DE LA RECHERCHE À L'INDUSTRIE

cea

www.cea.fr

Numerical optimisation issues

A broad introduction

J-B. Blanchard

jean-baptiste.blanchard@cea.fr

HPC and Uncertainty Treatment
with Open TURNS and Uranie | 2020/05/14

Outline

Introduction
- Purpose
- Mathematical model
- Vocabulary

Mono-criteria problems
- Rosenbrock function, a classical example
- Historical approach
- Examples with current algorithms
- Heuristic / model based techniques

Multi-objectives problems
- Vocabulary
- Example, in a nutshell

Conclusion

Outline

Introduction
- Purpose
- Mathematical model
- Vocabulary

Mono-criteria problems
- Rosenbrock function, a classical example
- Historical approach
- Examples with current algorithms
- Heuristic / model based techniques

Multi-objectives problems
- Vocabulary
- Example, in a nutshell

Conclusion

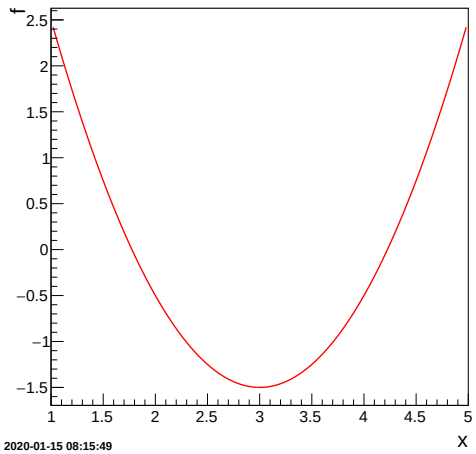
Purpose of optimisation

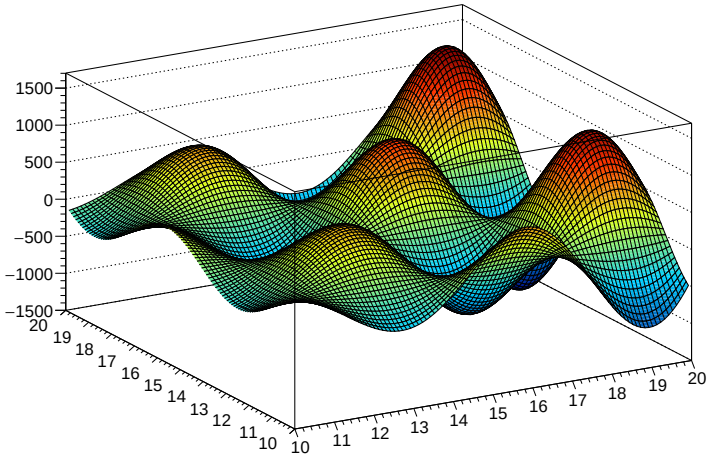
Find an optimum. An optimum level or state of something is the best level or state that it could achieve (Collins).

What for ?

- Location of warehouse: best location to minimise shipping cost
- Vehicle: minimise weight and air resistance through design
- Game strategy: find the best bet you can do given your hand and statistics
- Medicine: optimise insulin delivery to minimise blood sugar deviation
- Energy: optimise the usage of all available ressource to meet the energy request at lowest cost.
- Machine learning: get the best value for hyperparameters

→ And many more...





How to formalise this ? (1/2)

Seeking for the maximum of $f \Leftrightarrow$ Seeking for the minimum of $-f$

➟ Optimisation problem is always discussed as minimisation problem

Generic notations [1, 2, 3]

The aim is to obtain

$$\min_{x \in X} f(x), \quad \text{where} \quad \mathrm{f} : \mathbb{R}^{\mathrm{n}} \to \mathbb{R}^{\mathrm{q}} \tag{1}$$

knowing that X is a subset of $\mathbb{R}^n$ ($X \subseteq \mathbb{R}^n$).
One can have also aside equations to obey

$$h(x) = 0, \quad \text{where} \quad \mathrm{h} : \mathbb{R}^{\mathrm{n}} \to \mathbb{R}^{\mathrm{m}} \tag{2}$$

$$g(x) \leq 0, \quad \text{where} \quad \mathrm{g} : \mathbb{R}^{\mathrm{n}} \to \mathbb{R}^{\mathrm{p}} \tag{3}$$

Hereafter, when $q = 1$, the *real value* of x that minimise the f function is called x^*.

In the generic case (1/2)

Definitions

- **Decision variables**:
  - variables that can be changed to modify the system behaviour;
  - notation: $x = (x_1 \dots x_n)^T \in X$, ($X \subseteq \mathbb{R}^n$), n being the input dimension space ($n \geq 1$);
  - aim: find their best value $x^* = (x_1^* \dots x_n^*)^T$

- **Objectives** / **criteria**:
  - variables that are the measurement to be minimised
  - notation: function $f : \mathbb{R}^n \to \mathbb{R}^q$, $q \geq 1$
  - aim: get their lowest values
- Problem is **single-objective** (SO) when $q = 1$ and **multi-objectives** (MO) when $q > 1$

- **Constrains**:
  - any functions that affect decision variables / objectives (represent the context of the problem)
  - notation:
    - $h : \mathbb{R}^n \to \mathbb{R}^m$ called equality constrains ($m \geq 0$)
    - $g : \mathbb{R}^n \to \mathbb{R}^p$ called inequality constrains ($p \geq 0$)
  - aim: final solution(s) must respect these conditions
- Problem is **unconstrained** when $p = m = 0$ and **constrained** when $p + m > 0$

In the generic case (2/2)

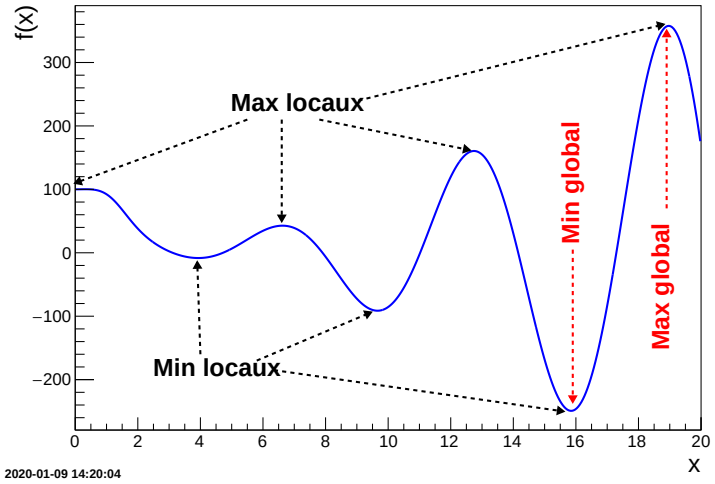
Nature of the optimum

- **Local minimum** : the x_0 defined so that

$$\exists \text{ neighbourhood } \mathcal{V}_{x_0} \text{ so that } \forall x \in \mathcal{V}_{x_0} \bigcap X, f(x) \geq f(x_0)$$

- **Global minimum** : the x_0 defined so that

$$\forall x \in X, f(x) \geq f(x_0)$$



Consequences : nature of the method

- **Local algorithm**: starts from a **candidate solution** and then iteratively moves to a neighbor solution until convergence to a minimum.
- **Global algorithm**: finds the global minima, through the use of numerical solution strategies (as analytical methods are not applicable). Generally more complex and costly.

Main hypothesis and their consequences

Underlying hypothesis, used in the next few slides, are described along with their consequences

- **Continuity**: both constrains and objectives are modeled by continuous function.
- No optimisation with only integers (called **discrete optimisiation** or **combinatory**).
- Using floating-point algorithms [4] leads to the need of having:
  - a **tolerance** ϵ to state wether a candidate x_C is valid ($|x_C - x^*| < \epsilon$) $\Leftrightarrow$ **stopping criteria**
    In real x^* rarely known. **Tolerance** is used a convergence measurement between iteration
  - a reasonnable accuracy on values (particularly for gradient-based methods, approximated by finite differences)

- **Deterministic**: constrains and objectives do not contain a stochastic part: identical configurations give same results.
- Uncertainty modelisation or intrinsic stochastic phenomena can be taken into account with **robust optimisation**, not discussed here.

- **Differentiability**: Not compulsory in the rest of the slides, but for some algorithm:
- gradient-based: functions need to be differentiable
- hessian-based: functions need to be doubly-differentiable

DE LA RECHERCHE À L'INDUSTRIE

Outline

Introduction
- Purpose
- Mathematical model
- Vocabulary

Mono-criteria problems
- Rosenbrock function, a classical example
- Historical approach
- Examples with current algorithms
- Heuristic / model based techniques

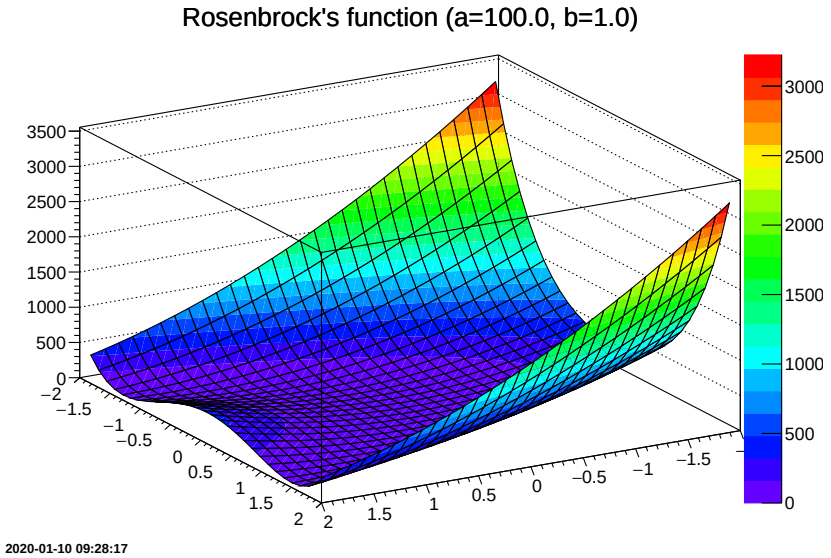
Multi-objectives problems
- Vocabulary
- Example, in a nutshell

Conclusion

One of many usual test function

Introduced by Rosenbrock to show the superiority of one its algorithm [5]

$$f(x_1, x_2) = 100 \times (x_2 - x_1^2)^2 + (1.0 - x_1)^2$$



Simple function ($n = 2$, $q = 1$, $p = m = 0$)

- **Single-objective**, **unconstrained**, **deterministic**, **differentiable**
- Does not provide ∇f nor $\nabla^2 f$
  $\Rightarrow$ Can use finite differences approximation
- Min value expected for $(x_1, x_2) = (1, 1)$

Will start by discussing the historical Newton method

Sufficient Optimality Conditions (unconstrained)

For unconstrained problem

Given a candidate $\hat{x} \in X$

- First order SOC: cancellation of the gradient

$$\nabla f(\hat{x}) = \begin{pmatrix} \frac{\partial f(\hat{x})}{\partial x_1} \\ \frac{\partial f(\hat{x})}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

  - ➙ $\hat{x}$ is a minimum, a maximum or a saddle point
- Second order SOC: the hessian is positive definite

$$\nabla^2 f(\hat{x}) = \begin{pmatrix} \frac{\partial^2 f(\hat{x})}{\partial x_1^2} & \frac{\partial^2 f(\hat{x})}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f(\hat{x})}{\partial x_1 \partial x_2} & \frac{\partial^2 f(\hat{x})}{\partial x_2^2} \end{pmatrix} \text{ is positive definite}$$

  - ➙ $\hat{x}$ is a local minimum
- GSOC: On X, f is (convex / strickly convex)
  - ➙ $\hat{x}$ is (a / the) global minimum

Newton's method: principle

The principle is too approximate locally, around x_k, the function with a quadratic model

$$m_k(x) = f(x_k) + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k),$$

By setting $d = x - x_k$, one can write this as

$$m_k(x_k + d) = f(x_k) + d^T \nabla f(x_k) + \frac{1}{2} d^T \nabla^2 f(x_k) d,$$

The aim is then to apply First order SOC

$$m_k(x_k + d) = \nabla f(x_k) + \nabla^2 f(x_k) d = 0$$

which means

$$d = -\nabla^2 f(x_k)^{-1} \nabla f(x_k) \Leftrightarrow x = x_k - \nabla^2 f(x_k)^{-1} \nabla f(x_k)$$

If hessian in x_k is invertible, an iteration of the local Newton method is a minimisation of the model

$$x_{k+1} = \min_{x \in \mathbb{R}^n} m_k(x)$$

This process is repeated until converge.

Newton's method: application

Never used like this, because:

1. Need to get a function that provide analytic gradient
2. It is long and complicated to get a proper analytic estimation of the hessian
3. The resulting Hessian might not be invertible
4. Constant step variation strategy is highly unstable, one might want to use

$$x_{k+1} = x_k - \alpha_k \nabla^2 f(x_k)^{-1} \nabla f(x_k),$$

   where many smart ways are available to define α_k so that it respect Wolfe's conditions
5. It converges quickly but only if the chosen initial value x_0 is close to the real x^*

Many existing variations to circumvent these issues

- **Quasi-Newton** algorithms rely on gradient information but do not request Hessian
- **Direct** algorithms do not require hessian nor gradient

Many packages provide a bunch of algorithms to handle these SO issues. We'll been showing few examples from NLOPT [6].

Application with few techniques

The names in bold font are the name from the NLOPT package

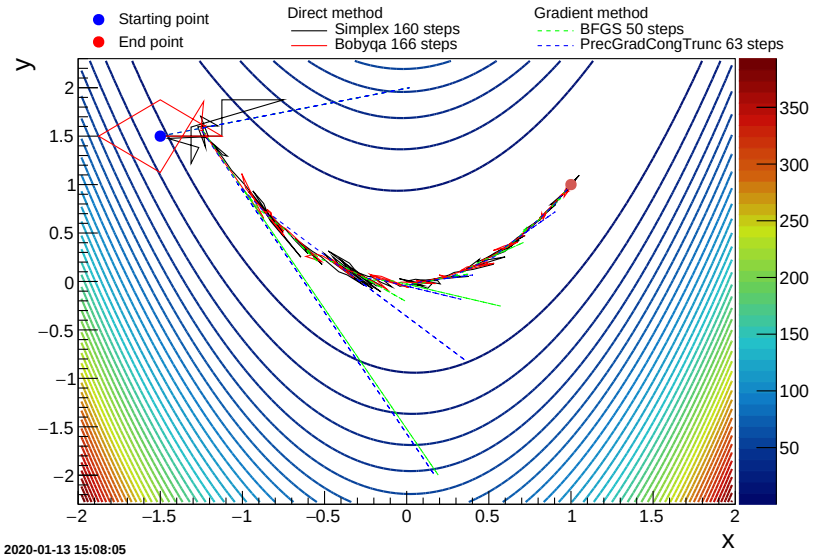
Direct methods

- NELDERMEAD (generally known as SIMPLEX):
  - compute a bloc $\mathcal{B}$ ($n+1$ estimations) and its barycentre b_k
  - test solutions along line from b_k to $\mathcal{B}$'s worst solution (w_k)
  - include best solution in the bloc instead of w_k and start over
- BOBYQA: *trust-region* method
  - compute a model approximating the objective $m_k(s) \simeq f(x_k + s)$ (for small s) while maintaining $\Delta k > 0$ (trust-region radius)
  - m_k is constructed by interpolating quadratic approximation for $f(x)$ at several points close to x_k
  - compute a trial step s_k if it is good, move to $x_{k+1} = x_k + s_k$, if not stay put (x_k)

Gradient-descent methods

Both solutions below are getting the gradient information from finite-difference approximation and are approximate Hessian

- BFGS:
  - combine *secant method* and *Broyden update* to get H_k matrix
  - inverse it with *Cholesky factorisation*
- NEWTON: Preconditioned truncated Newton
  - use *conjuguate gradient* method to define *conjuguate direction* that annulate the gradient in the quadratic approximation



Not discussed here

Parallelisation:

Iterative methods means open loops: iteration $k+1$ needs result of k to be defined

- ➔ Cannot distribute computations
- ➔ Introduce new **stopping criteria**: maximal number of iterations / function calls (avoid infinite loop)
- ➔ **Multistart**: use one ressource by starting point and let the optimisations work independently
  - It will use as many ressources as provided
  - Changing starting point will make the result (a bit more) robust toward local minimum sensitivity

Constrains:

Some algorithms can natively handle constrains. In general the possible solutions might depend on

- the kind of constrain : equality / inequality / both constrains ?
- the nature of the problem : linear ? convex ?

There are several strategies to deal with these:

- rewrite the problem: when possible, introduce the constrains into the objectif
- investigate the **dual** problem: write the **Lagrangian**

$$L(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x),$$

  $\lambda \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^p$ being called **Lagrange's multipliers**
- use the **Augmented Lagrangian method**

$$L_c(x, \lambda) = L(x, \lambda) + \frac{c}{2}||h(x)||^2,$$

  where $c \in \mathbb{R}^+$ (or general **penalty method** for inequality constrains as well).

Other possibilities

SO stategy not based on Newton:

Heuristic: family of methods not supported by rigorous theoretical framework but might be a good solution anyhow [7]

- Neighborhood local search
- Variable neighborhood search
- Simulated annealing

Other solutions might include surrogate models to get helpfull information, see next slides

Efficient Global Optimisation

Combining surrogate models and optimisation techniques [8]

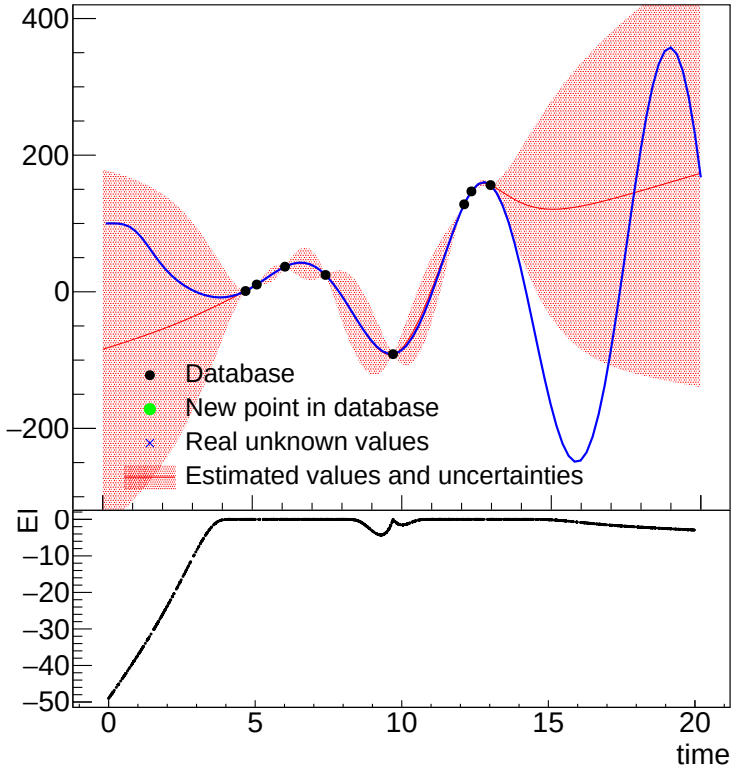
Efficient Global Optimisation (EGO)

From a small database (here 8 points)

- Construct a kriging model [9]
- Compute the Expected Improvement with the kriging model
  - ➔ using genetic algorithm to get the minimum t^*
- Compute the real new value with the code at t^*
- Reconstruct the kriging on the database + t^*
- Continue this process iteratively...

Ongoing work to parallelise this process

Typically used for very time/cpu consuming code. Investigating different approaches to estimate the new points.



Efficient Global Optimisation

Combining surrogate models and optimisation techniques [8]

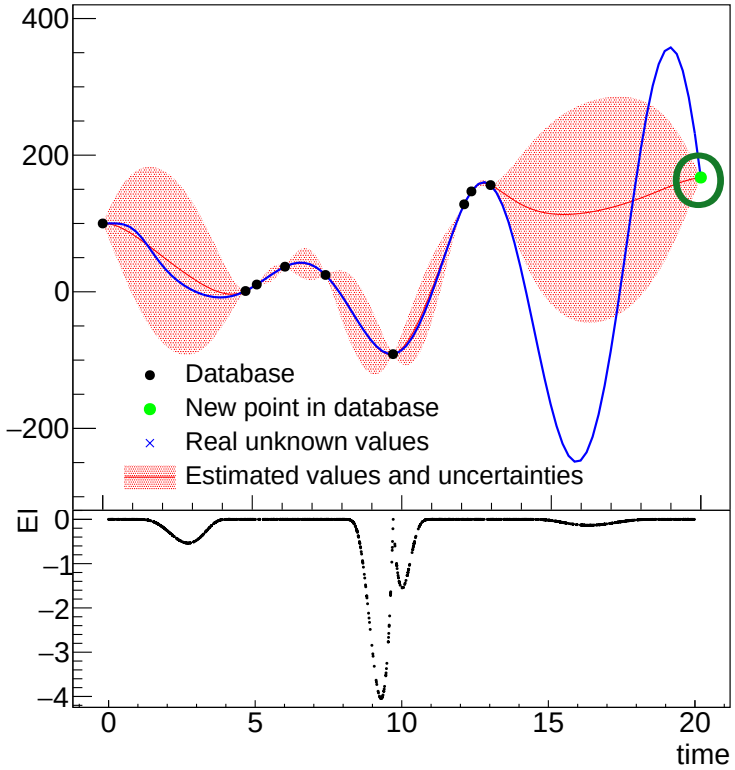
Efficient Global Optimisation (EGO)

From a small database (here 8 points)

- Construct a kriging model [9]
- Compute the Expected Improvement with the kriging model
  - ➔ using genetic algorithm to get the minimum t^*
- Compute the real new value with the code at t^*
- Reconstruct the kriging on the database + t^*
- Continue this process iteratively...

Ongoing work to parallelise this process

Typically used for very time/cpu consuming code. Investigating different approaches to estimate the new points.



Efficient Global Optimisation

Combining surrogate models and optimisation techniques [8]

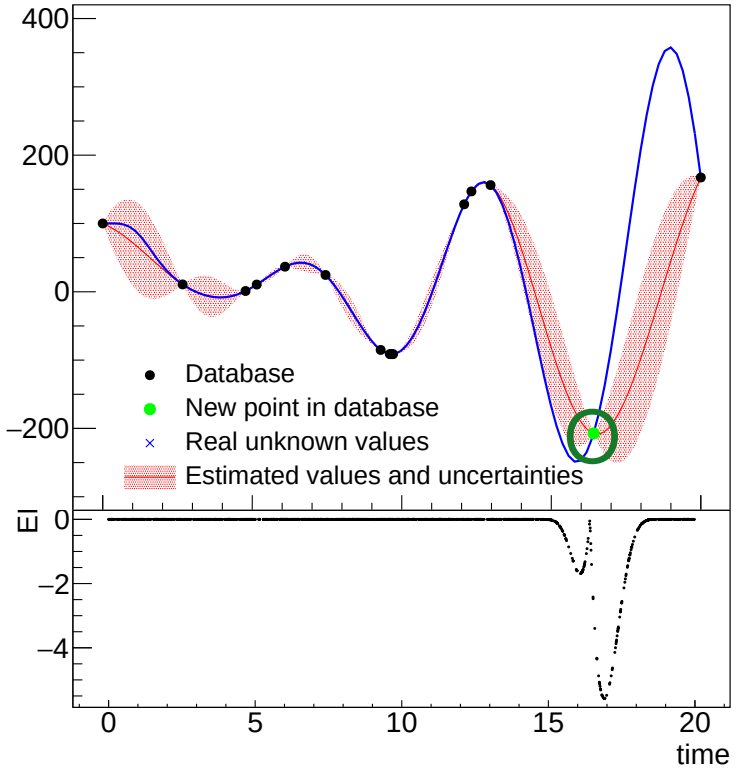
Efficient Global Optimisation (EGO)

From a small database (here 8 points)

- Construct a kriging model [9]
- Compute the Expected Improvement with the kriging model
  - ➔ using genetic algorithm to get the minimum t^*
- Compute the real new value with the code at t^*
- Reconstruct the kriging on the database + t^*
- Continue this process iteratively...

Ongoing work to parallelise this process

Typically used for very time/cpu consuming code. Investigating different approaches to estimate the new points.



Efficient Global Optimisation

Combining surrogate models and optimisation techniques [8]

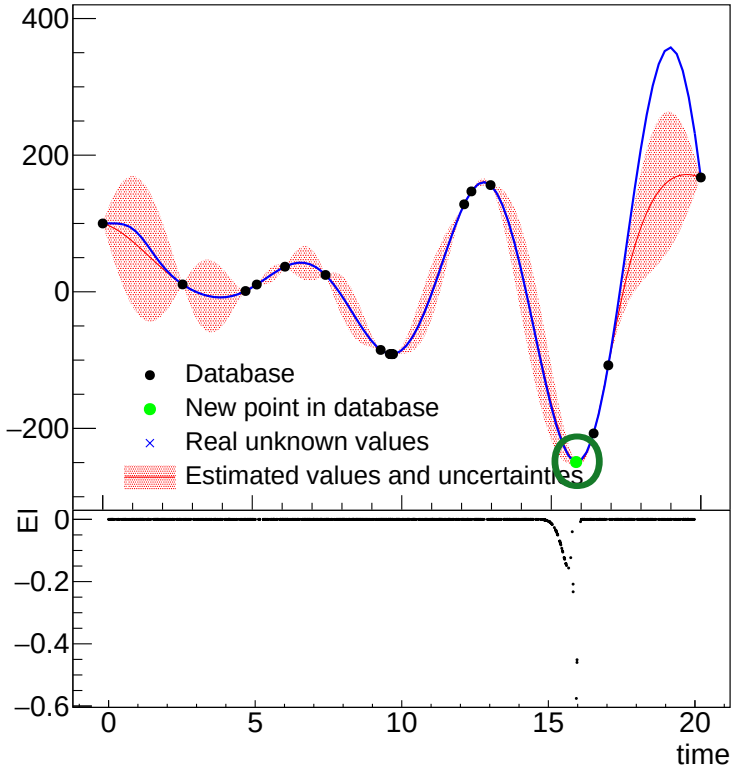
Efficient Global Optimisation (EGO)

From a small database (here 8 points)

- Construct a kriging model [9]
- Compute the Expected Improvement with the kriging model
  - ➙ using genetic algorithm to get the minimum t^*
- Compute the real new value with the code at t^*
- Reconstruct the kriging on the database + t^*
- Continue this process iteratively...

Ongoing work to parallelise this process

Typically used for very time/cpu consuming code. Investigating different approaches to estimate the new points.



Outline

Introduction
- Purpose
- Mathematical model
- Vocabulary

Mono-criteria problems
- Rosenbrock function, a classical example
- Historical approach
- Examples with current algorithms
- Heuristic / model based techniques

Multi-objectives problems
- Vocabulary
- Example, in a nutshell

Conclusion

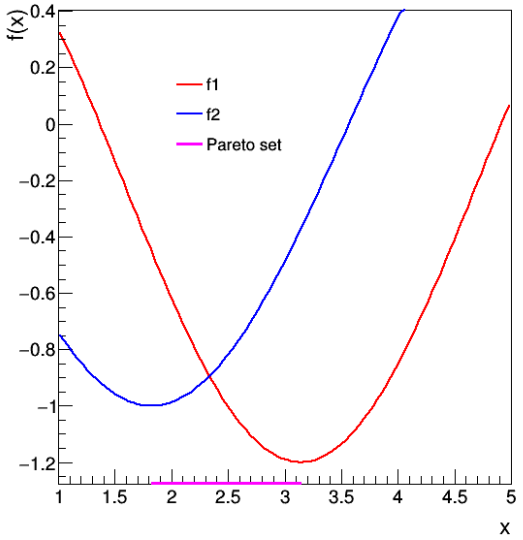
Multi-objective optimisation vocabulary

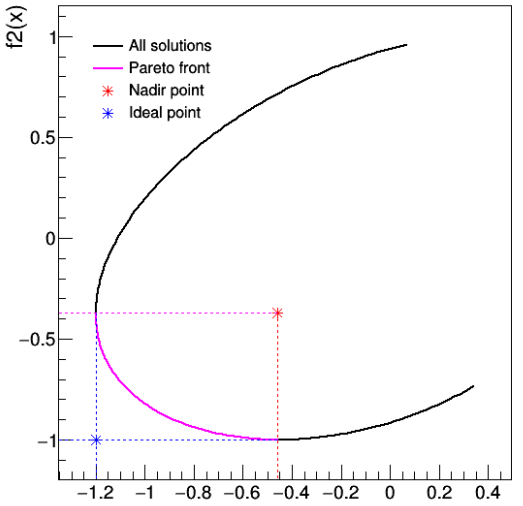
- When several criteria must be taken into consideration, the solution kept is always a **compromise**.
- Multi-criteria optimisation consists in finding a set of "**acceptable**" solutions according to criteria and constraints posed.
- In the diagram at top opposite, we search the values of x which optimize the criteria f_1 and f_2 .
- Let x_1 and x_2 the minimum of f_1 and f_2 , respectively. For all $x_i < x_2$, if $x_2 > x_j > x_i$ then

$$f_1(x_j) < f_1(x_i) \text{ and } f_2(x_j) < f_2(x_i)$$

  Also true for $x_i > x_1$ if $x_1 < x_j < x_i$: x_j **dominates** x_i
- However for $x_2 < x_i < x_1$ no value of x_j does improve both criteria simultaneously (previous equation).
- Compromise solutions are to be found in the area $x_2 < x < x_1$ called the Pareto zone ($\mathcal{P}$).
- The group of corresponding solutions in the space of criteria (listed below opposite) is called the Pareto front. They are said to be **non-dominated**: if $x_a, x_b \in \mathcal{P}$ then

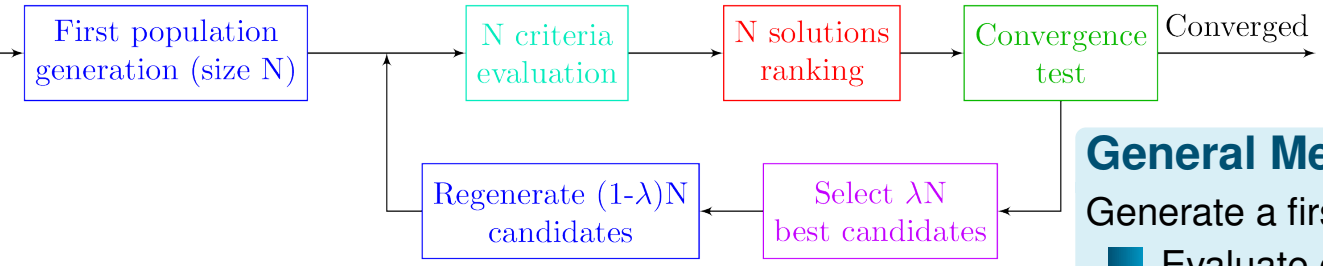
$$f_1(x_a) < f_1(x_b) \text{ and } f_2(x_a) < f_2(x_b) \text{ is impossible}$$





Example of multi-objective optimisation

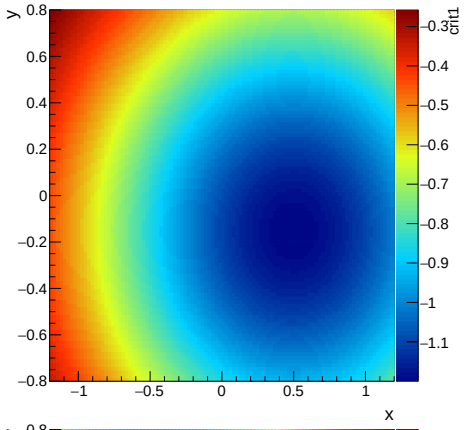
With two antagonist criteria (crit1, crit2) depending both on x and y

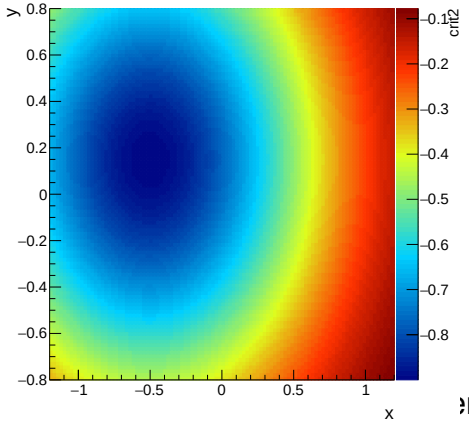


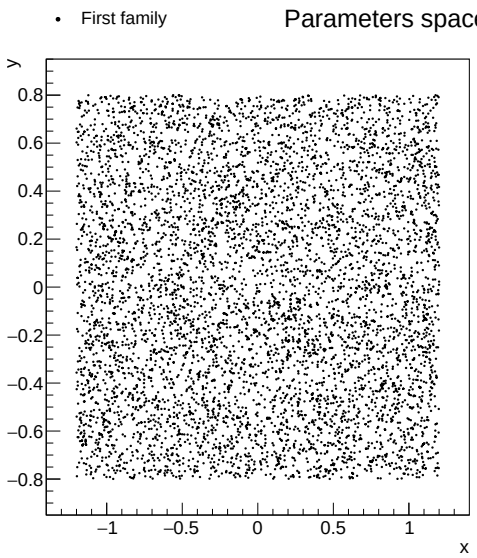
General Methodology [10]

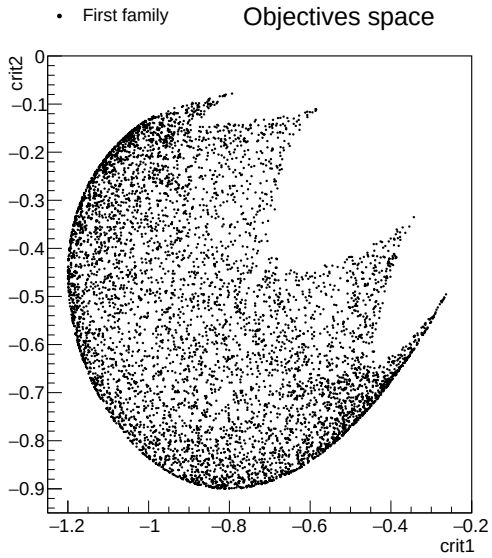
Generate a first family of N people and then

- Evaluate criteria for all people
- Rank them according to criteria
- Test convergence.
  - Converged: stop
- Create new people from the best λN
- Start all over



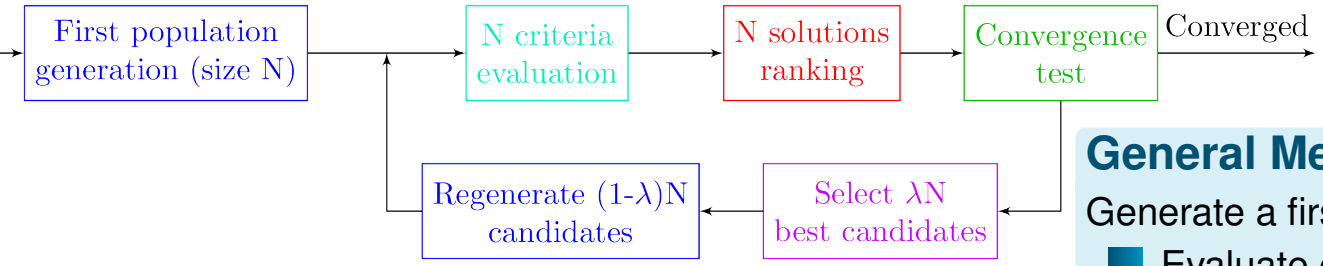






Example of multi-objective optimisation

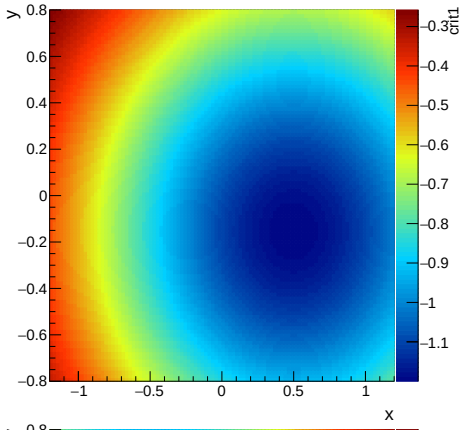
With two antagonist criteria (crit1, crit2) depending both on x and y

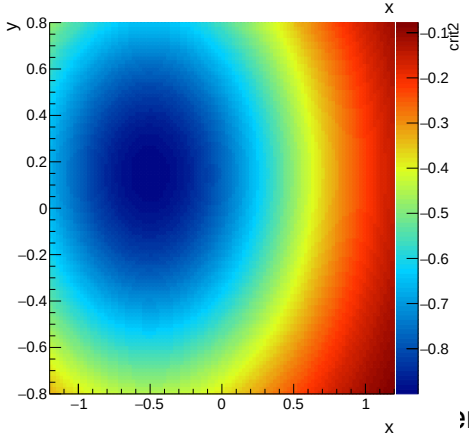


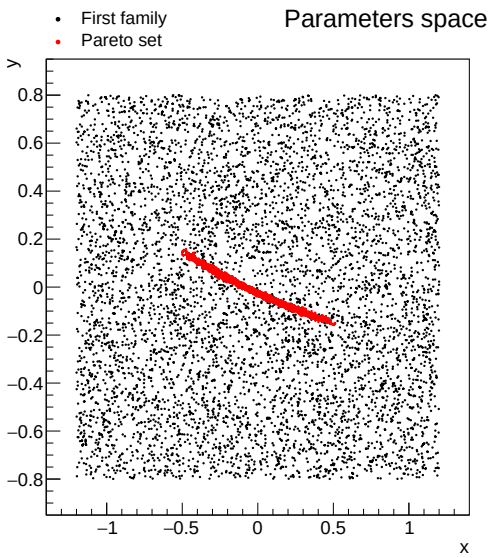
General Methodology [10]

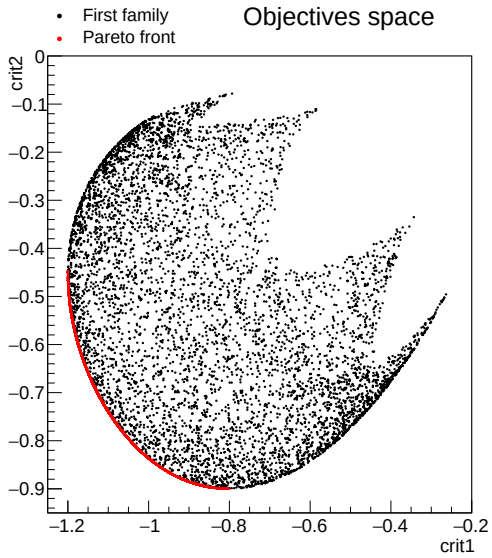
Generate a first family of N people and then

- Evaluate criteria for all people
- Rank them according to criteria
- Test convergence.
  - Converged: stop
- Create new people from the best λN
- Start all over









Outline

Introduction
- Purpose
- Mathematical model
- Vocabulary

Mono-criteria problems
- Rosenbrock function, a classical example
- Historical approach
- Examples with current algorithms
- Heuristic / model based techniques

Multi-objectives problems
- Vocabulary
- Example, in a nutshell

Conclusion

Summary

Optimisation is a complex problematic with, unfortunatly no magical solutions to work with

It depends a lot on

- Your aim:
  - Single / Multi objective case ?
  - constrains or not
  - deterministic / stochastic / combinatory
- Your code / function:
  - what you know about its behaviour (linear, smooth, noised. . .)
  - the time it needs to run
  - the information it provides (precision, gradient, more ?)

You might need to restart oftenly, so it might be wise to always save the results of a run for bookkeeping

References I

- Michel Bierlaire.
  Introduction à l'optimisation différentiable.
  PPUR presses polytechniques, 2006.
- Edwin KP Chong and Stanislaw H Zak.
  An introduction to optimization, volume 76.
  John Wiley & Sons, 2013.
- Carl T Kelley.
  Iterative methods for linear and nonlinear equations, volume 16.
  Siam, 1995.
- Ulrich Drepper.
  What every programmer should know about memory.
  Red Hat, Inc, 11:2007, 2007.
- HoHo Rosenbrock.
  An automatic method for finding the greatest or least value of a function.
  The Computer Journal, 3(3):175–184, 1960.
- Steven G. Johnson.
  The nlopt nonlinear-optimization package, 2008.
  http://ab-initio.mit.edu/nlopt.
- Michel Bierlaire.
  Optimization: principles and algorithms.
  Number BOOK. EPFL Press, 2015.

References II

- Donald R Jones, Matthias Schonlau, and William J Welch.
  Efficient global optimization of expensive black-box functions.
  Journal of Global optimization, 13(4):455–492, 1998.
- G. Matheron.
  La théorie des variables régionalisées, et ses applications.
  Fasicule 5 in Les Cahiers du Centre de Morphologie Mathématique de Fontainebleau, 1970.
- J-B. Blanchard.
  Methodological reference guide for uranie v3.11.0.
  Technical report, 2017.
  Updated with every new release.

Commissariat à l'énergie atomique et aux énergies alternatives
Centre de Saclay | 91191 Gif-sur-Yvette Cedex
T. +33 (0)1 69 08 73 20 | F. +33 (0)1 69 08 68 86

Direction de l'énergie nucléaire
Département de modélisation des systèmes et structures
Service de Thermohydraulique et de mécanique des fluides

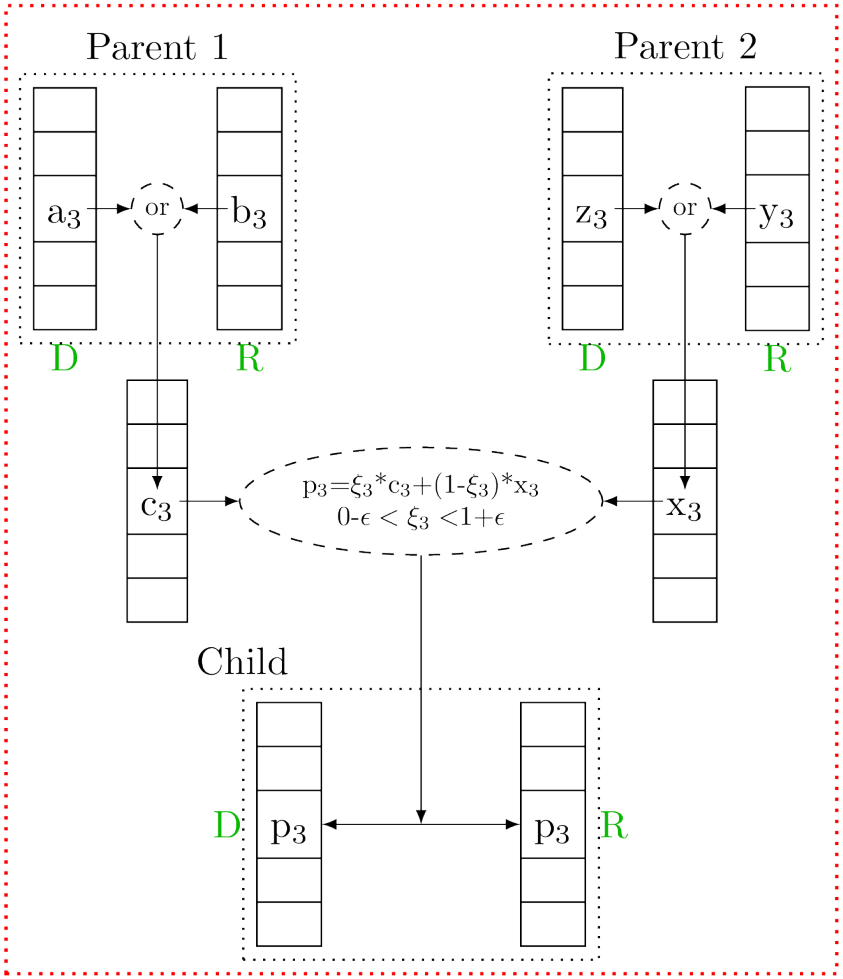
Etablissement public à caractère industriel et commercial | R.C.S Paris B 775 685 019

Backup outline

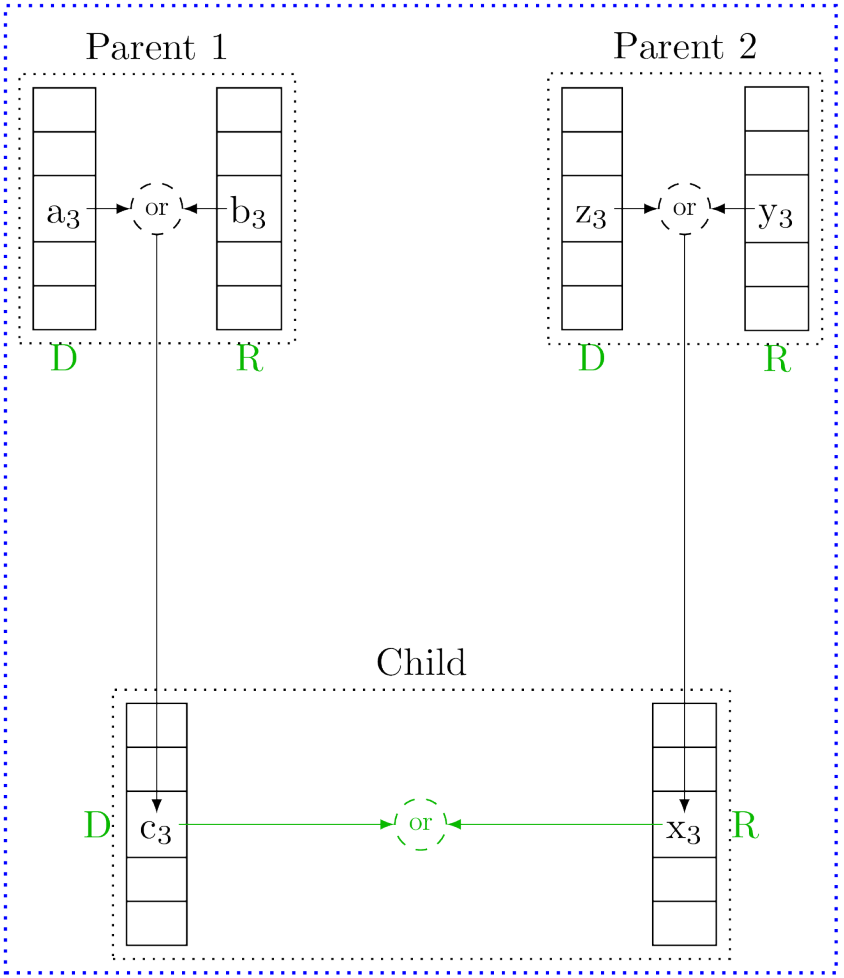
Backup

Vizir genetic in a nutshell
The expected improvement definition

Genetic evolution in a drawing



Homozygote (proba Ho)



Heterozygote (proba (1-Ho))

+ possible mutation
(proba M)

The expected improvement definition

$$E[I(\mathbf{x})] = (f_{min} - \hat{y}(\mathbf{x}))\Phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{\sigma}(\mathbf{x})}\right) + \hat{\sigma}(\mathbf{x})\phi\left(\frac{f_{min} - \hat{y}(\mathbf{x})}{\hat{\sigma}(\mathbf{x})}\right)$$

$\phi(.)$ and $\Phi(.)$ are respectively the standard normal density and its cumulative distribution.