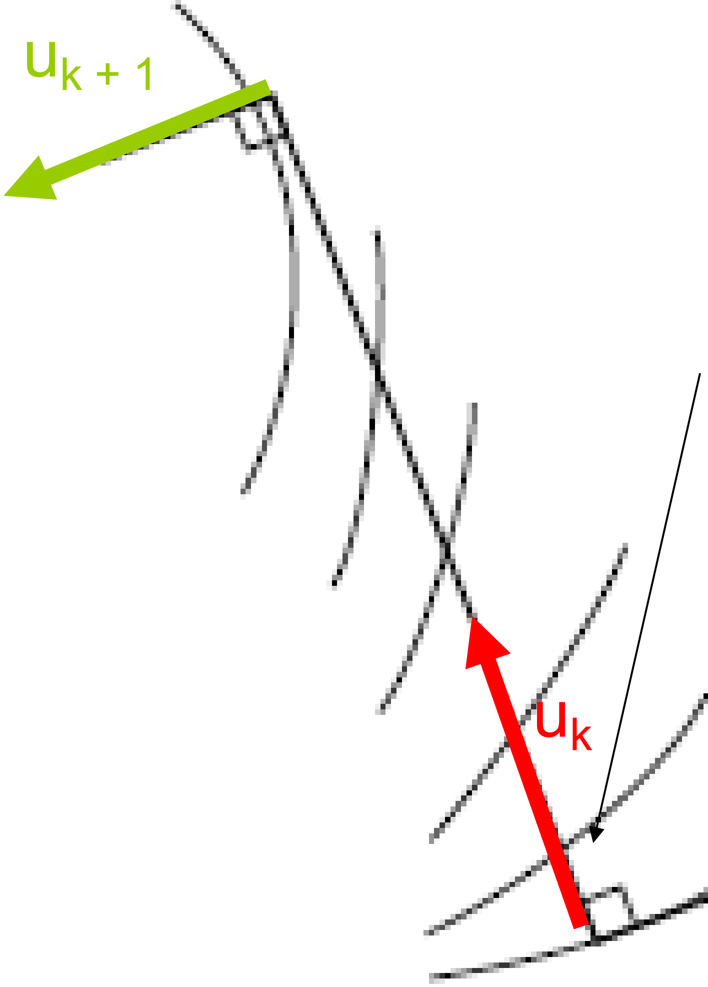
$$\alpha_k = \frac{g_k^2}{U_k^T A U_k}$$

$$\overrightarrow{X_{k+1}} = \overrightarrow{X_k} + \alpha_k \overrightarrow{U_k}$$ **(new point)**

$$\overrightarrow{g_{k+1}} = \overrightarrow{g_k} + A\overrightarrow{U_k}$$

$$\beta_k = \frac{\mathrm{g}_\mathrm{k}^2}{g_{k+1}^2}$$

$$\overrightarrow{U_{k+1}} = -\overrightarrow{g_{k+1}} + \beta_k \overrightarrow{U_k}$$ **(new vector of descent)**



It is seen that the U_k successiv
are perpendicular to each othe

A step started
perpendicular to the lines
levels

Convergence of conjugate gradient algorithm:

We can show that if f is EXACTLY quadratic

$$f(\vec{X}) = c - \vec{B} \cdot \vec{X} + \frac{1}{2} \vec{X} \cdot (A\vec{X})$$

Then just N iterations N dimensions are necessary

In practice, one rarely investigates exactly quadratic functions (If their minimum is analytically known).

So what if the algorithm does not converge after N iterations?

Simply restart from the last point obtained.

Interest of the conjugate gradient method:

VERY fast it converges to a minimum:
It can be shown that N-dimensional he simply steps N calculations
If the function is exactly square.

In practice :
Very often used in 'serious' calculations.

disadvantage:
Requires knowledge of the Hessian of the function F to determine the no descent.

The Fletcher-Reeves algorithm

This algorithm is based on the Conjugate Gradient

We have seen that the conjugate gradient algorithm chooses optimally descent directions through u_k.

The calculation of u_k **does not *explicitly* call the Hessian**

Then the function is minimized along the axis u_k **=> Minimizing 1D.**
and the conjugate gradient, the length downhill d_k, Is calculated with the Hessian

Principle Fletcher Reeves
We use the same downward directions as the conjugate gradient
Is minimized along the axis u_k **with classical minimization method to 1D**
(Using 3 points for example).
***advantage:* no need to know the Hessian**

In practice : **The Fletcher Reeves algorithm**

0. Start a starting point X0

1. **Calculate** u_k

$$\vec{u}_k = \vec{\nabla} f(X_k) + \frac{\left\| \vec{\nabla} f(X_k) \right\|}{\left\| \vec{\nabla} f(X_{k-1}) \right\|} \vec{u}_{k-1}$$

2. Minimize the function

$$g(d) = F(\vec{X}_k - d\vec{u}_k)$$

d is a positive real. It is thus minimizing 1D. Use your method favorite (Trisection, quadrature etc ...). The minimum is to$_k$

3. X$_{k+1}$ is then $\vec{X}_{k+1} = \vec{X}_k - d_k \vec{u}_k$

4. If you have not converged, again in 1

Fletcher-Reeves Convergence

It depends on your choice of minimization algorithm to 1D.

If you take the quadrature method, then, Fletcher-Reeves behavior will be very similar to the conjugate gradient.

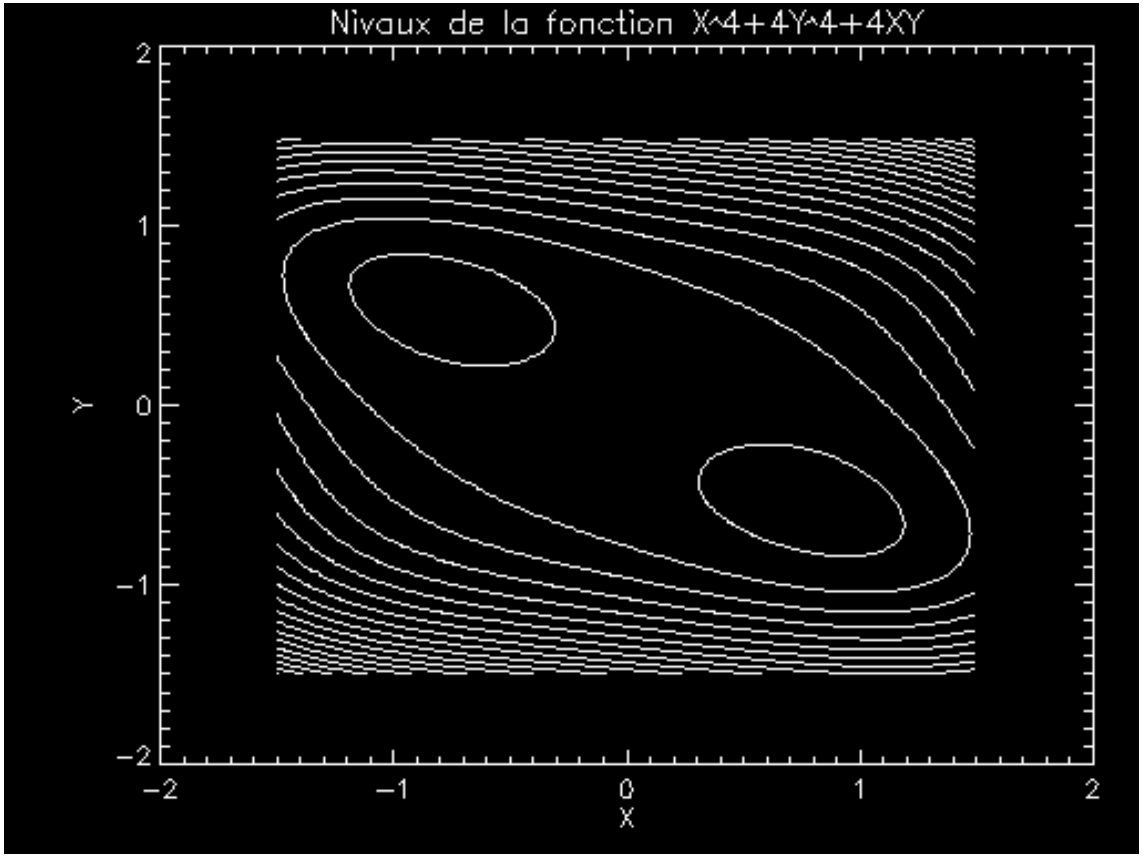
Rule to always respect (for all algorithms):
When we get a minimum, we must *restart* algorithm for this (new starting point) to be * on * convergence.

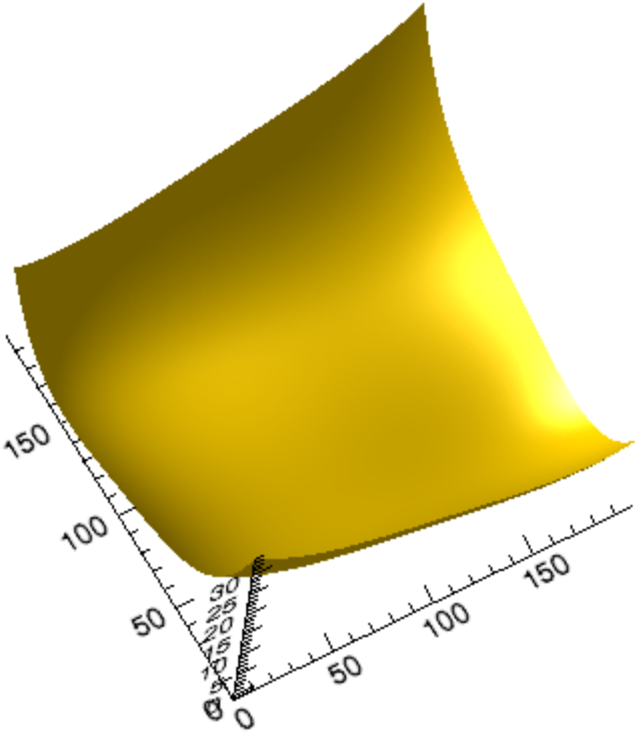
Some authors advocate the algorithm periodically reset
(⇔ impose $u_{k-1} = 0$ **)**

Fletcher Reeves Conjugate Gradient Vs.

It minimizes the function:

F (x, y) = x^4four are^4 4x + y





Which has two real minimum

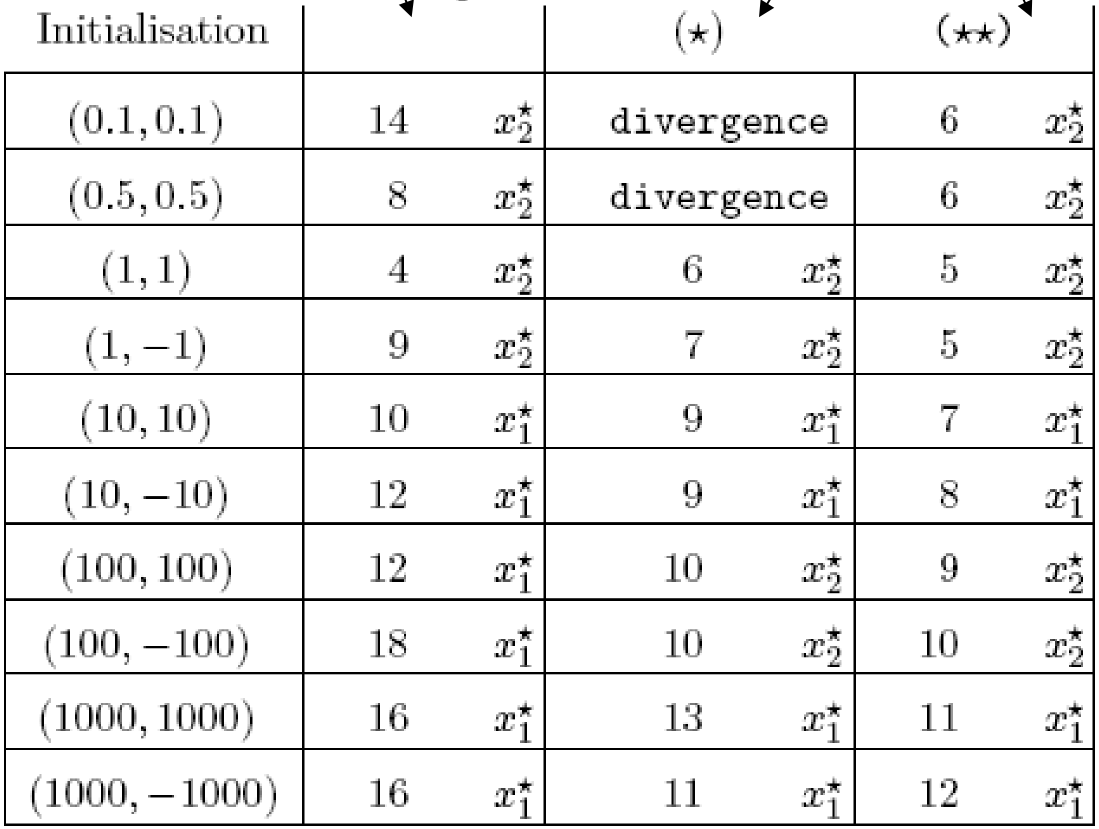
X * 1 = [- $8^{1/4}$ / 2, $2^{1/4}$/ 2] X * 2 = [$8^{1/4}$ / 2, -$2^{1/4}$/ 2]

X * 1 = [- 0.84, 0.59] X^*2 = [0.84, -0.59]

conjugate gradient

Fletcher-Reeves, with 1D as minimization method the method of the "golden rule."

N iterations



Initialisation			$(\star)$		$(\star\star)$	
$(0.1, 0.1)$	14	$x_2^\star$	`divergence`		6	$x_2^\star$
$(0.5, 0.5)$	8	$x_2^\star$	`divergence`		6	$x_2^\star$
$(1, 1)$	4	$x_2^\star$	6	$x_2^\star$	5	$x_2^\star$
$(1, -1)$	9	$x_2^\star$	7	$x_2^\star$	5	$x_2^\star$
$(10, 10)$	10	$x_1^\star$	9	$x_1^\star$	7	$x_1^\star$
$(10, -10)$	12	$x_1^\star$	9	$x_1^\star$	8	$x_1^\star$
$(100, 100)$	12	$x_1^\star$	10	$x_2^\star$	9	$x_2^\star$
$(100, -100)$	18	$x_1^\star$	10	$x_2^\star$	10	$x_2^\star$
$(1000, 1000)$	16	$x_1^\star$	13	$x_1^\star$	11	$x_1^\star$
$(1000, -1000)$	16	$x_1^\star$	11	$x_1^\star$	12	$x_1^\star$

Note 1:
The iteration nb does not depend much of the distance to the solution

Note 2:
The better Fletcher-Reeves with periodic stimulus

*** No recovery**
**** revival every 2 iterations**

Conclusion on deterministic methods:

They are very effective As the study is relatively simple function **:**
For example: quadratic function

They converge towards a LOCAL minimum :
It is never possible to say if you reach a local or a global minimum.
To find more minima must restart the algorithm with
different starting points.

All methods presented here may not be effective if :

If the function has a complex shape with numerous dimensions.
If there are many local minima "parasites"

Or if you need to know the minimum GLOBAL function:

We must then use other methods !!!

Method to find a global minimum:

"Simulated Annealing" in English

’Simulé recuit ’ in French ....
Also called "slow relaxation"

What to do to find a global minimum?

A possible method is "Simulated Annealing ». Another alternative is « genetic algorithm »
Simulated anneaeling in recent years has solved very complex problems like "traveling salesman" where deterministic methods are quickly trapped in local minima.

We briefly describe here.

It is a type method "Monte Carlo" therefore requires random numbers
.

It is inspired from thermodynamics ...

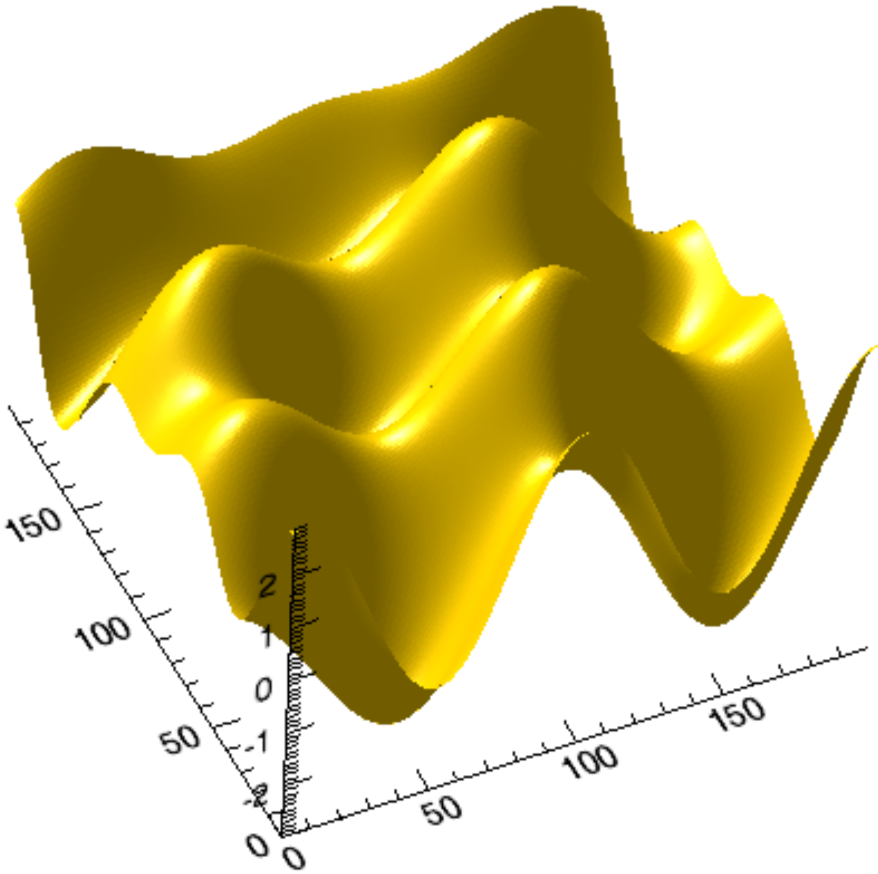
principle:
Natural systems spontaneously evolve towards a minimum energy or maximum total entropy:

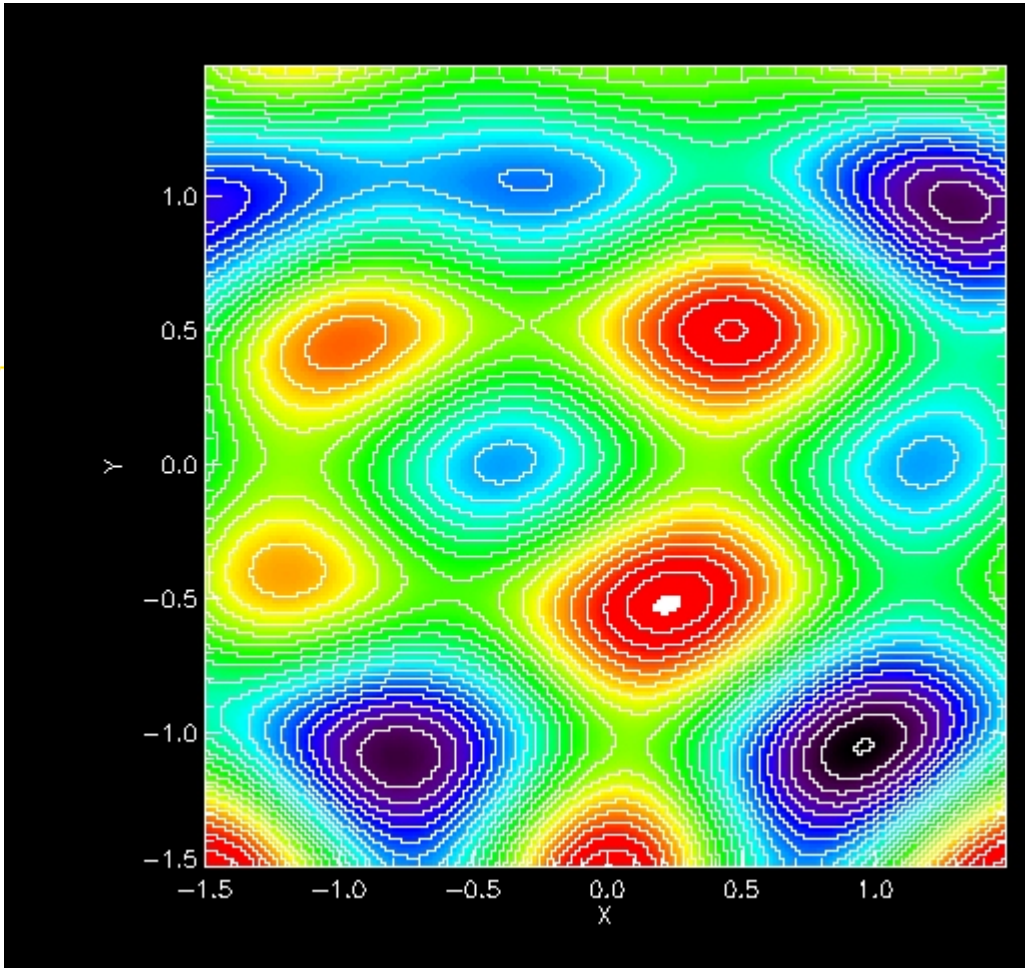
Example: gas (maximum entropy)
spatial conformation of a molecule (minimum energy)
crystallization (maximum entropy)
etc ....

The system explores all possibilities during some time (slow relaxation) then stabilizes inside minimum. It is a cooling process.
Over this cooling is slower, the system will be close to the absolute minimum.

The function F (X) will be called to minimize the according to "cost."
We must therefore minimize the "cost"

Example: Here is a function F (X, Y) which it is very difficult to determine the min absolute. A deterministic method will give you just the nearest local minimum.





Now let’s get inspired from thermodynamics:

In thermodynamics the probability of an energy state E is

$$P(E) \propto e^{-E/kT}$$

**Expressing the idea that if a system is in thermal equilibrium at temperature T,
then its possible energy states are distributed probabilistically
with a probability distribution that decreases exponentially with energy.**

So even if the lower energy state is most likely, it is still possible
to "jumps" (with low probability) through a local minima of energy, but then
eventually converge to the global energy minimum.

So sometimes the system goes back to larger energies , but in average it gets down towards lower energies.

This is the FORCE of this approach. A deterministic method cannot do that

$$P(E) \propto e^{-E/kT}$$

The simulated-annealing method uses that probability.

- E is the cost function F: F is the function fo be minimized. It is treated as an energy

- T is a dummy control parameter is slowly decrease the temperature T of the system.

- The Boltzmann constant K will be replaced by an arbitrary constant so that P (E) is a number accessible by the machine.

The probability of passing from one state E_k to state E_{k+1} will be :

$$p(E_k \rightarrow E_{k+1}) = e^{-\frac{(E_{k+1}-E_k)}{kT}}$$

If E_{k+1} <E_k then P> = 1 => The transition is always accepted. (it minimizes !!!)

If E_{k+1} > E_k so 0 <P <1 => pulling a figure X between 0 and 1. If X <P then the transition is accepted

In practice, the method will be:

Surgery
repeated
N times
constant T

1- Starting from Ek energy, and a temperature T

2 randomly generate a new state, K + 1, close to the K state, energy Ek + 1 ($\Delta E \sim KT$)

3- Let'd chek if we accept the state K + 1
For this we calculate the transition from K to to K + 1:
Draw a random number between 0 and 1 X (distrib. Uniform)
If X> P then accept it, otherwise it reject it

4- When I no longer decreases, it decreases T and start again in 1

In step 2, the random generator should be able to generate new whose configurations ΔE Typical is about KT ...
It is in step 2 that hides the effectiveness of the method.

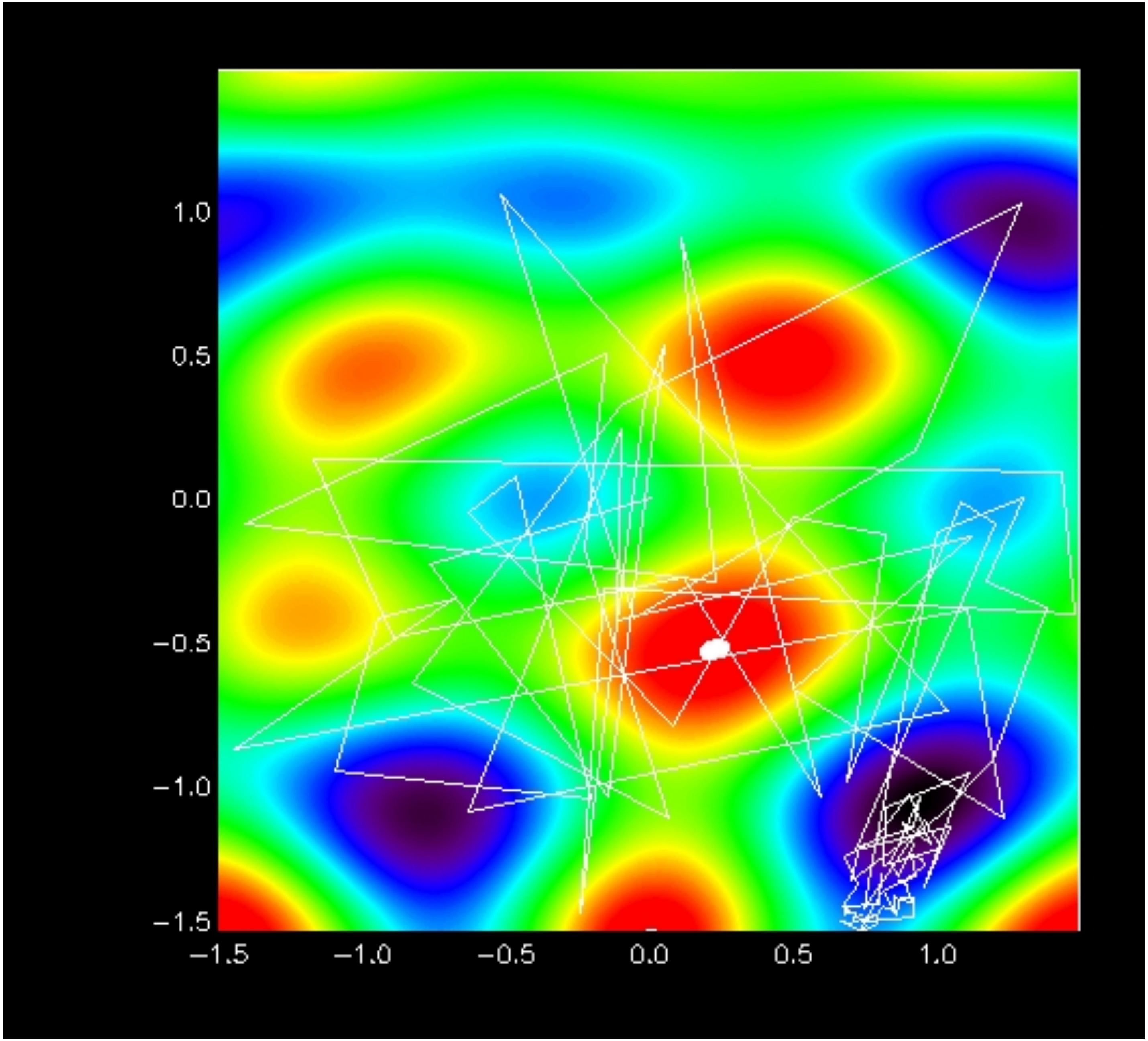
example:

Starting point:
[0,0]

Arrival point :
[0.68, -1.42]

10^5 iterations

310 changes accepted



Conclusion on simulated annealing:

Method that can be effective when seeking an absolute minimum.

The result depends STRONGLY on method used to generate a new random state.

Very time consuming calculation => Use only if other methods are ineffective.

The method of "simulated annealing" is the basis of other methods of simulati type of physical "Monte Carlo" called "Metropolis" methods for simulate a gas or a crystal.

Alternative to simulated annealing: Genetic Algorithms.

Conclusion on minimizing without restraint

The problem is to minimize a function F (X) N unconstrained dimensions on X.

-All methods require X0 start.
- Deterministic methods converge to the *local minimum* the closest.
- The more you know about the function (Gradient, Hessian) plus minimization will be effective. (Conjugate gradient, Fletcher-Reeves)
- . Methods to Sun N ALL trying to reduce the problem to one dimension: it minimizes successively given directions

If the function is "noisy" of many local minima ... then we must used a method of the type "slow relaxation" as the "Simulated Anealing".
This is a Monte Carlo method.
But the price is ... (1) a time of great computing
(2) There must be a generator of new states effective and there is no "practical recipe"

Minimization Constraint

Many problems in physics, engineering and economics require minimize function but soums to several constraints.

Examples:

aeronautics:

The optimal design of a wing that minimizes friction but ensure that Constraints on the wing are not too strong.

Chemistry:

Chemical composition in equilibrium, which minimizes the Gibbs energy but which retains the total mass

Etc ...

Several techniques exist.

We shall see one here, which is often found in thermodynamics:

The method of Lagrange multipliers.

Method can also be used analytically.

The Lagrange multipliers method can solve the following problem:

Let X**= [X1, x2, ..., xn] a set of N variables**

We want to minimise f (X**) knowing that** X **is constrained so that we want g (X) = 0**

Example (from economy):

We want to maximize the function:

$$f(x,y) = x^{2/3} y^{1/3}$$

X and Y represent them working investment (x) and capital (y)

Knowing that the total cost is

$$g(x,y) = p_1 x + p_2 y = c$$

If there was no constrain, it would suffice to increase x and y indefinitely.

But the constraint g (x, y) = c makes this impossible.
So we need to maximize f (x, y) with constrain g(x, y) = c

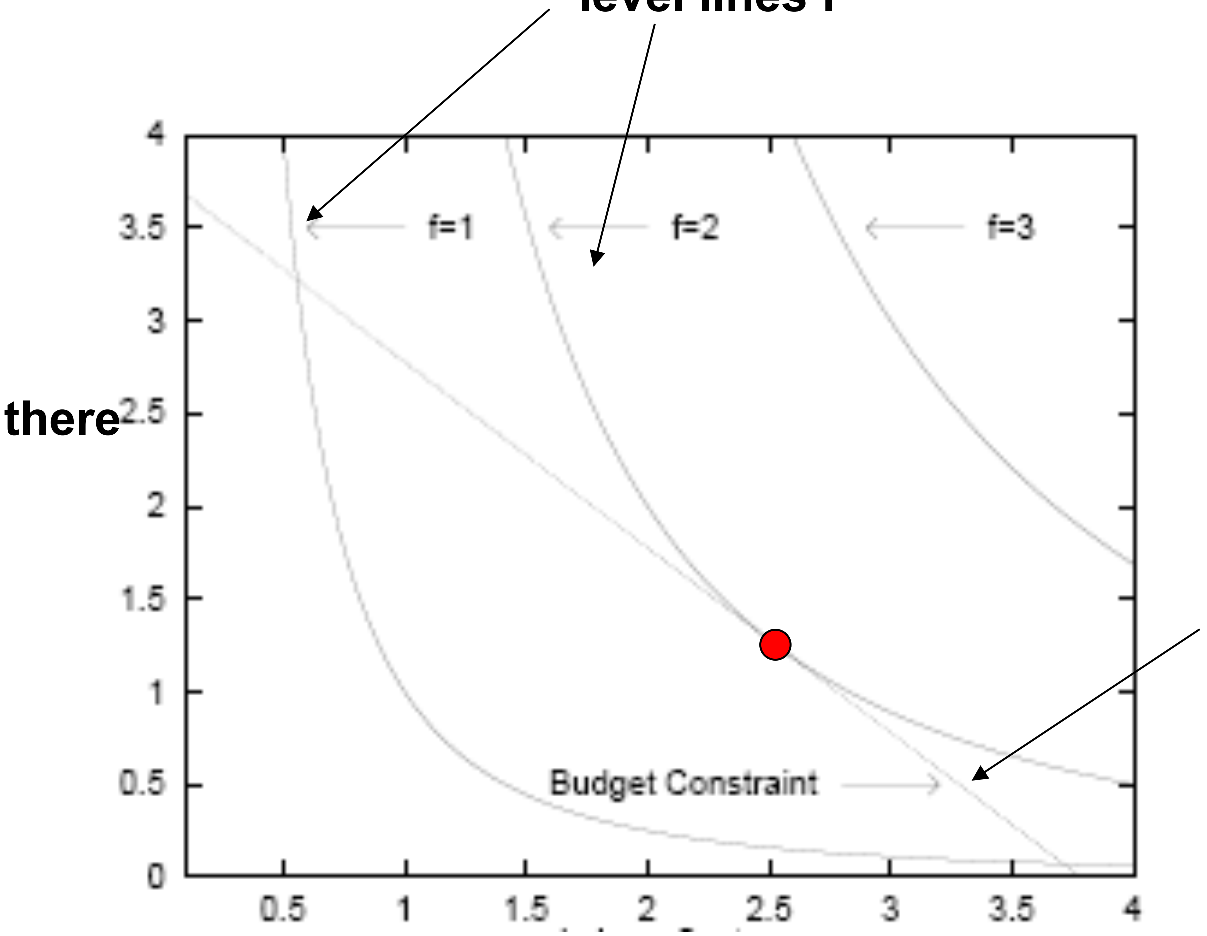
It is a typical a « constrained optimization problem ».

How to solve this problem ?

Let's F study the level lines of F, for constant values of G

We will take for g (x, y) = c ⇔ x + y = 3.78 ...

level lines f



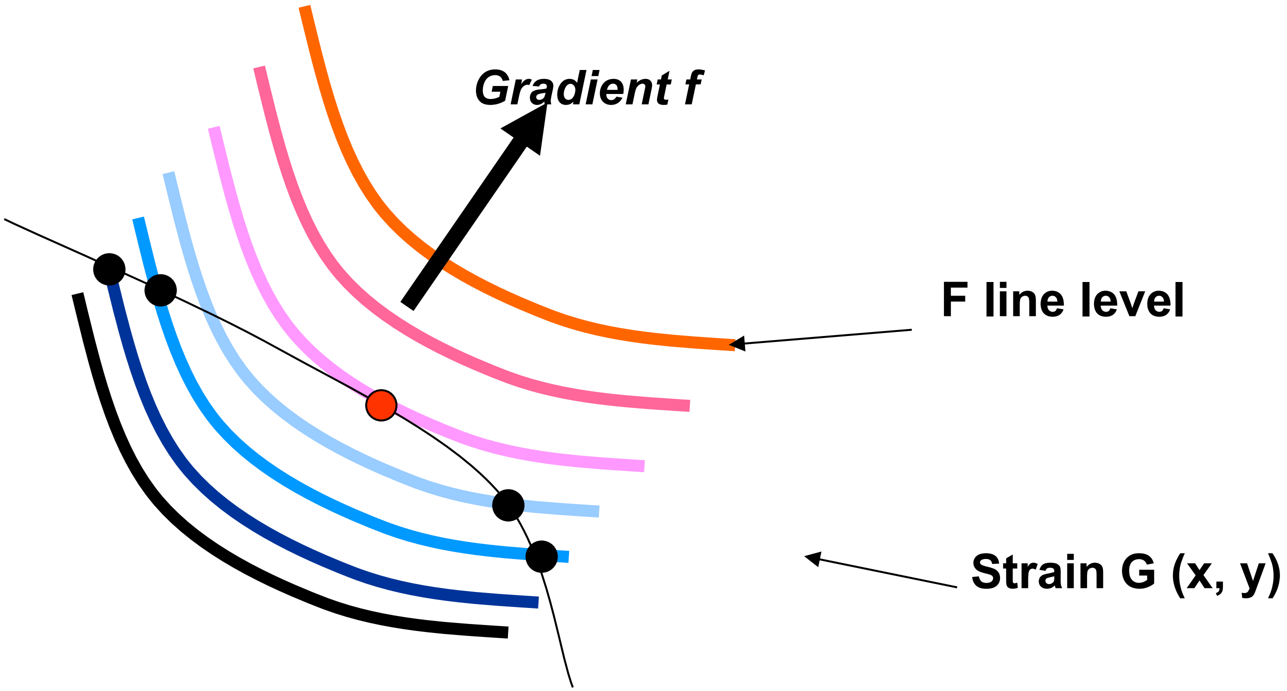
there

G (x, y)

X

Note that at the level of the possible solution (red) of the f level line
Is parallel to the line level g.
To this extend $\vec{\nabla} f = k\vec{\nabla} G$

graphic demonstration



● : F can be increased again, moving to the right or left

● : One can increase f: the left and right directions minimize f

Then the the maximal condition is:

"F // gradient is the gradient of G"

$$\vec{\nabla} f = \lambda \vec{\nabla} G \, , \lambda \in \Re$$

$$\Leftrightarrow$$

$$\begin{cases} \dfrac{\partial f}{\partial x} - \lambda \dfrac{\partial g}{\partial x} = 0 \\ \dfrac{\partial f}{\partial y} - \lambda \dfrac{\partial g}{\partial y} = 0 \\ \dfrac{\partial f}{\partial z} - \lambda \dfrac{\partial g}{\partial z} = 0 \end{cases}$$

λ is called "Lagrange Multiplier"

N equations (All gradients)

with N unknowns (x, y, z, etc ...)

+ 1 equation: G (x, y) = 0 One unknown:λ

So N + 1 equations a N + 1 unknowns

This system is then solved numerically by the methods we saw being ...

Summary : It maximizes F (X) with the constraint G (X) = 0. X a vector with N components

1- Ask H (X,λ) = F (x) -λG (X)

2- Write $$\forall i \quad \frac{\partial f}{\partial x_i} - \lambda \frac{\partial g}{\partial x_i} = 0$$

3. Find the x_i and λ of N equations of the gradient solutions of the equation + G (X) = 0

example:
We want to produce cylindrical cans of volume V0 set but Minimum area of S. Both parameters are r and h

Function to be minimized

$$S(r,h) = 2\pi rh + 2\pi r^2$$

$$V(r,h) = V_0 = \pi r^2 h$$

constraints

$$H = S(r,h) - \lambda V(r,h)$$

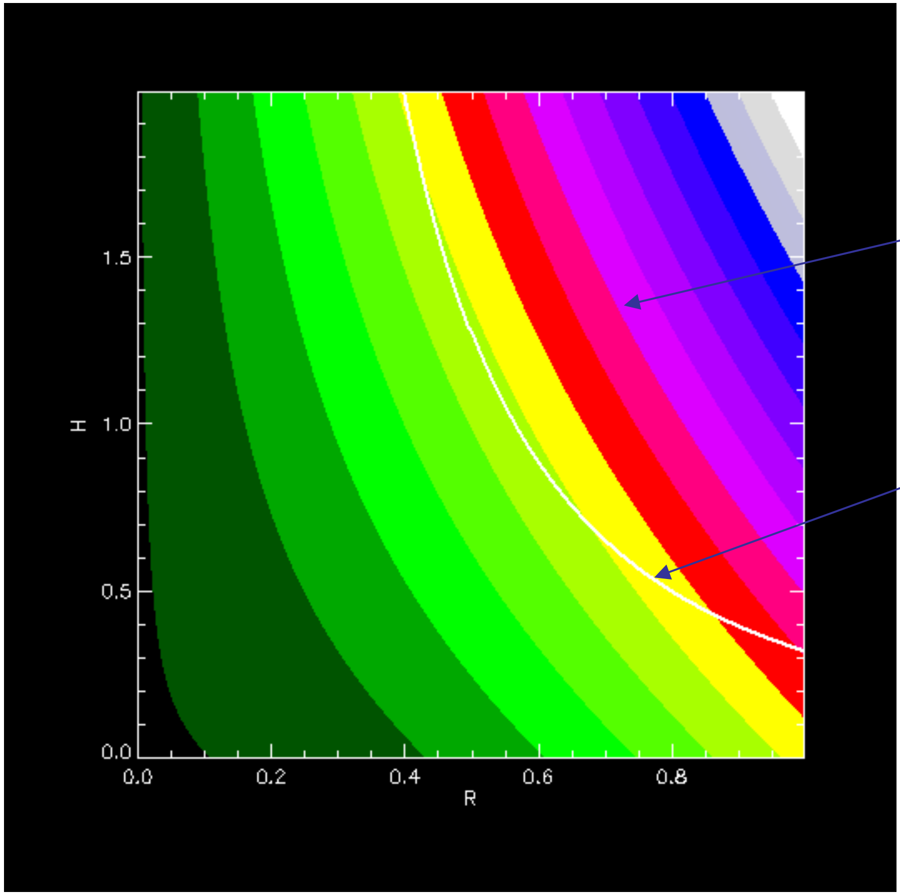
$$\Rightarrow$$

$$\begin{cases} \dfrac{\partial S}{\partial r} - \lambda \dfrac{\partial V}{\partial r} = 0 \Leftrightarrow 2\pi h + 4\pi r - \lambda 2\pi rh = 0 \\ \dfrac{\partial S}{\partial h} - \lambda \dfrac{\partial V}{\partial h} = 0 \Leftrightarrow 2\pi r - \lambda \pi r^2 = 0 \\ \pi r^2 h = V_0 \end{cases}$$

3 equations
3 unknowns: r, h,λ

$$\begin{cases} 2\pi h + 4\pi r - \lambda 2\pi r h = 0 \\ 2\pi r - \lambda \pi r^2 = 0 \\ \pi r^2 h = V_0 \end{cases}$$

System of nonlinear equations solved by substitution

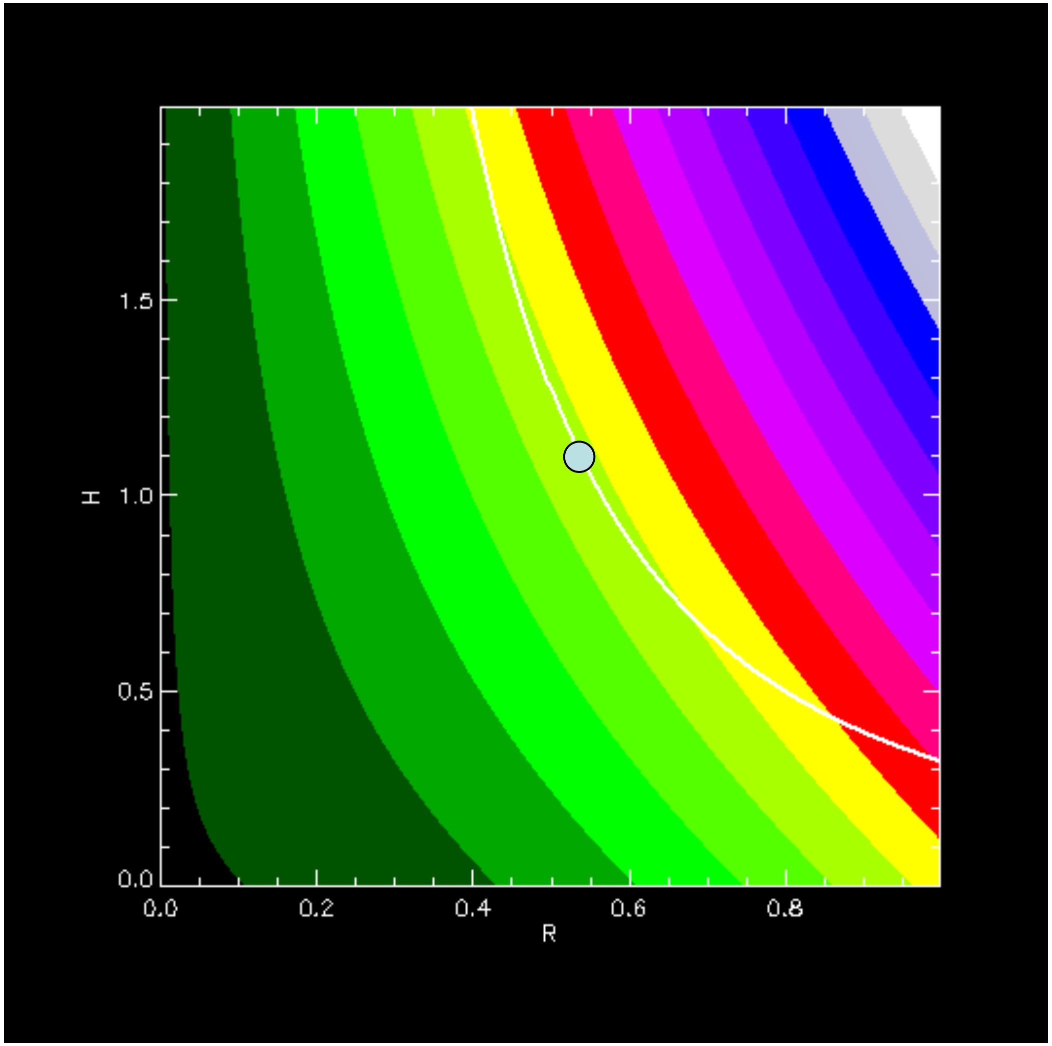


Colors: levels Lines S maximize

Online: Constraint Pi r^2 h = V_0= 1

$$\begin{cases} 2\pi h + 4\pi r - \lambda 2\pi r h = 0 \\ 2\pi r - \lambda \pi r^2 = 0 \\ \pi r^2 h = V_0 \end{cases} \Longrightarrow \begin{cases} 2\pi h + 4\pi r - \lambda 2\pi r h = 0 \\ \lambda = 2 / r \\ \pi r^2 h = V_0 \end{cases}$$

$$\begin{cases} h = 2r \\ \lambda = 2 / r \\ \pi r^2 h = V_0 \end{cases} \Longrightarrow \begin{cases} h = 2r \\ \lambda = 2 / r \\ r = \left(\dfrac{V_0}{2\pi} \right)^{1/3} \end{cases}$$



$$\begin{cases} h = 1.08 \\ \lambda = 3.69 \\ r = 0.54 \end{cases}$$

V = 1

What to do when we have more of ONE constraint?

We introduce as many Lagrange multipliers as constraints:

Example : N **variables and** M **constraints**

Let x1, ..., Xn N variables.

We want to maximize F (x1, ..., x N)

With M-type constraints G_i(X1, ..., Xn) = 0, for i = 1 ... M

Method

Lagrange multipliers are introduced M: λ_1..., λ_M

And n solves the equations N $\frac{\partial f}{\partial x_i} = \sum_{j=1}^{M} \lambda_j \frac{\partial G_i}{\partial x_i}$

The M + constraints. This allows to findλ_i and all x_i

The Lagrange multipliers method is the simplest method to optimize a problem with constraints.

the parameter λ has no real meaning here.

Other methods exist that we will not be detailed here.

It is widely used in thermodynamics where, for example, it allows to calculate the chemical equilibrium composition of a chemical reaction

Summary: How to tackle a minimization problem

Either minimize the function F (x1, ..., Xn) N variables.

1. Is the problem constrained ?

If yes: Lagrange multipliers or other

2. Is the a simple problem? (Idea of the minimum, absence of many parasite minima)

If yes then we can use all the deterministic methods

3. Simple Problem: Do you know the gradient of F?

If yes , you are in the best case: use eg conjugate gradient (Hessian) or Fletcher Reeves

If not, and N = 1: Use a method of trisection and quadrature
If not, and N> 1: Only the "Amoeba" method ("Amoeba") or mixed method not studied here

4. If the problem is complex (No idea of the minimum several parasites minima etc ...)

You are in the most difficult cases. Try a Monte Carlo method such as simulated annealing ("Simulated Annealing"). Or check genetic algorithm.