A review of optimization methods for the identification of complex models

Rodolphe Le Riche

A review of optimization methods for the identification of complex models

Rodolphe Le Riche
CNRS and Ecole des Mines de Saint-Etienne

presentation given as part of the MEXICO meeting on optimization, La Rochelle, Fr, Nov. 2014
http://reseau-mexico.fr

Goal of the presentation

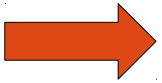
- Why numerical optimization may be useful to the scientist making models ?

- Provide some concepts on numerical optimization from various backgrounds.

- Cite a few well-known algorithms, but no claim to be exhaustive.

- Optimization beyond the algorithm run.

Talk outline

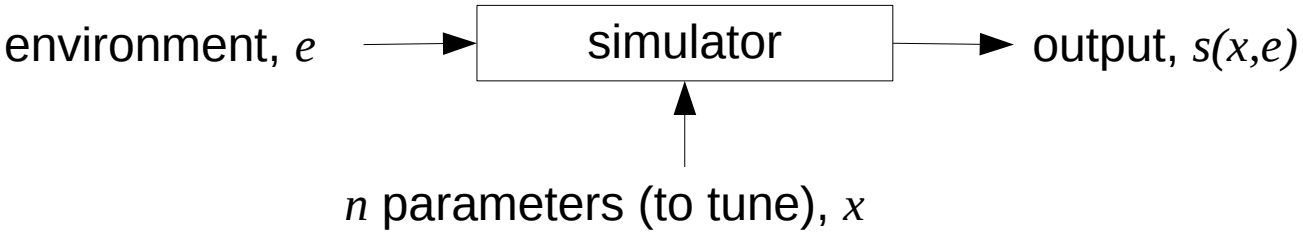


- Why numerical optimization may be useful to scientists making models ?
- Formulations and problems classification
- Optimization engines
- A few references
- Levenberg-Marquardt algorithm
- CMA-ES
- Concluding remarks
- (other algorithms)

Why numerical optimization may be useful to scientists making models ?

Context of model parameter identification

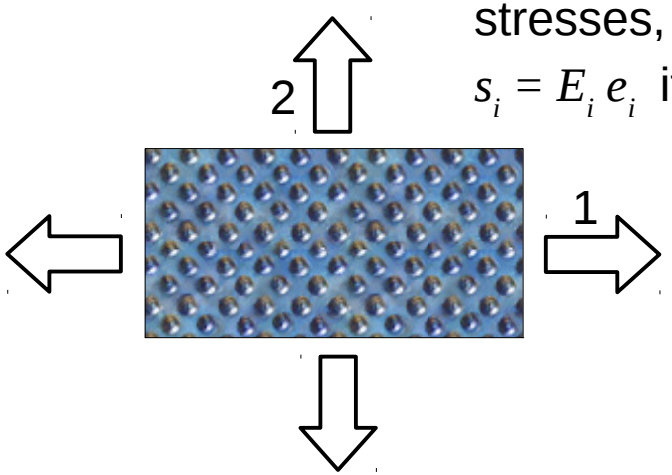
Ideally, scientists know the physical meaning of the parameters that are in the models and know how to tune them based on specific experiments

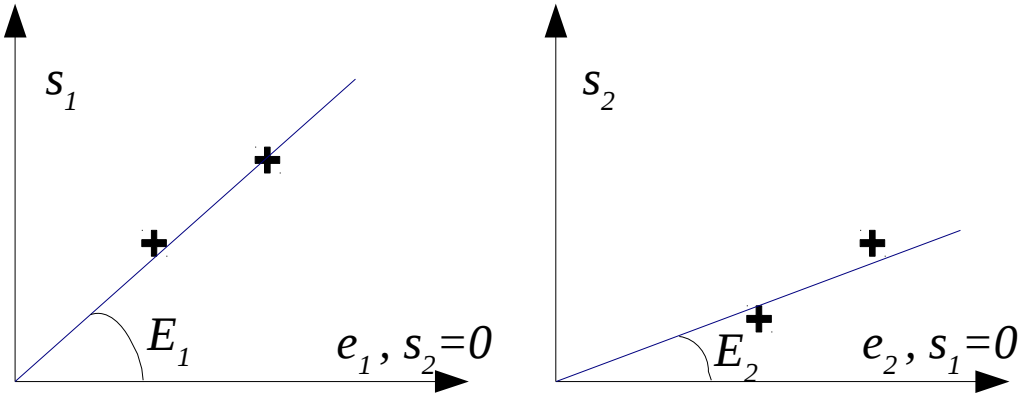


1 parameter directly read with 1 experiment, separation of parameter effects
Example in mechanics, 2 elastic constants to set = slopes of stress-strain curves

stresses, s, strains, e, x = elastic constants = (E_1,E_2) ,

$s_i = E_i\, e_i$ if $s_j=0$

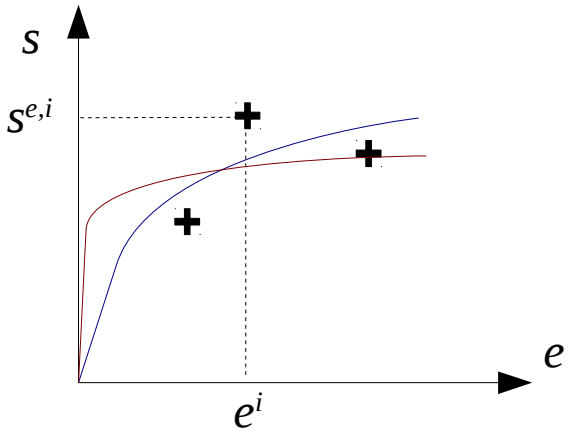




Why numerical optimization may be useful to models ?

Context of model parameter identification

But the effects of parameters are often difficult to separate



Example : behavior with added contributions (linear and non linear stresses)

$s = Ee + Ke^p$, $x = (E,K,p)$

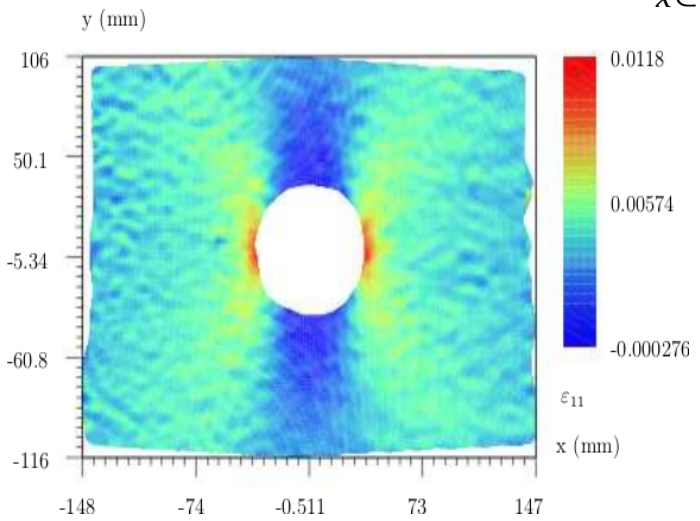
→ formulate an optimization problem : minimize a distance, f, between experiments and simulations by changing x

$$\min_{x \in S} f(x)$$

Classically, $f(x) = \frac{1}{2}\sum_{i=1}^{m}\left(s^{e,i} - s(x, e^i)\right)^2 = \frac{1}{2}\left\|s^e - s(x)\right\|^2$

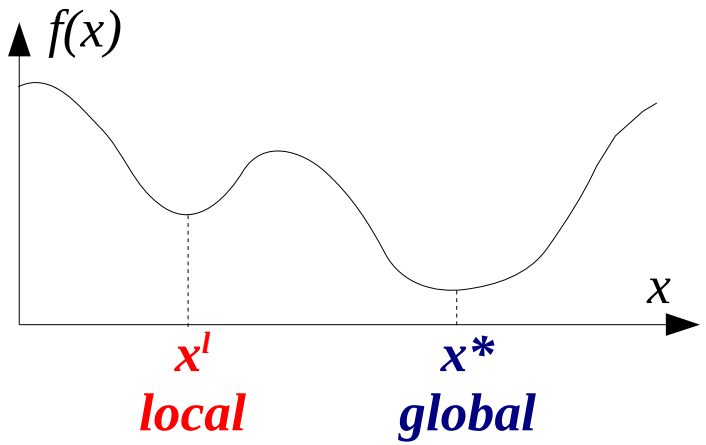
Why numerical optimization may be useful to models ?
Context of model parameter identification

To try to solve $\min_{x \in S} f(x)$ a computer will be helpful



1) because S will typically be a large space, e.g., R^n with $n \geq 2$

Expl : Identification of 4 elastic constants with 1 experiment (Silva et al., *Identification of material properties using FEMU : application to the open hole tensile test*, Appl. Mech & Matls, 2007)

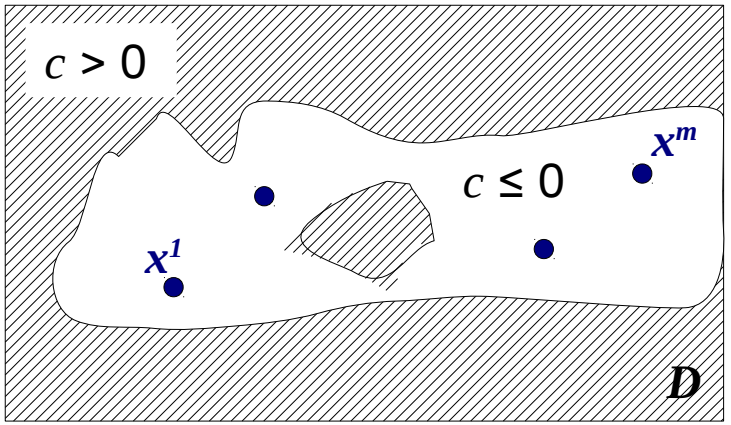


2) because understanding where f is smallest may be complex, i.e., beyond intuition, because of multimodality.

Expl : Identification of a material non linear law, cf. red and blue solutions on previous slide (+ Le Riche and Guyon, *Least Squares Parameter Estimation and the Levenberg-Marquardt Algorithm : Deterministic Analysis, Sensitivities and Numerical Experiments*, TR, 1999)

Why numerical optimization may be useful to models ?

Context of DoE or optimal placement



Find the location of points (experiments, sensors, actuators, …) that offers best covering while satisfying some constraints (e.g., bounds on parameters, sensors outside protected regions, …)

$$\min_{\substack{x \in S \\ g(x) \le 0}} f(x)$$

where

$$S \equiv D^m \subset \mathbb{R}^{m \times d}$$

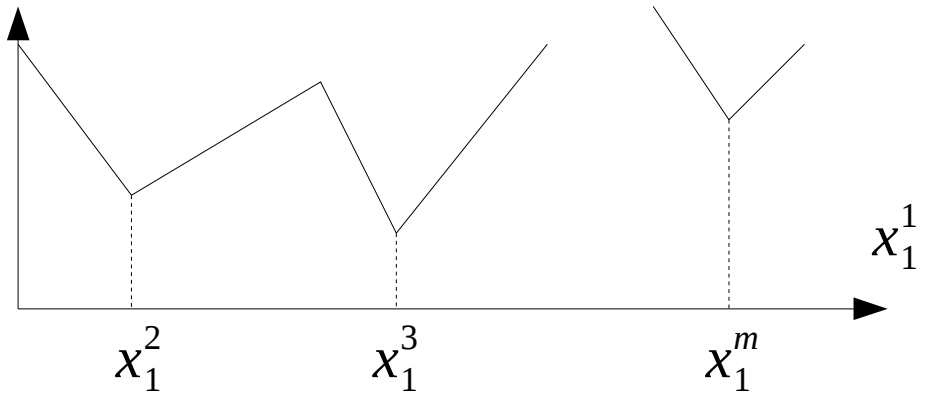
$$x = \left(x_1^1, \ldots, x_d^1, x_1^2, \ldots, x_d^m\right)$$

$$f(x) = -\min_{i \neq j} \text{dist}\left(x^i, x^j\right)$$

$$g(x) = \max_{i=1,m} c\left(x^i\right)$$

This family of problems is typically high dimensional and multi-modal, not well-suited to human intuition

$$\min_{i \neq 1} \text{dist}\left(x^1, x^i\right)$$



Talk outline

- Why numerical optimization may be useful to scientists making models ?
- Formulations and problems classification
- Optimization engines
- A few references
- Levenberg-Marquardt algorithm
- CMA-ES
- Concluding remarks
- (other algorithms)

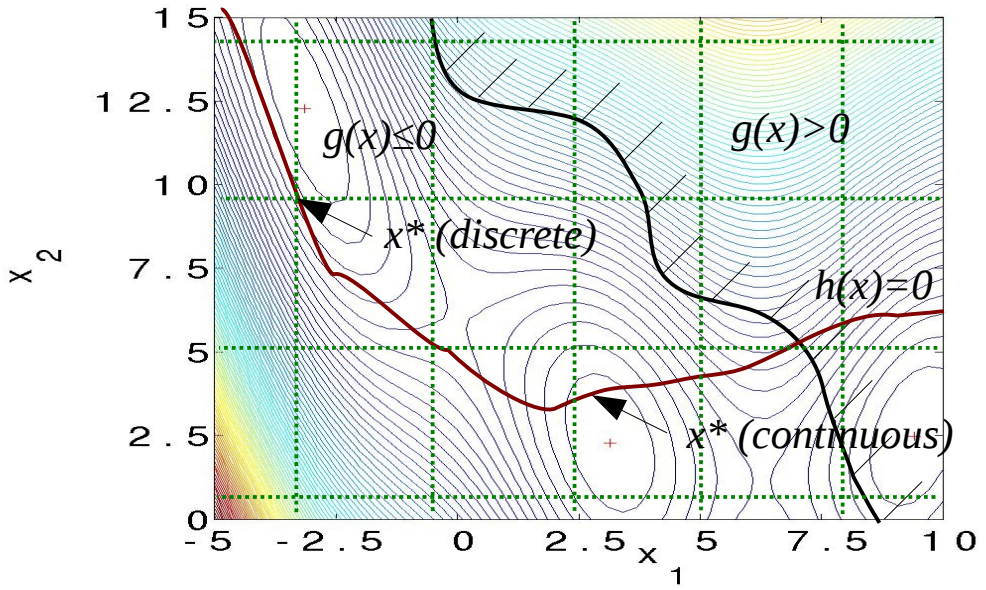
Formulations

x , n optimization variables
S , search space , $x \in S$
f , objective or cost function to minimize, $f : S \rightarrow \mathbb{R}$
g , inequality constraint, $g(x) \leq 0$
h , equality constraint, $h(x) = 0$

$$\min_{x \in S} f(x)$$

such that $g(x) \leq 0$
$h(x) = 0$

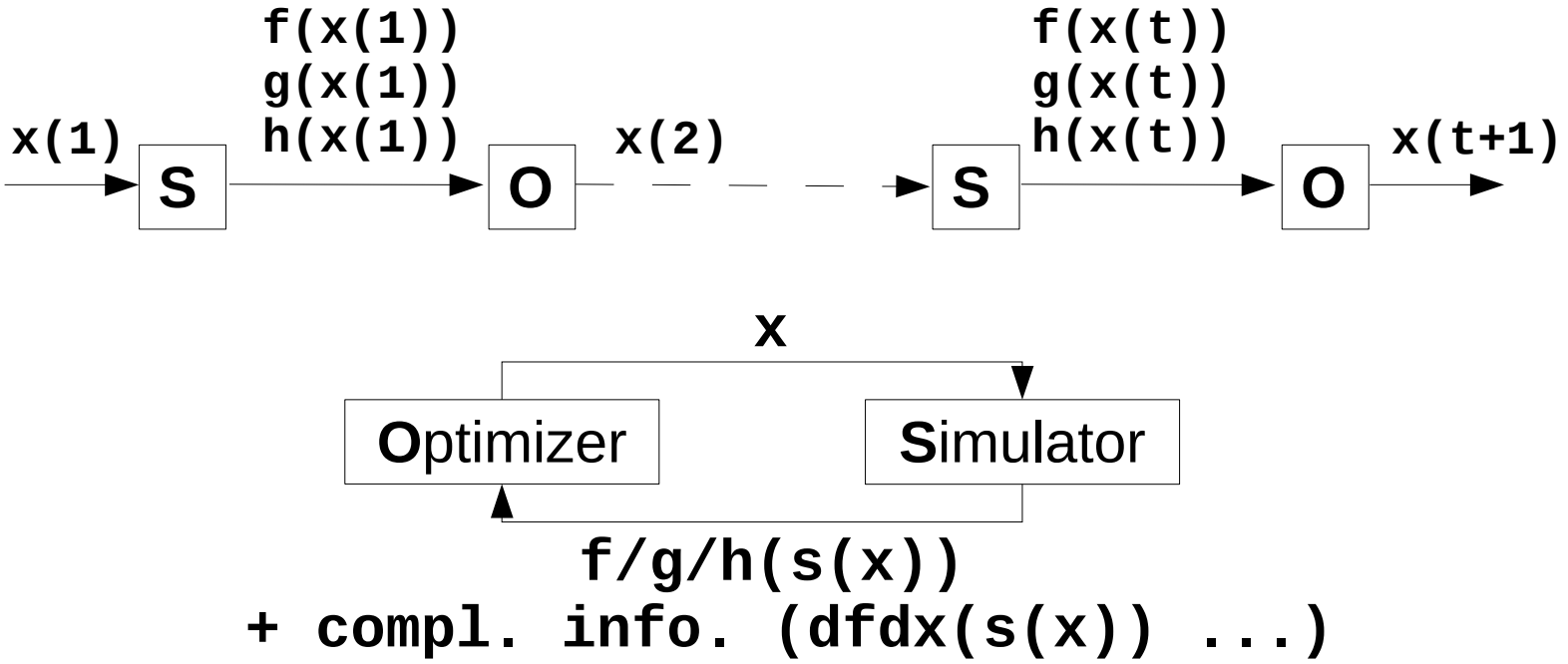
!!! the choice of x affects the mathematical properties of f, g, h and orients the selection of opt. algorithm



Expl : Displacements (f) are non linear functions of thicknesses (x = thickness) but linear functions of x = 1/thickness (for statistically determinate structures)

Optimization algorithms

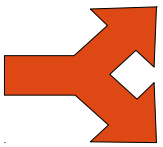
An optimizer is an algorithm that iteratively proposes new x's based on past trials in order to approximate the solution to the optimization problem



The ***cost*** of the optimization is the number of calls to the simulator s (usually = number of calls to f)

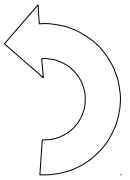
Problems classification

compose the following switches to find the pb. category and the algo



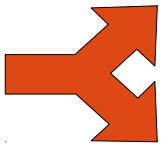
unconstrained (no g and h)

constrained



Lagrangians, penalizations, e.g.,

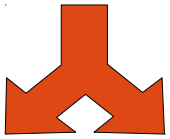
$$\min_{x \in S} \; f(x) + p \times max^2(0, g(x))$$



mono-objective, $\min_{x \in S} f(x)$

multi-objective, $\min_{x \in S} f_1(x)$ & $\ldots f_m(x)$

MO pbs typically have set of (Pareto) solutions, cf. Dimo's talk



uni-modal
of which convex,
of which linear
→ local optimization

multi-modal
→ global optimization

continuous variables

$S \subset \mathbb{R}^n$

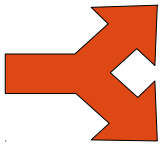
integer programming

$S \subset \mathbb{N}^n$

no gradients x(

mixed programming

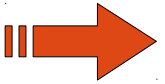
$S \subset \left\{ \mathbb{R}^{n1} \cup \mathbb{N}^{n2} \right\}$



costly simulations → metamodels

non costly simulations

costly = sim. budget < 1000. cf. Victor's talk



number of variables n,
cost of optimizer,
...

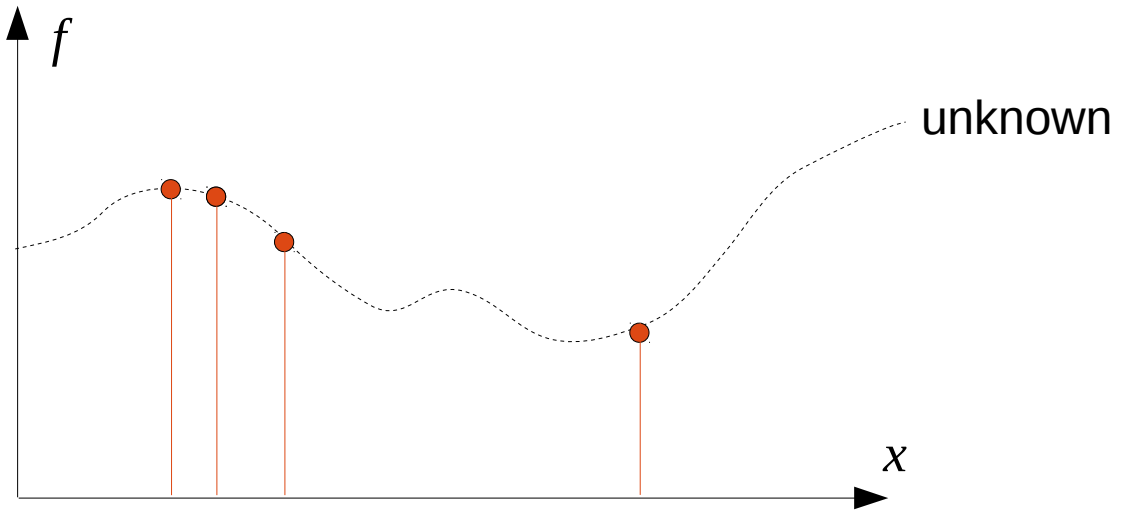
Talk outline

- Why numerical optimization may be useful to scientists making models ?
- Formulations and problems classification
- Optimization engines
- A few references
- Levenberg-Marquardt algorithm
- CMA-ES
- Concluding remarks
- (other algorithms)

Optimization engines

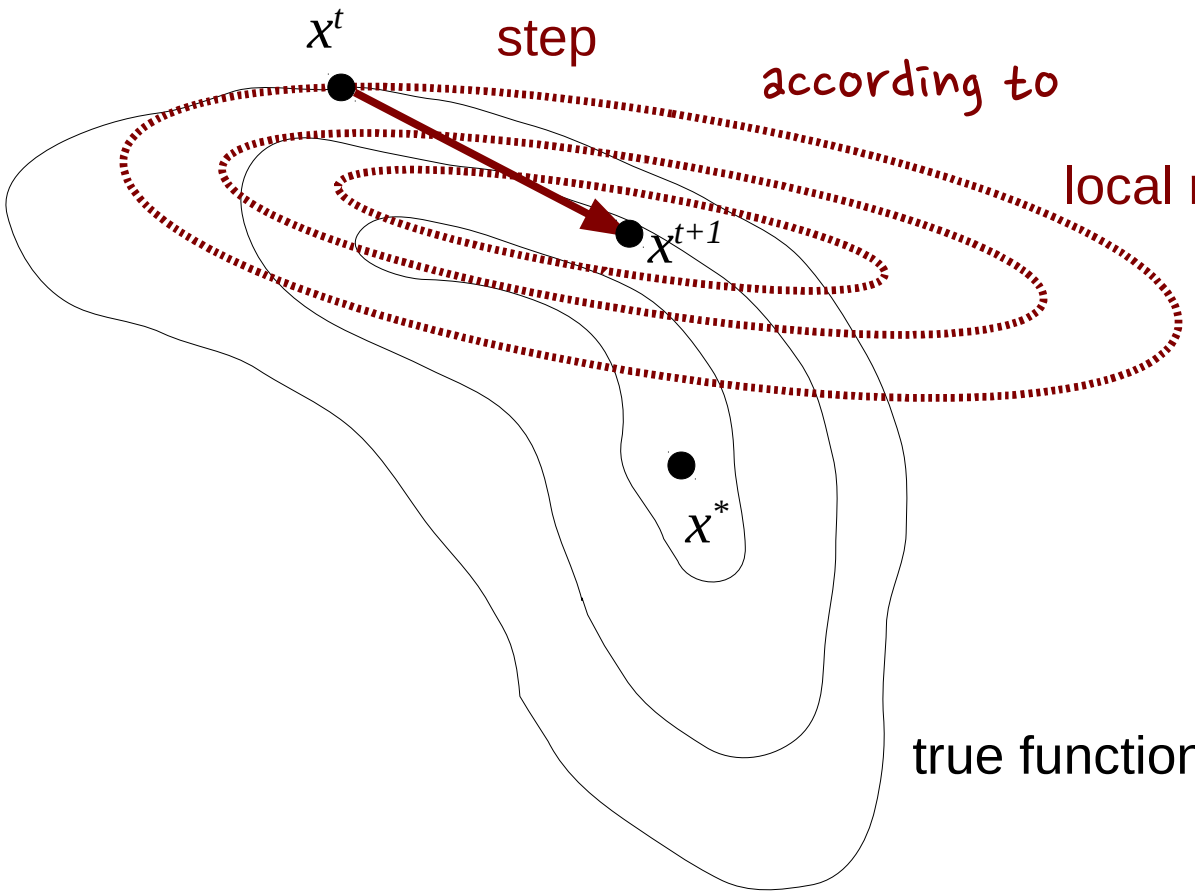
$$\min_{x \in S} \; f(x)$$

and the function's landscape is known through (costly) point measures



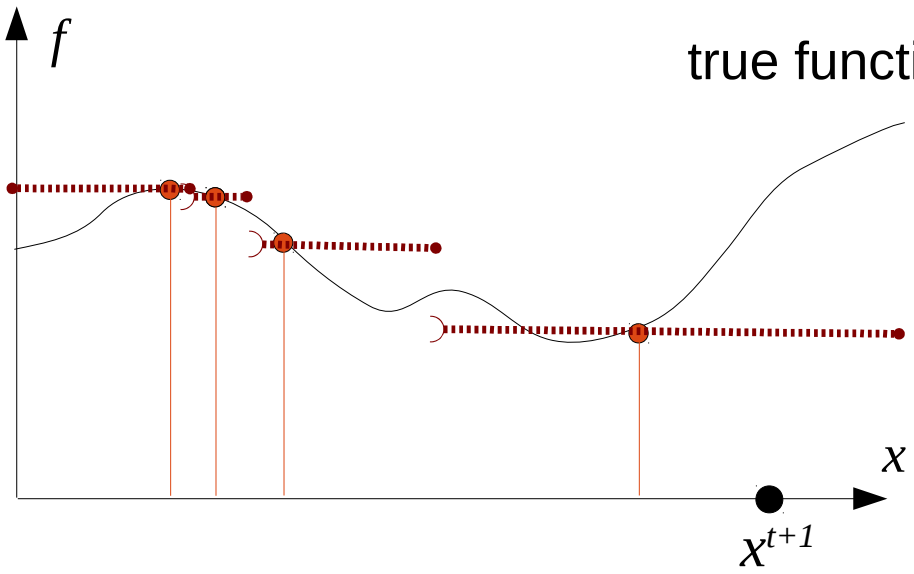
Optimization engine = basic principle to minimise the function

Optimization engines
Local model of the function



- For local optimization.
- Efficient (low number of calls to f) because focuses on neighborhood of current iterate, suitable for n large (>1000).
- At the risk of missing global optimum if f multi-modal.

Global model of the function



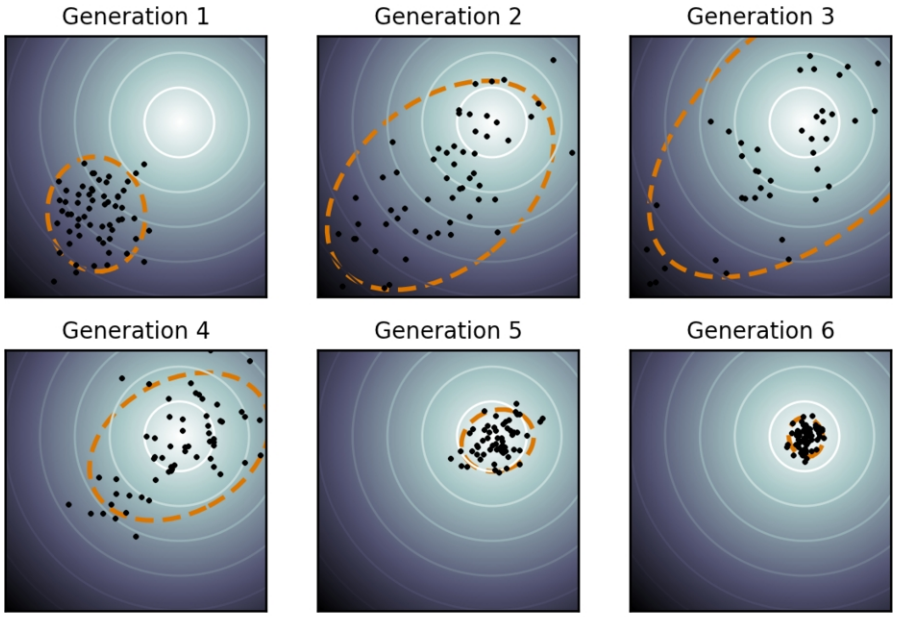
global model of the function
(here = function value of closest known x^i, cf. DIRECT algo.)

The next iterates x^{t+1} are compromises between intensification (search in low areas) and exploration (search in unknown areas).

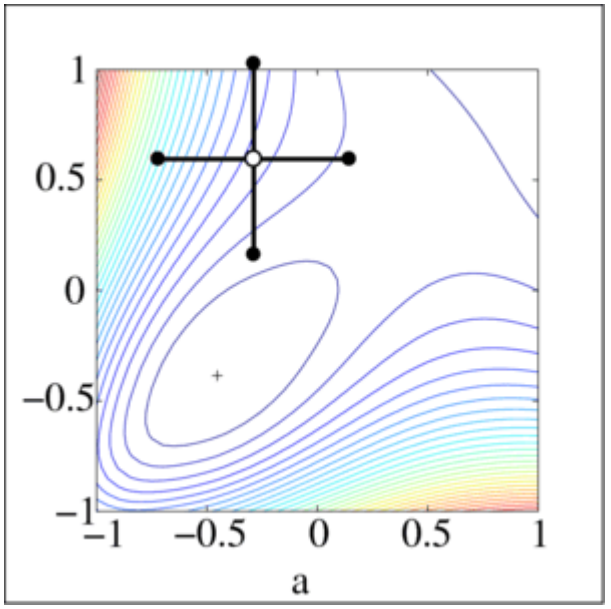
For global optimization.
Efficiency depends on the agreement between the model and the true function but searching the whole S is subject to the « curse of dimensionality ».

Optimization engines
local sampling

sampling in probability
e.g., from a multi-normal density, cf. CMA-ES algorithm



sampling with a geometrical pattern
e.g., from a pattern search algorithm



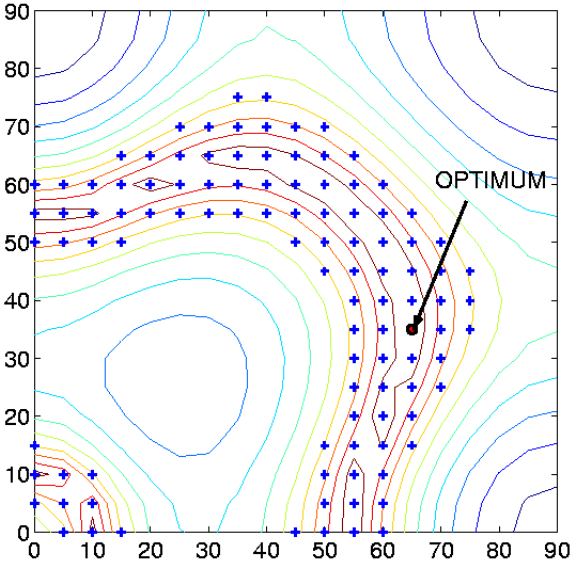
- Sampling from a current point with the notion of step.
- Sampling → methods do not require f continuity or regularity.
- Sampling in probability → S can be continuous, or discrete or mixed.
- May converge locally.
- Efficient and more suitable to large n (>10) than global sampling methods.

(both graphs from wikipedia, cma-es – left – and pattern search – right –)

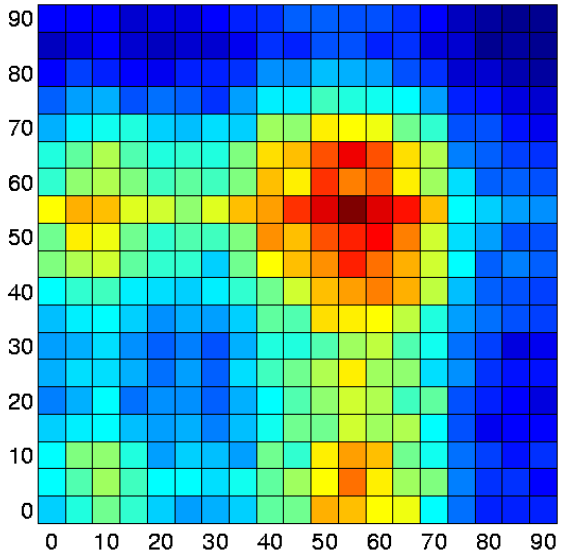
Optimization engines
global sampling

The next x iterates can be sampled anywhere in S from a density learned on observed points. Expl. Estimation of Distribution Algorithms (illustration below), Branch and Bound, … .

f(x) and selected points



$$p_X(x) = \prod_{i=1}^{n} p_i(x_i)$$



(from Grosset et al., *A double distribution statistical algorithm for composite laminate optimization*, SMO, 2006)

- Sampling → methods do not require f continuity or regularity.
- Sampling in probability → S can be continuous, or discrete or mixed.
- May converge globally.
- Typically costly.

Talk outline

- Why numerical optimization may be useful to scientists making models ?
- Formulations and problems classification
- Optimization engines
- A few references
- Levenberg-Marquardt algorithm
- CMA-ES
- Concluding remarks
- (other algorithms)

Optimization engines
the quest for the best optimizer

4 « optimization engines » and hundreds of algorithms that vary in their implementation and composition of these principles.

What is the best optimizer ?

If you have no a priori on the functions you want to optimize, there is no best optimizer (*No Free Lunch Theorem in optimization*, D. Wolpert, 1997).

But there are always a priori's on the considered f's (no « needle in the haystack », class of problems such as identification of a given physical model, …).

→ The best algorithm depends on the considered f's and the allocated computational budget.

A few references (1)

No claim to be exhaustive, fair, well-informed, …
just a few references to be practical.

$S \subset \mathbb{R}^n$, **continuous optimization**

Constrained linear programming
→ *Simplex algorithm*, Dantzig, 1947.
$\min_{x \in \mathbb{R}^n} \; c^T x$ such that $Ax - b \leq 0$
Try to achieve such formulation because n and the number of constraints can be very large (thousands) and the resolution almost immediate.

Unconstrained, uni-modal, smooth function
→ *BFGS* algorithm, (Broyden-Fletcher-Goldfarb-Shanno, 1970)
A quasi-Newton method requiring the calculation of gradient of f.
→ *Levenberg-Marquardt* (Levenberg 1944, Marquardt 63)
A quasi-Newton method specialized for non-linear least-squares (model identification!). Gradient of f to calculate.

Unconstrained, moderately multi-modal function without derivative
→ *NEWUOA* algorithm (Powell, 2004)
A trust-region algo. with quadratic approximation, no function derivative, sparse sampling (~$2n+1$ interpolation points), n up to 160 variables.

A few references (2)

$S \subset \mathbb{R}^n$, **continuous optimization**

(– Continued – **unconstrained, moderately multi-modal function without derivative)**

→ *Nelder-Mead* algorithm (1965), robust and simple, may stop on non-stationary points, up to n~30. Cf. *A constrained, globalized and bounded Nelder-Mead algorithm for eng. optim.*, Luersen et al., SMO 2003.

Unconstrained, multi-modal function without derivative

→ *CMA-ES* algorithm (Hansen et al., from 1996 on)
may be the *BIPOP-CMA-ES* variant for restarts. up to n~100-1000.

Unconstrained, multi-modal function without derivative and expensive functions

→ *CMA-ES* with surrogate for moderately expensive functions (may be the *saACM-ES* variant by Loshchilov et al., 2013)
→ *EGO* (Jones, 1998) for very expensive functions (budget < 1000).

Constrained non linear continuous optimization

→ *Sequential Quadratic Programming*
a trust region iterative method with a series of Quadratic Problems solved.
Up to n~100-1000.

A few references (3)

$S \subset \mathbb{N}^n$, **integer programming**

linear integer programming $\min_{x \in \mathbb{Z}^n} c^T x$ such that $Ax - b \leq 0$

→ *cutting plane* or *branch and bound* methods
solutions in polynomial time, very large number of variables and constraints possible. As in continuous linear programming, work on the formulation to obtain such a problem, if possible.
Cf. *Introduction to operations research,* Hillier & Lieberman, 1990.

non linear integer programming

→ Univariate Marginal Density Algorithm, *UMDA*, Baluja 1994 as PBIL, Mühlenbein 1996. For its simplicity.
→ *MIMIC* (Mutual Information Maximizing Input Clustering) algorithm : De Bonnet, Isbell and Viola, 1997. Accounts for variables couplings.
→ other problem specific metaheuristics in particular for combinatorial problems, e.g., E*volutionary Algorithms*, (cf. *Introduction to evolutionary computing*, Eiben and Smith, 2003).
EAs = Possible line of approach for mixed problems, $S \subset \left\{ \mathbb{R}^{n1} \cup \mathbb{N}^{n2} \right\}$

All these algorithms are expensive (1000-10^6 calls to f).

A few references (4)

Constrained mixed non linear optimization , $S \subset \mathbb{R}^{n1} \cup \mathbb{N}^{n2}$

box constraints $x^{\min} \le x \le x^{\max}$
are simple to handle inside the algorithms, either by satisfying the Karush, Kuhn and Tucker optimality conditions, or by re-sampling –> look for versions of the above algorithms with box constraints, e.g., *BFGS-B*.

non linear equality constraints
can be turned into two inequality constraints

$$h(x)=0 \;\Leftrightarrow\; h(x)\le 0 \text{ and } -h(x)\le 0$$

non linear inequality constraints
can be merged into the objective function by a penalty technique, a recommanded form being the *augmented Lagrangian* (Rockaffelar, 1973)

$$L_{\text{aug}}(x,\lambda) = \begin{cases} f(x)-\lambda^2/4\text{r} & \text{if} \quad g(x)\le -\frac{\lambda}{2\text{r}} \\ f(x)+\lambda\, g(x)+r\left[g(x)\right]^2 & \text{else} \end{cases}$$

(index λ and g with i and sum if many constraints)

$\lambda =$ Lag. multipliers, $min_{x\in S} L_{\text{aug}}(x,\lambda^t) \rightarrow x(\lambda^t)$, $\lambda^{t+1} \leftarrow \lambda^t + 2\text{r}\, g(x(\lambda^t))$

Talk outline

- Why numerical optimization may be useful to scientists making models ?
- Formulations and problems classification
- Optimization engines
- A few references
- Levenberg-Marquardt algorithm
- CMA-ES
- Concluding remarks
- (other algorithms)

Non-Linear Least Squares and the Levenberg-Marquardt algorithm

$$\min_{x \in S \subset \mathbb{R}^n} f(x) \quad \text{where} \quad f(x) = \frac{1}{2}\sum_{i=1}^{m} \left(s^{e,i} - s(x, e^i)\right)^2 = \frac{1}{2}\left\| s^e - s(x) \right\|^2$$

Motivation : if $s^{e,i} = s_i(x^*) + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma_i^2)$,

max likelihood measures $\Leftrightarrow \min_{x \in S} f(x)$

Principle : iteratively solve,

$$\begin{cases} \min_{x \in S \subset \mathbb{R}^n} \frac{1}{2}\left\| s(x^k) + \nabla s(x^k)(x - x^k) - s^e \right\|^2 \\ \text{such that } \frac{1}{2}\left\| x - x^k \right\|^2 \leq \frac{1}{2}\delta_k^2 \end{cases}$$

a local quadratic approximation with trust region

Refs. :

Le Riche and Guyon, *Least Squares Parameter Estimation and the Levenberg-Marquardt Algorithm : Deterministic Analysis, Sensitivities and Numerical Experiments*, TR, 1999.

Besson et al., *Object Oriented Programming applied to the Finite Element Method - Part II : Application to Material Behavior*, Europ. J. of Finite Elements, 1998.

Non-Linear Least Squares and the Levenberg-Marquardt algorithm

Optimality conditions (gradient of Lagrangian cancels out) yield :

0. Initialize : $\nu>1$, x^0 , $\lambda_0>0$, $\Delta^{\min}>0$

1. $\left(\nabla s(x^k)^T \nabla s(x^k)+\lambda_k I\right)(x^{k+1}-x^k) = -\nabla s(x^k)^T\left(s(x^k)-s^e\right)$

$$\Rightarrow x^{k+1}$$

2. If $f(x^{k+1}) < f(x^k)$, $\lambda_k \leftarrow \lambda_k/\nu$, $k \leftarrow k+1$, goto 1

   else if $\left\|x^{k+1}-x^k\right\|>\Delta^{\min}$, $\lambda_k \leftarrow \lambda_k \times \nu$, goto 1

   else stop

- LM is specialized for least squares.
- LM is based on a local model of the function. cf. "optim. engines"
- Requires the gradient of the simulator, s, w.r.t. x.
- n up to [100-1000] , but if the gradient is estimated by finite differences ($n+1$ calls to s each time) it becomes costly.

Non-Linear Least Squares and the Levenberg-Marquardt algorithm

- $\nabla f(x) = \nabla s(x)^T \left(s(x) - s^e \right)$

  when progress is rare, $\lambda_k \rightarrow$ large and LM becomes a steepest descent search, with guaranteed progress at the price of small steps.

- $\nabla^2 f(x) = \nabla s(x)^T \nabla s(x) + \sum_{i=1}^{m} \nabla^2 s_i(x) \left(s_i(x) - s_i^e \right) \underset{s \rightarrow s^e}{\rightarrow} \nabla s(x)^T \nabla s(x)$

  i.e., LM algorithm is a quasi-Newton method near the solution when $\lambda_k \rightarrow 0$.

- In case of non identifiability, LM performs a smallest dist. to x^0 regularization (rigorous when s linear in x),

$$\begin{cases} \min\limits_{x \in S} \ \frac{1}{2} \| x - x^0 \|^2 \\ \text{such that} \ \ f(x) = \min\limits_{y \in S} \ f(y) \end{cases}$$

Levenberg-Marquardt post-processing local sensitivity analysis

$$s(x+\Delta x) = s(x)+\nabla s(x)\Delta x+O(\Delta x^2)$$

$$\|s(x+\Delta x)-s(x)\|^2 = \Delta x^T \underbrace{\nabla s(x)^T \nabla s(x)}_{n\times n} \Delta x+O(\Delta x^3)$$

(κ_i^2, v_i) singular values / vectors

$$\frac{\|s(x+\alpha v_i)-s(x)\|^2}{\|\alpha v_i\|^2} = \kappa_i^2 + O(\alpha) \quad , \quad 0<\alpha \ll 1$$

$\Rightarrow$ κ_i^2 sensitivities to parameters

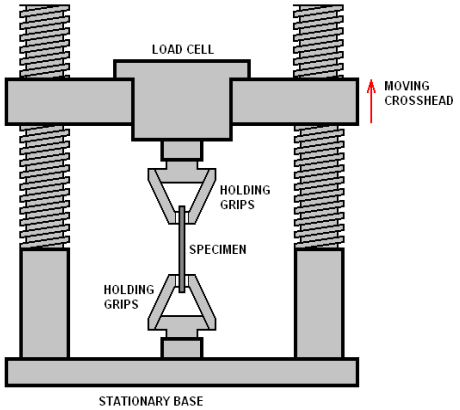
Sensitivity to experiments s^e

$$\approx cond_2\left(\nabla s(x)^T \nabla s(x)\right) = \frac{max_i \kappa_i^2}{min_i \kappa_i^2}$$

Levenberg-Marquardt algorithm
Expl. Identification of material model

Identify 4 non-linear material parameters

2 experiments



$$x = \left(E, K, n, R_i\right)$$

$$\boldsymbol{e} = \left\{\epsilon_1, \ldots, \epsilon_m\right\}$$

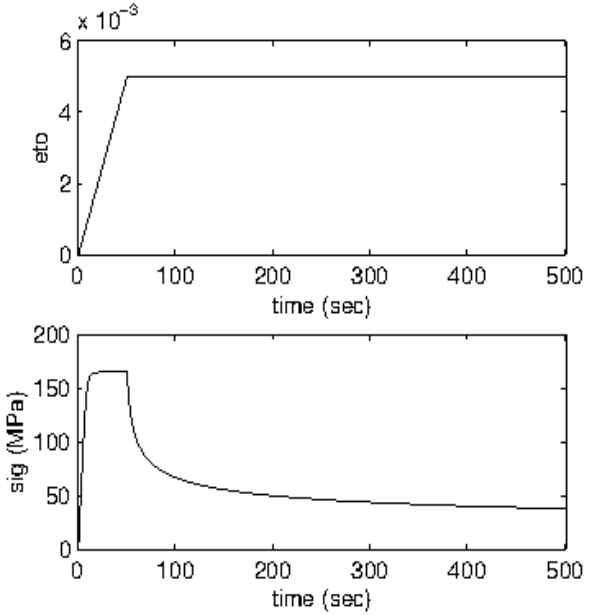
simulator

$$\begin{cases} \dot{p} = \eta\left(\frac{|\sigma| - R_i}{K}\right)^n \\ \dot{\epsilon}_i = \eta\dot{p} \\ \dot{\epsilon}_e = \dot{\epsilon} - \dot{\epsilon}_i \end{cases}$$

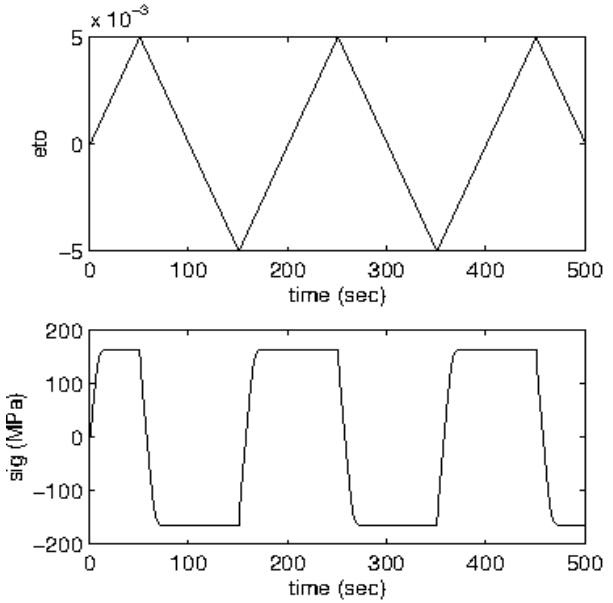
$$\text{and} \begin{cases} \sigma = E\epsilon_e \\ \eta = sign(\dot{\epsilon}) \end{cases}$$

$$\boldsymbol{s}\left(x\right) = \left\{s\left(x, \epsilon_1\right), \ldots, s\left(x, \epsilon_m\right)\right\}$$

relaxation



fatigue



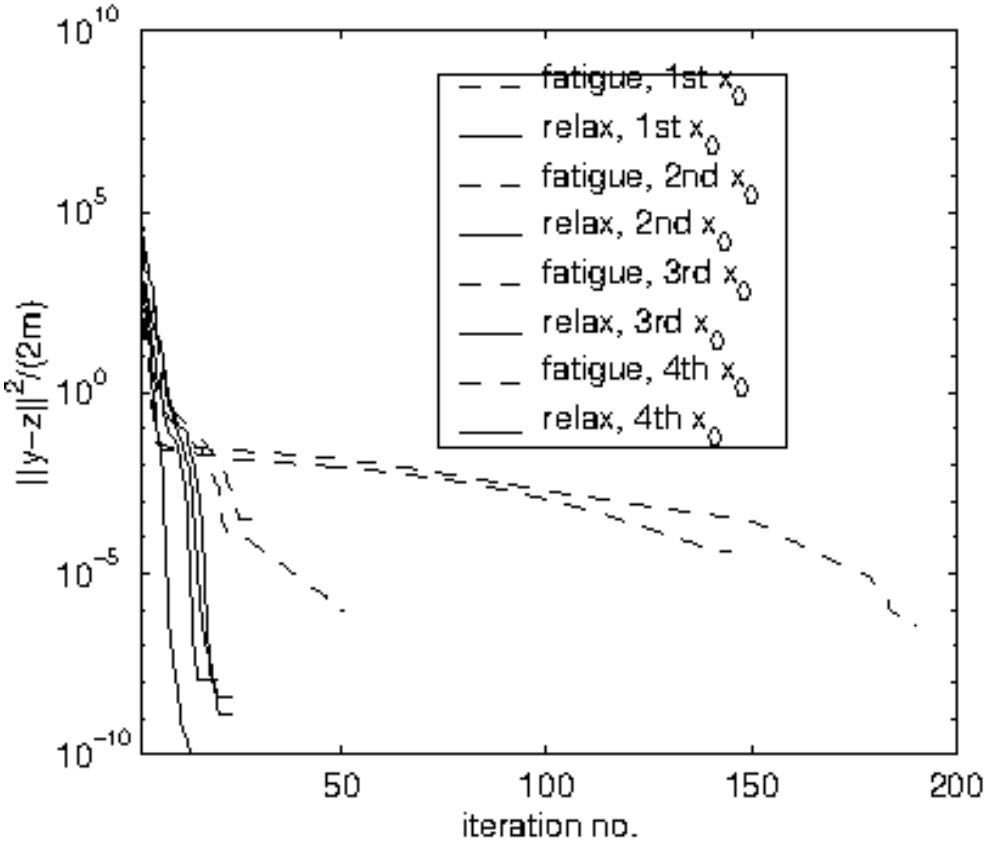
Levenberg-Marquardt algorithm
Expl. Identification of material model

It is easier to identify the model with the relaxation experiment.

This is seen through the condition number (~10^4 for the relaxation against ~10^8 for the fatigue)

The eigenvector associated to the smallest eigenvalue of the fatigue experiment, v4, corresponds to the asymptote

$$s(x^*) = \text{const}$$
$$v4^T . \nabla s(x^*) \approx 0$$



From Levenberg-Marquardt to evolution strategies

- s non-linear , $n > m$ $\Rightarrow$ local optima
- other, non smooth, distances , $f_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$

  for example f_1 for sparsity
- constraints handling through penalties

  $\Rightarrow$ a more general optimization algorithm discussed next

Talk outline

- Why numerical optimization may be useful to scientists making models ?
- Formulations and problems classification
- Optimization engines
- A few references
- Levenberg-Marquardt algorithm
- CMA-ES
- Concluding remarks
- (other algorithms)

Stochastic optimizer : CMA-ES

(N. Hansen et al., from 1996 on, then co-work with A. Auger, M. Schoenauer, M. Sebag, D. Brockoff, I. Loshchilov)

CMA-ES = *Covariance Matrix Adaptation Evolution Strategy* = optimization through multi-normal law sampling and updating.

Optimization engine = local sampling in probability.

A state of the art method for stochastic optimization.

Simplified flow chart of *ES-(1+1)*

Initializations : x, $f(x)$, m, C, t_{max}.

While $t < t_{max}$ do,

Sample $N(m,C)$ --> x'

Calculate $f(x')$, $t = t+1$

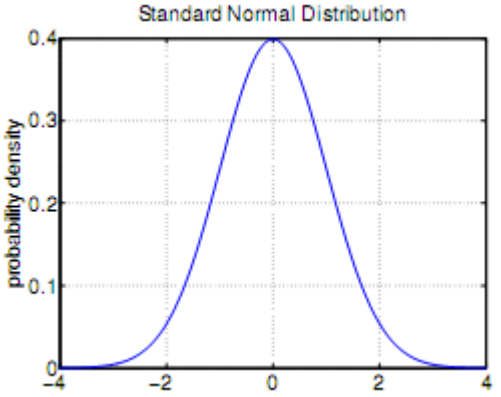
If $f(x')<f(x)$, $x = x'$, $f(x) = f(x')$ Endif

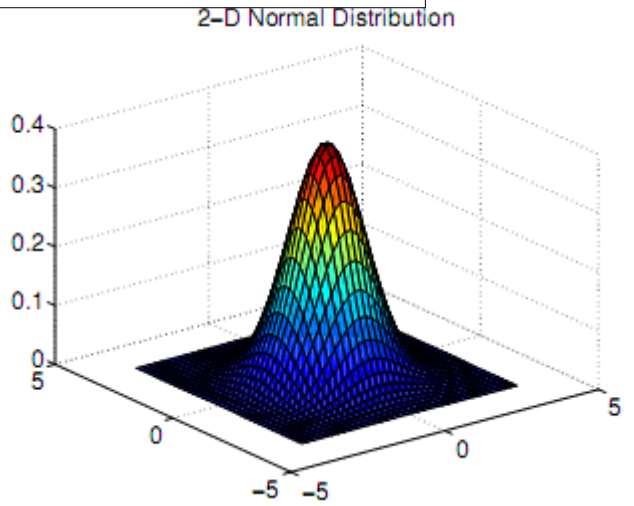
Update m *(e.g., $m=x$)* and C

End While

Normal law

$N(m,C)$





Numerical sampling of a multi-normal law

To sample $\boldsymbol{Y} \sim N(\boldsymbol{m}, \boldsymbol{C})$, perform the eigen analysis

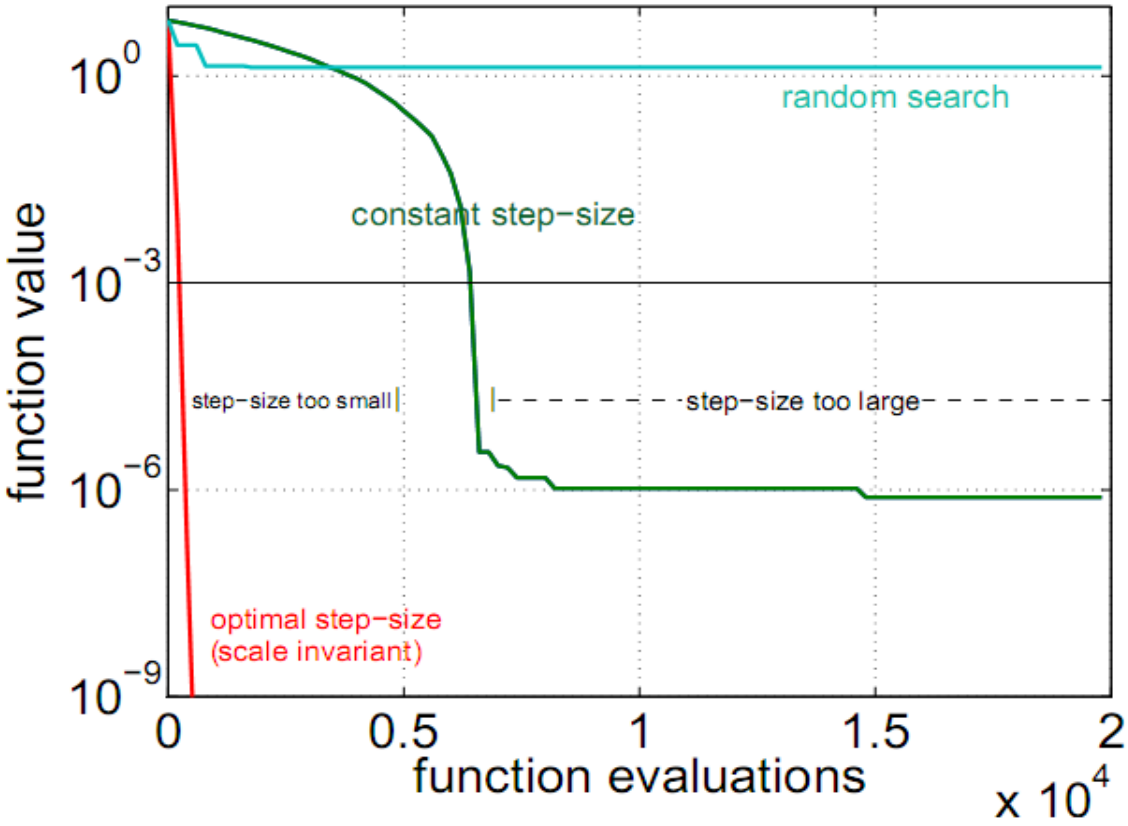
$\boldsymbol{C} = U D^2 U^T$ and then notice $\boldsymbol{Y} = \boldsymbol{m} + U D \mathrm{E}$, $\mathrm{E} \sim N(0, \boldsymbol{I})$

(prove that $E Y = \boldsymbol{\mu}$ and $\mathrm{Cov}(Y) = \boldsymbol{C}$,
+ a Gaussian vector is fully determined by its 2 first moments)

In R,

```
Ceig <- eigen(C)
y <- mu + Ceig$vectors %*% diag(sqrt(Ceig$values))
                        %*% matrix(rnorm(n))
```

Adapting the step size (C) is important



$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2$$

in $[-0.2, 0.8]^n$
for $n = 10$

(A. Auger et N. Hansen, 2008)

Above, ES(1+1) with isotropic step : $\boldsymbol{C} = \sigma^2 \mathrm{I}$, σ is the step size.
With an optimal step size ($\approx \|x\|/n$) for the sphere function, performance only degrades in $O(n)$!

Simplified CMA-ES flow chart

CMA-ES is an evolution strategy *ES-(μ,λ)* :

Initializations : m, C, t_{max}, μ , λ

While $t < t_{max}$ do,

Sample $N(m,C)$ --> $x^1,\ldots,x^\lambda$

Calculate $f(x^1),\ldots,f(x^\lambda)$, $t = t+\lambda$

Rank : $f(x^{1:\lambda}),\ldots,f(x^{\lambda:\lambda})$

Update m and C with the μ best, $x^{1:\lambda}$,...,$x^{\mu:\lambda}$

End While

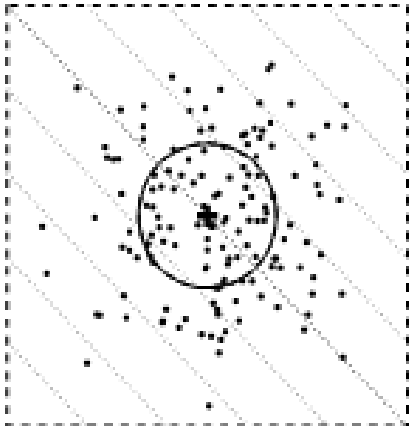
m and C are updated with

- the best **steps**,
- a **time cumulation** of these steps.

Simplified CMA-ES : C adaptation with the last good steps

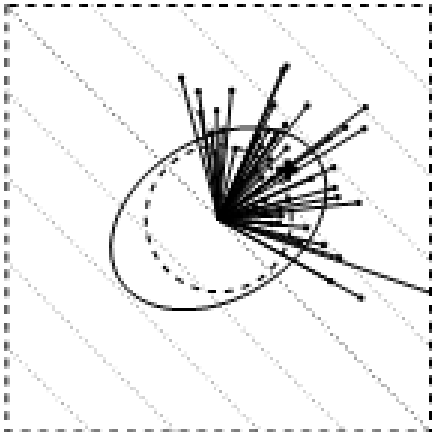
(A. Auger et N. Hansen, 2008)

Initialization : $m \in S$, $C = I$, $c_{cov} \approx 2/n^2$



sampling

$$x^i = m + y^i$$
$$y^i \propto N(0, C)$$
$$i = 1, \ldots, \lambda$$

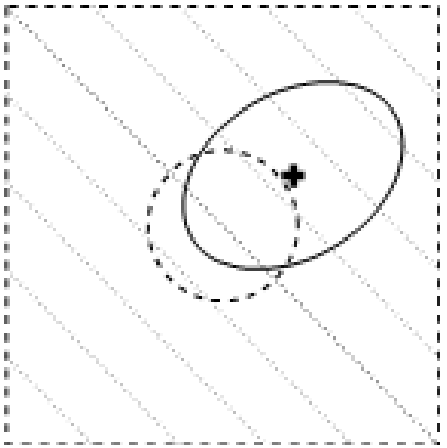


selection

$$y_w = \frac{1}{\mu} \sum_{i=1}^{\mu} y^{i:\lambda}$$

C rank 1 update

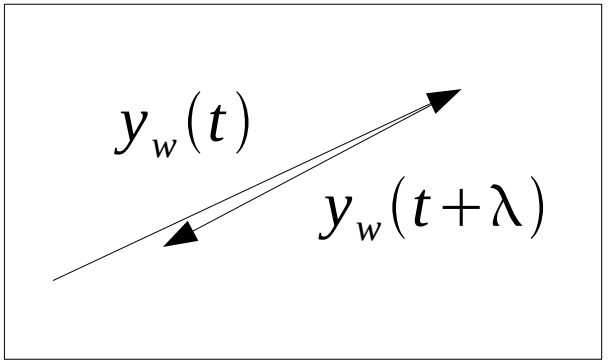
$$C \leftarrow (1 - c_{cov}) C + c_{cov}\, \mu\, y_w\, y_w^T$$



update m

$$m \leftarrow m + y_w$$

Simplified CMA-ES : time cumulation of good steps



When successive steps are anti-correlated, the step size should decrease, which is impossible so far since $y_w y_w^T = (-y_w)(-y_w)^T$.

Cumulation (time exponential damping) :

$$p_c \leftarrow \underbrace{(1-c_c)}_{\text{damping}} p_c + \underbrace{\sqrt{1-(1-c_c)^2}\sqrt{\mu}\, y_w}_{\text{normalization}}$$

$$C \leftarrow (1-c_{cov})C + c_{cov}\, p_c\, p_c^T \quad \leftarrow \text{rank 1}$$

$$c_c \approx 3/n \quad , \quad c_{cov} \approx 2/n^2$$

Further comments on CMA-ES

(check the BBOB web site for latest versions)

Invariance properties

• CMA-ES invariant w.r.t. monotonic transformation of f, e.g., f ← log(f), because comparison based.
• CMA-ES invariant w.r.t. change in coordinate system (rotation + translation) because of update formula.

Additional features of state of the art CMA-ES (/ previous slides) :

- steps are weighted, $y_w = \sum_{i=1}^{\mu} w_i \, y^{i:\lambda}$
- Rank 1 and μ updates of C carried out simultaneously,
- Global scaling of the covariance matrix, $C \rightarrow \sigma^2 C$.
- Restarts with a small and a large population size (BIPOP version).

Concluding remarks
Optimization added value beyond the algorithm run

- Optimization starts with a precise description of what to do with a simulator (mathematical formulation) in a decision making process.
- Problem solving will often show simulator / formulation flaws : repeat the optimization.
- Use the added value of the optimization effort : metamodels are constructed, (local or global) sensitivities become available.