# Optimisation de paramètres pour les modèles « incendies »

Benjamin BATIOT, Institut Pprime, Université de Poitiers

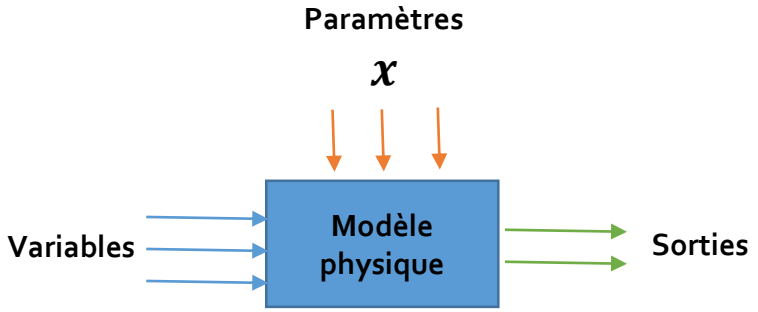
Anthony COLLIN, LEMTA, Université de Lorraine

# Introduction

- Vous faites une observation d'une réalité physique
- Le modèle que vous avez choisi pour représenter cette réalité physique se compose de paramètres inconnus propres à votre observation
- Les méthodes de résolution directe ne peuvent pas s'appliquer
- Comment trouver la valeur de ces paramètres ?
  ➔ Là commence l'optimisation !

# Introduction

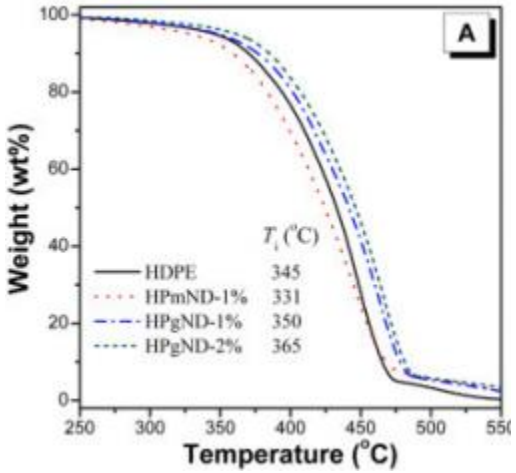
- Tout problème physique, lié à l'incendie ou non, peut se résumer de la sorte,




- Sorties : grandeurs observées par le modèle physique, comme le MLR, …
- Variables : grandeurs qui vont faire évoluer les sorties, le temps, …
- **Paramètres** : grandeurs qui caractérisent le modèle physique, des constantes physiques, les dimensions d'une pièce en feu, les propriétés physiques des matériaux, …

# Introduction

- Problème de l'optimisation : le modèle physique nécessite d'être « calibré » / ajusté à partir de données expérimentales.




Exemple : modèle de dégradation

*Quels paramètres de la loi d'Arrhénius $(A, E)$ permettent de représenter au mieux les données expérimentales ?*

- Objectif du module est de présenter les grands principes des méthodes d'optimisation et les critères qui sont à considérer avec attention.

# Introduction

## Problème d'optimisation

- Recherche d'un jeu de paramètres d'entrée $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ où chaque paramètre $x_i$ est défini par un domaine de définition

→ Espace de phase

Ces domaines de définition constituent les contraintes du problème d'optimisation.

- En pratique : intérêt certain de normaliser les paramètres par,

$$\tilde{\boldsymbol{x}} = \frac{\boldsymbol{x}-\boldsymbol{x}_{\mathbf{min}}}{\boldsymbol{x}_{\mathbf{max}}-\boldsymbol{x}_{\mathbf{min}}} \qquad \text{ou} \qquad \tilde{\boldsymbol{x}} = \mathbf{2}\frac{\boldsymbol{x}-\boldsymbol{x}_{\mathbf{min}}}{\boldsymbol{x}_{\mathbf{max}}-\boldsymbol{x}_{\mathbf{min}}} - \mathbf{1}$$

$$\tilde{\boldsymbol{x}} \in [0,1] \qquad\qquad \tilde{\boldsymbol{x}} \in [-1,1]$$

# Introduction

**Problème d'optimisation**

- A chaque jeu de paramètres d'entrée $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$, on associe un « score » qui rend compte de la qualité de la solution produite par le modèle physique, en procédant à une comparaison entre les résultats prédits et les données expérimentales.
- Construction d'une fonction « coût » ou d'une fonction objectif :

$$f = \left\| \boldsymbol{y}^{\text{prédit}}(\boldsymbol{x}) - \boldsymbol{y}^{\text{expé}} \right\|$$

Solution prédite

Données expérimentales

# Introduction

## Problème d'optimisation

- Construction d'une fonction « coût » ou d'une fonction objectif :

$$f = \left\| \boldsymbol{y}^{\text{prédit}}(\boldsymbol{x}) - \boldsymbol{y}^{\text{expé}} \right\| = \|\boldsymbol{z}\|$$

Quelle norme ?

- Norme 1,

$$f = \sum_{j=1}^{M} \left| y_j^{\text{prédit}}(\boldsymbol{x}) - y_j^{\text{expé}} \right|$$

- Norme 2, norme euclidienne,

$$f = \sqrt{\sum_{j=1}^{M} \left( y_j^{\text{prédit}}(\boldsymbol{x}) - y_j^{\text{expé}} \right)^2}$$

- Norme infinie,

$$f = \max\left( \left| \boldsymbol{y}^{\text{prédit}}(\boldsymbol{x}) - \boldsymbol{y}^{\text{expé}} \right| \right)$$

- $M$ correspond au nombre totale de données « observables », par exemple, les données expérimentales sont connus sur 200 pas de temps.

# Introduction

**Problème d'optimisation**

- Construction d'une fonction « coût » ou d'une fonction objectif :

$$f = \left\| \boldsymbol{y}^{\text{prédit}}(\boldsymbol{x}) - \boldsymbol{y}^{\text{expé}} \right\| = \|\boldsymbol{z}\|$$

- Inégalité de Minkowski (inégalité triangulaire pour les normes *p*)

$$\|\boldsymbol{z}\|_\infty \leq \|\boldsymbol{z}\|_p \leq n^{\frac{1}{p}} \|\boldsymbol{z}\|_\infty$$

Ainsi, toutes les normes sont équivalentes

# Introduction

**Problème d'optimisation**

- Objectif : rechercher le meilleur couple de paramètres d'entrée, $\boldsymbol{x}$, qui minimisera de manière globale la fonction coût,

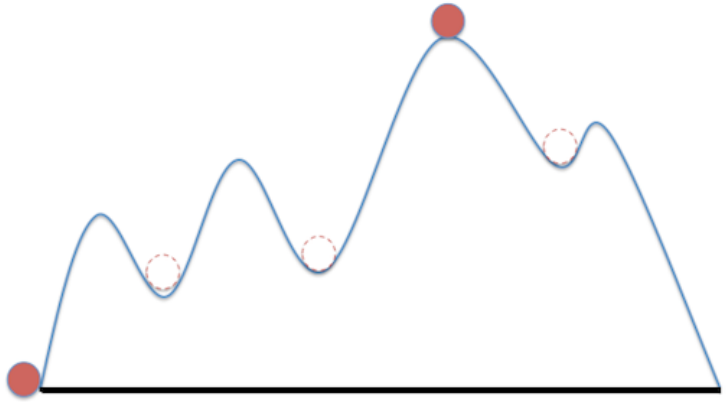
$$f = \left\| \boldsymbol{y}^{\text{prédit}}(\boldsymbol{x}) - \boldsymbol{y}^{\text{expé}} \right\|$$

Distinguer le minimum global et le minimum local

# Introduction

## Minimum global et le minimum local

- Considérons une balle posée au sommet d'une colline. Son équilibre est instable et au moindre coup de vent, la balle va glisser sur le long des flancs pour rejoindre la position d'énergie potentielle minimum, le pied de la colline, **le minimum global.**



- Notre colline présente quelques faux-plats ! La balle peut donc se trouver bloquée dans une cuvette et ne pas atteindre le bas de la colline. **C'est un minimum local.**

# Introduction

## Problème d'optimisation

- Objectif : rechercher le meilleur couple de paramètres d'entrée, $\boldsymbol{x}$, qui minimisera de manière globale la fonction coût,

$$f = \left\| \boldsymbol{y}^{\text{prédit}}(\boldsymbol{x}) - \boldsymbol{y}^{\text{expé}} \right\|$$

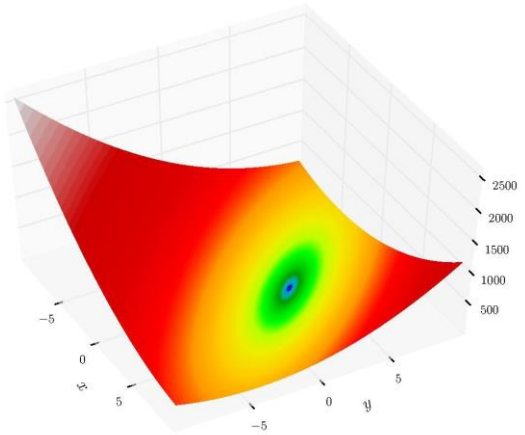
- Efficacité d'un algorithme d'optimisation réside dans sa capacité à s 'affranchir des différents minimums locaux.

## Caractéristiques de la fonction coût

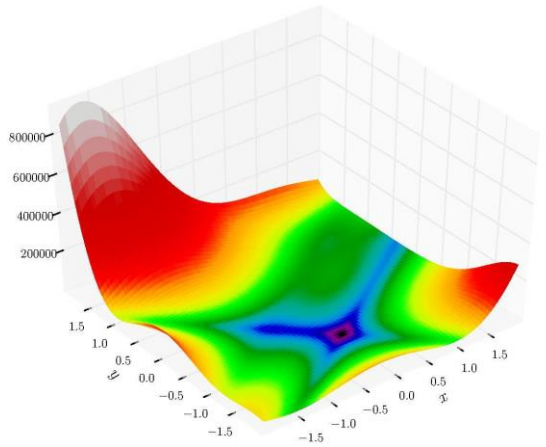
- Son expression peut être connue analytiquement, voire même dérivable ;
- Sa valeur ne peut être prédite que numériquement.

# Introduction

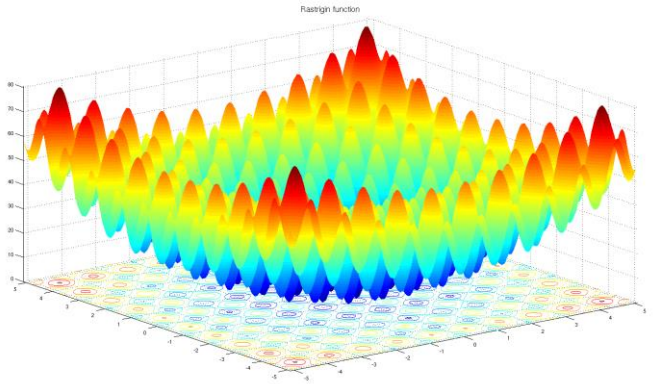
## Représentation de la fonction « coût »




« Booth function »




« Glodstein price function »




« Rastrigin function »

**Choix de la méthode**

# Introduction

**Analyse de sensibilité sur les paramètres à identifier**

*Thème d'un atelier ESIA 2015*

- Idée : est-ce que l'évolution d'un paramètre d'entrée sur sa gamme de recherche a une influence sur la fonction « coût »

*Est-ce que $f$ est sensible au paramètre d'entrée $x_i$ ?*

- L'application d'une analyse de sensibilité globale (type ANOVA) peut apporter un certain nombre d'éléments sur le comportement des paramètres d'entrée à identifier sur la fonction coût.

*Un paramètre non sensible = un paramètre non identifiable*

# Méthodes de recherche

**Méthodes de recherche**

- Optimisation numérique où les paramètres recherchés, $\boldsymbol{x}$, appartiennent à $\mathbb{R}^{\boldsymbol{n}}$, où $n$ est le nombre de paramètres recherchés ;
- Optimisation discrète (ou combinatoire) où $\boldsymbol{x}$ est fini ou dénombrable ;
- Commande optimale, où $\boldsymbol{x}$ est un ensemble de fonctions ;
- Optimisation stochastique, où $\boldsymbol{x}$ constituent des paramètres aléatoires ;
- Optimisation multicritère où $\boldsymbol{x}$ doit minimiser plusieurs fonctions « coûts » à la fois.

# Méthodes de recherche

**Algorithmes d'optimisation**

- Programmation linéaire : méthode du simplex, points intérieurs, …
- Optimisation non linéaire locale :
  - Avec **dérivées** et sans contraintes : méthodes de gradient, de Newton, de quasi-Newton, approche de Levenberg-Marquardt, …
  - Avec dérivées et avec contraintes : méthode SQP, Wilson, méthode de pénalisation, méthode lagrangienne, …
  - Sans dérivée : DFO, NEWUOA, MADS, …

# Méthodes de recherche

## Algorithmes d'optimisation

- Optimisation non linéaire globale :
  - Méthodes déterministes : simplexe non linéaire, réseaux de neurones, krigeage, …
  - Méthodes stochastiques : méthodes évolutionnaires, recuit-simulé, recherche de tabou.

# Critères d'arrêt

**Critères d'arrêt des algorithmes d'optimisation**

- Idéalement, la recherche d'un minimum est réussie quand,
$$\|\nabla f(\boldsymbol{x}_k)\| = 0$$
- En posant $\varepsilon > 0$, comme la précision demandée, le cas d'une optimisation différentiable sans contrainte devient,
$$\|\nabla f(\boldsymbol{x}_k)\| < \varepsilon$$

Mais bien souvent, $f$ ne possède pas d'expression analytique à dériver.

# Critères d'arrêt

## Critères d'arrêt des algorithmes d'optimisation

- Réalisation de tests d'optimalité,
  - Stagnation de la solution :

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| < \varepsilon(1 + \|\boldsymbol{x}_k\|)$$

  - Stagnation de la valeur de la fonction « coût » :

$$\|f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)\| < \varepsilon(1 + |f(\boldsymbol{x}_k)|)$$

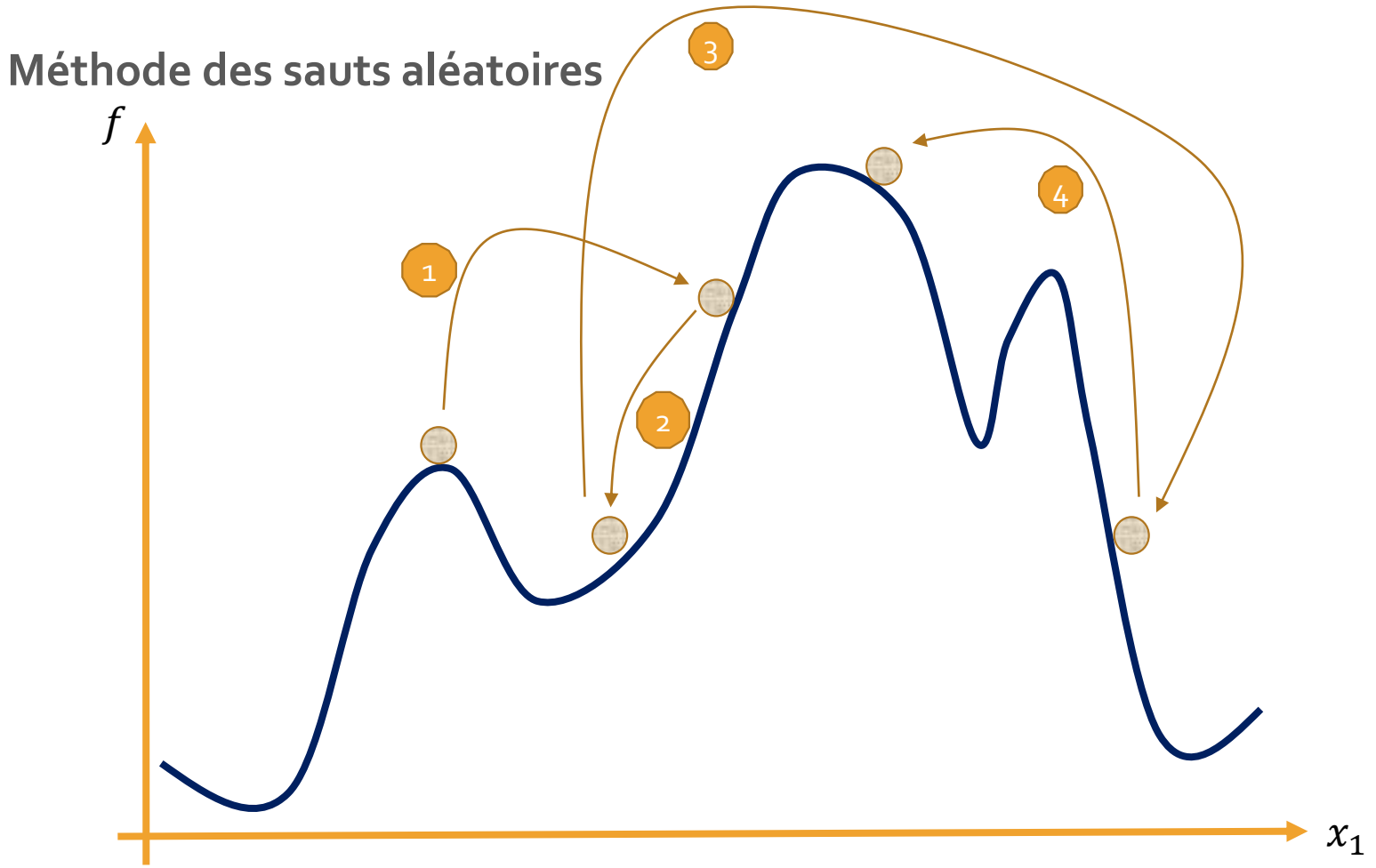
  - Nombre d'itérations dépassant un seuil fixé à l'avance :

$$k < IterMax$$

# Méthodes de recherche aléatoire

- Méthode des sauts aléatoires
- Méthode de la promenade aléatoire

# Méthode des sauts aléatoires



# Méthode des sauts aléatoires

## Méthode des sauts aléatoires

- Soit $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ le vecteur de paramètres recherchés dont les composantes sont bornées par $l_i \leq x_i \leq u_i, \forall x_i \in [\![1, n]\!]$ ;
- Soit $\boldsymbol{r}$ un vecteur de variables aléatoires, défini par $\boldsymbol{r} = \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}$ où $r_i$ est uniformément distribuée sur $[0,1]$ ;
- A chaque vecteur $\boldsymbol{r}$, on associe un vecteur $\boldsymbol{x}$ par,

$$\boldsymbol{x} = \begin{bmatrix} l_1 + r_1(u_1 - l_1) \\ \vdots \\ l_n + r_n(u_n - l_n) \end{bmatrix}$$

- La recherche du jeu de paramètres optimum $\boldsymbol{x}^*$ s'obtient par,

$$\boldsymbol{x}^* = \arg\left[\min\left(f(\boldsymbol{x})\right)\right]$$

# Méthode des sauts aléatoires

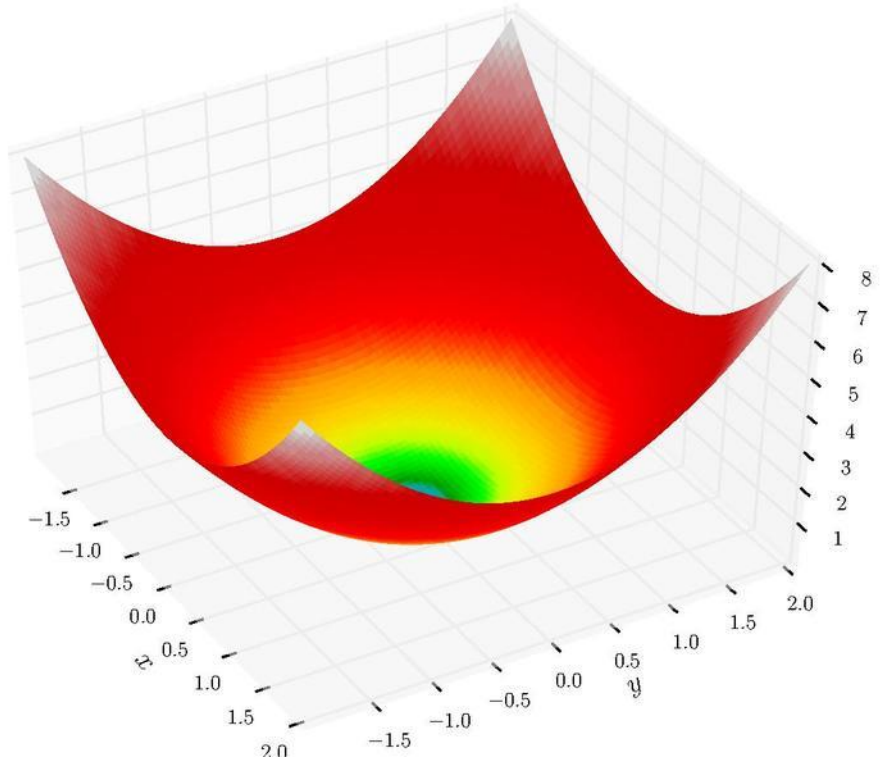
**Fonction Test pour l'optimisation :** « Sphere function »

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

Domaines de définition :

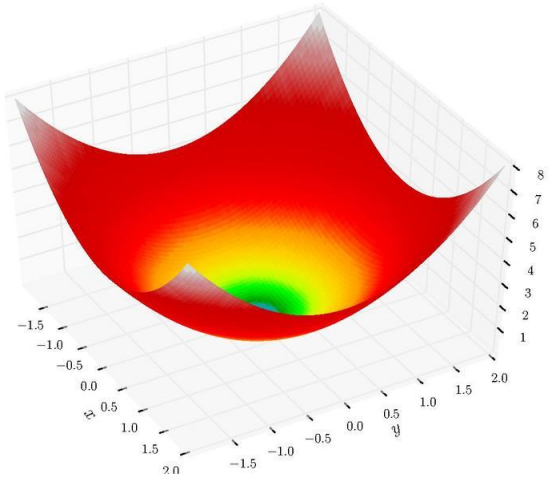
$-100 \leq x_i \leq 100$

Solution : $\boldsymbol{x}^* = (0, \cdots, 0)$



# Méthode des sauts aléatoires

**Fonction Test pour l'optimisation :** « Sphere function »



Nb de paramètres : 2

Nb d'itérations : 100 000

Couple optimum : $\boldsymbol{x}^* = (-0{,}233; -0{,}236)$

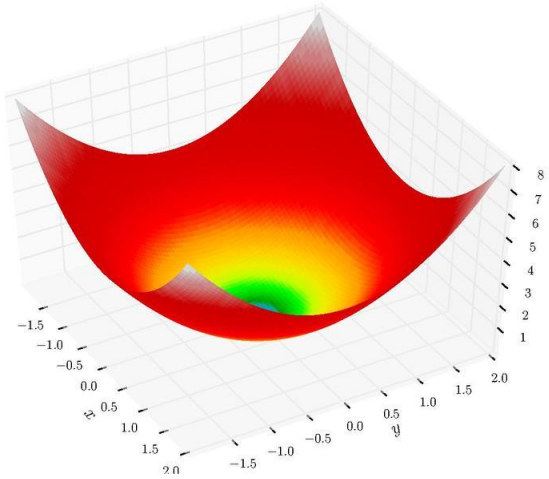
… identifié au bout de 2 580 itérations

Fonction coût : $f(\boldsymbol{x}^*) = 0{,}1103$

… vitesse de convergence lente, précision ?

# Méthode des sauts aléatoires

**Fonction Test pour l'optimisation :** « Sphere function »



Nb de paramètres : **4**

Nb d'itérations : 100 000

Couple optimum : $\boldsymbol{x}^* = (-1{,}09; -0{,}44; -5{,}87; -4{,}65)$

… identifié au bout de 97 095 itérations

Fonction coût : $f(\boldsymbol{x}^*) = 57{,}588$

… méthode inefficace pour de multi-paramètres

# Méthode des sauts aléatoires

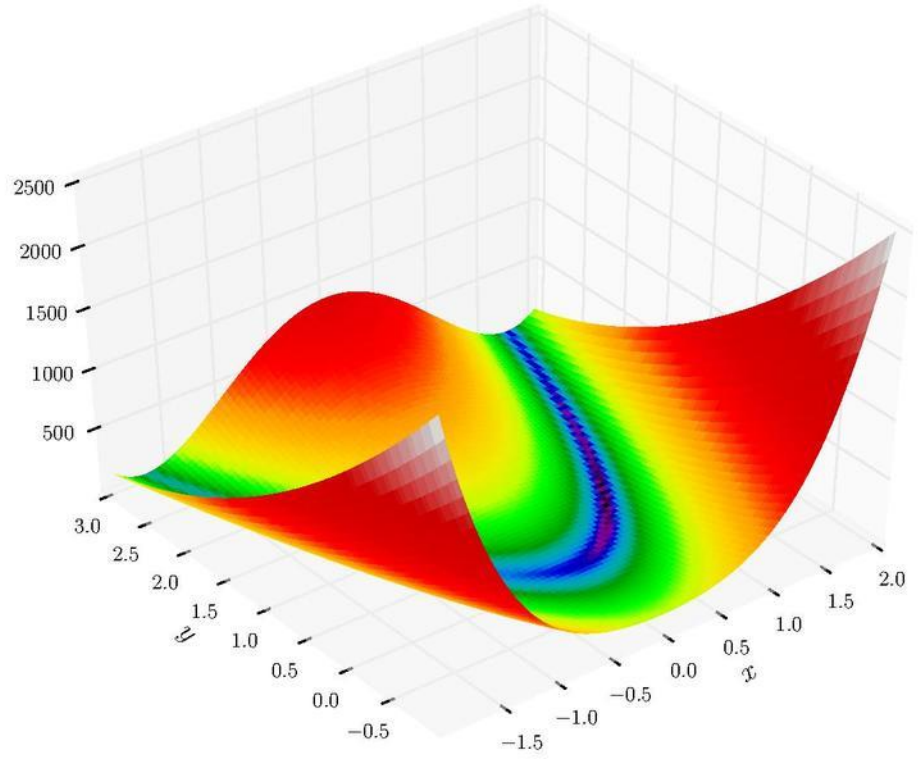
**Fonction Test pour l'optimisation :** « Rosenbrock function »

Définition : $f(\boldsymbol{x}) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right]$

Domaines de définition :

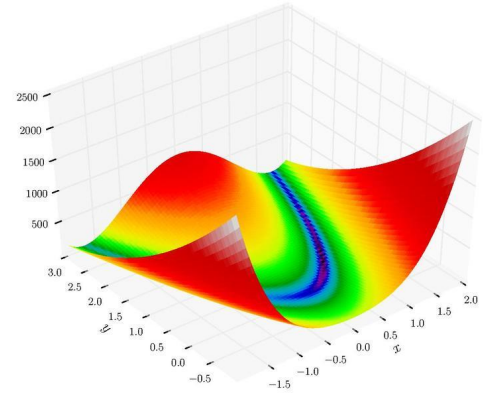
$-10 \leq x_i \leq 10$

Solution : $\boldsymbol{x}^* = (1, \ldots, 1)$



# Méthode des sauts aléatoires

**Fonction Test pour l'optimisation :** « Rosenbrock function »




Nb de paramètres : 2

Nb d'itérations : 100 000

Couple optimum : $\boldsymbol{x}^* = (1{,}057, 1{,}117)$

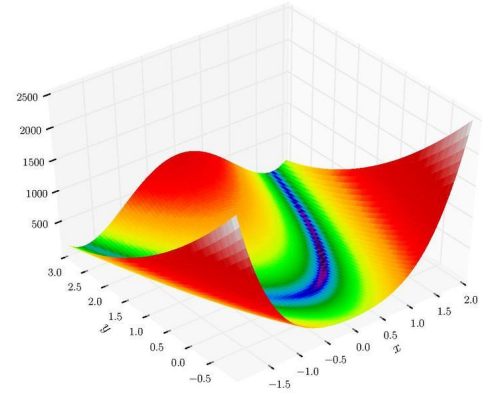
… identifié au bout de 11 605 itérations

Fonction coût : $f(\boldsymbol{x}^*) = 0{,}0032$

… vitesse de convergence lente, précision ?

# Méthode des sauts aléatoires

**Fonction Test pour l'optimisation :** « Rosenbrock function »




Nb de paramètres : **4**

Nb d'itérations : 100 000

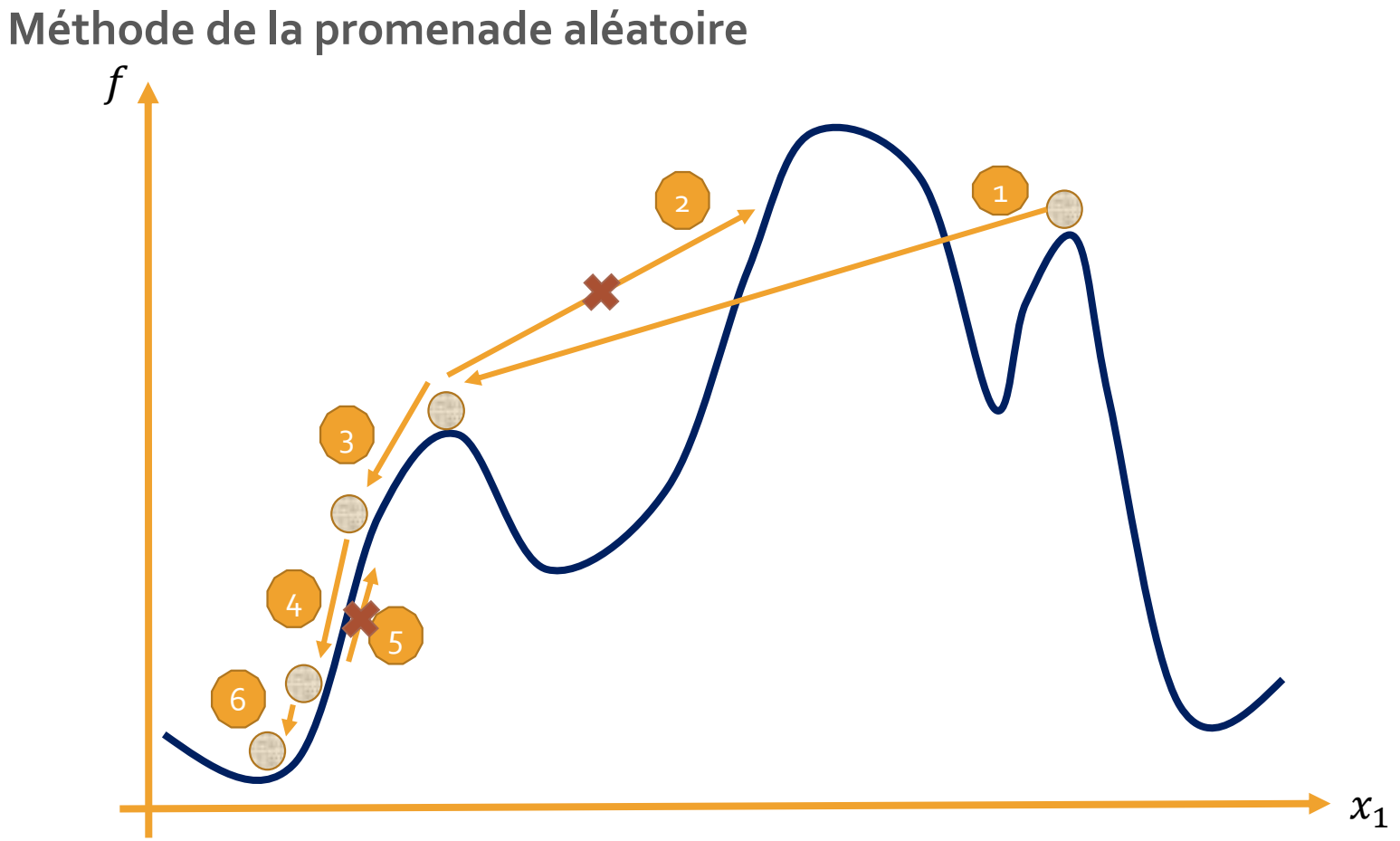
Couple optimum : $\boldsymbol{x}^* = (0{,}36; -0{,}09; 0{,}01; 0{,}14)$

… identifié au bout de 85 225 itérations

Fonction coût : $f(\boldsymbol{x}^*) = 9{,}73$

… vitesse de convergence lente, précision ?

# Méthode de la promenade aléatoire



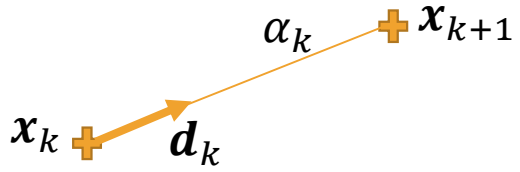
# Méthode de la promenade aléatoire

## Méthode de la promenade aléatoire

- Processus de marche aléatoire dans l'espace des phases :

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$$

$\boldsymbol{x}_k$ est la $k$ème position sur l'espace des phases, $\boldsymbol{d}_k$ est une direction normée de déplacement et $\alpha_k$ est la distance relative au déplacement



# Méthode de la promenade aléatoire

**Algorithme**

- Initialisation : $\boldsymbol{x}_0$, $\alpha_0$, $k = 0$ et $cc = 1$ ;

- Définition d'une direction aléatoire :

$\boldsymbol{r} = \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}$ où $r_i$ est une variable uniformément distribuée sur $[-1,1]$ ;

Si $R = \sqrt{r_1^2 + \cdots + r_n^2} > 1$,

alors génération d'un nouveau $\boldsymbol{r}$

sinon $\boldsymbol{d}_k = \frac{1}{R} \begin{bmatrix} r_1 \\ \vdots \\ r_n \end{bmatrix}$

# Méthode de la promenade aléatoire

- Détermination de la nouvelle position : $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$

- Test sur la nouvelle fonction « coût » associée à la nouvelle position :

Si $f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$,

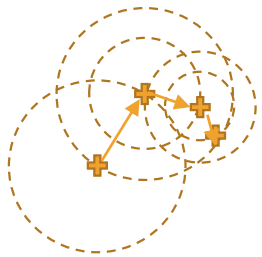
alors $k = k + 1$ et $cc = 1$

sinon Si $cc < N_{cc}$

alors $cc = cc + 1$

sinon $\alpha_k = \frac{\alpha_k}{2}$ et $cc = 1$

Si $\alpha_k < \varepsilon$ alors ARRET



# Méthode de recherche aléatoire

- Avantages des méthodes de recherche aléatoire :
  - La fonction « coût » peut être discontinue et non dérivable ;
  - L'algorithme recherche un minimum global ;
  - Ces approches sont à utiliser quand toutes les autres méthodes échouent (pratiquement aucun paramètre pour les méthodes) ;
  - L'utilisation de ces méthodes doit être une première étape avant de prospecter un algorithme plus performant.

# Méthode de recherche aléatoire

- Inconvénients des méthodes de recherche aléatoire :
  - Convergence très lente ;
  - Explosion combinatoire pour les problèmes de grandes dimensions.

**Première étape avant de prospecter un algorithme plus performant.**

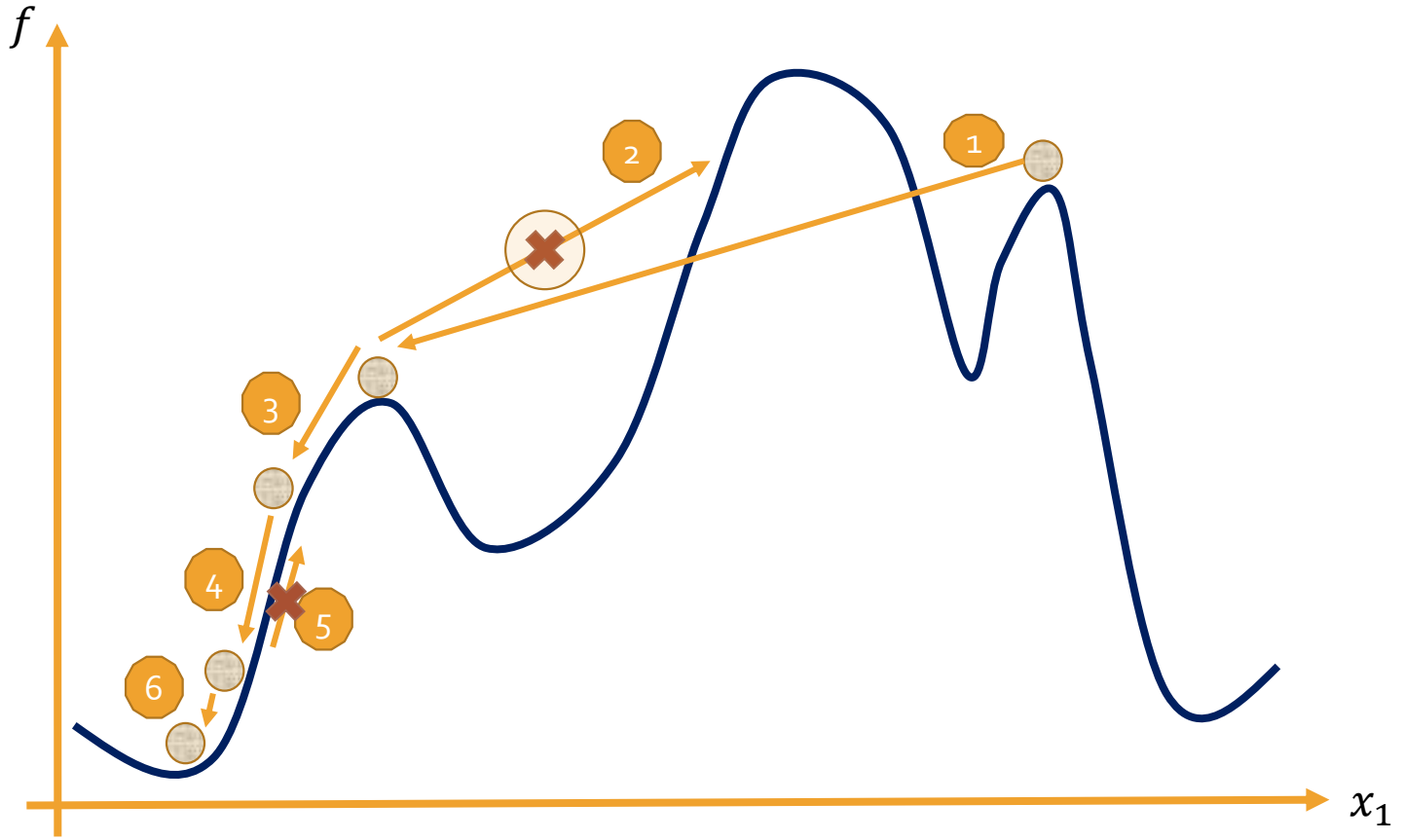
# Méthode du recuit simulé

## Méthode du recuit simulé

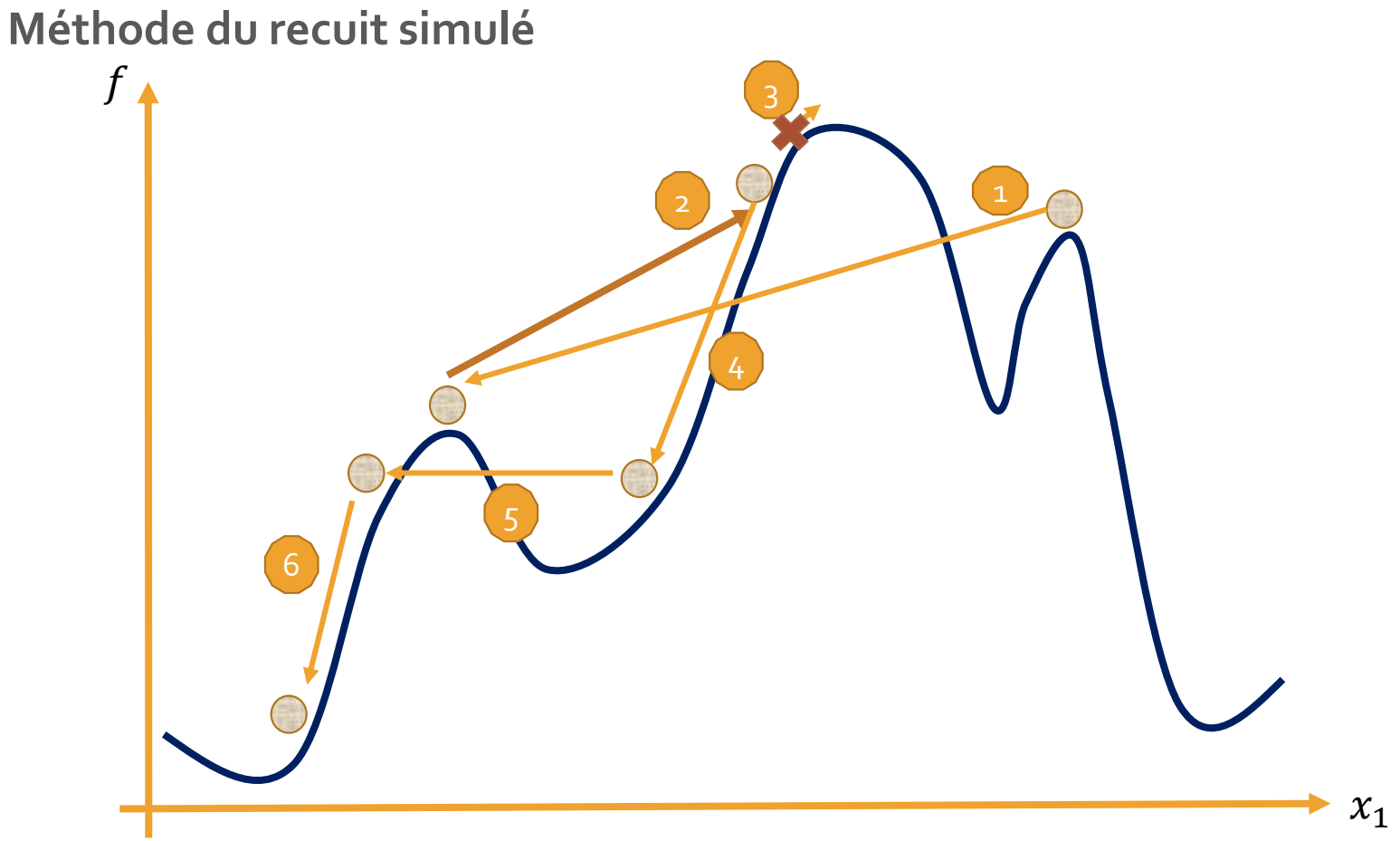
- Méthode d'optimisation présentant des analogies avec le processus thermique en physique de la matière : le recuit
  - Faire fondre un solide
  - Le refroidir lentement

# Méthode du recuit simulé

## Retour sur la méthode de la promenade aléatoire



# Méthode du recuit simulé



# Méthode du recuit simulé

**Algorithme de Métropolis** (simulation du recuit)

- Un état est défini par $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ le vecteur contenant les paramètres à estimer ;
- Un niveau d'énergie, noté $E_k$, est associé à l'état $\boldsymbol{x}_k$ par sa fonction coût,

$$E_k = f(\boldsymbol{x}_k)$$

- Pour une perturbation aléatoire de l'état, $\boldsymbol{x}_k \rightarrow \boldsymbol{x}_{k+1}$, avec un niveau $E_{k+1}$,
  - Si $E_{k+1} < E_k$, alors $\boldsymbol{x}_{k+1}$ est le nouvel état
  - Si $E_{k+1} > E_k$, alors $\boldsymbol{x}_{k+1}$ est le nouvel état avec une probabilité de $e^{-\frac{E_{k+1}-E_k}{k_B T}}$

  Où $k_B$ est la constante de Boltzmann et $T$ la température [K], et l'ensemble $k_B T$ décroit au cours des itérations.

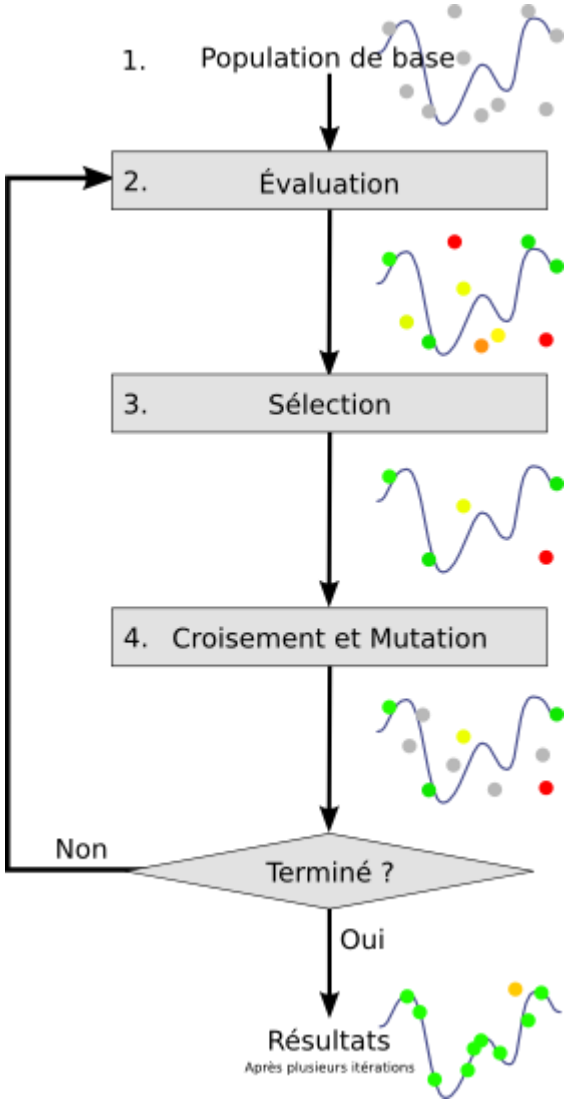
# Méthode du recuit simulé

**Avantages de la méthode du recuit simulé**

- Possibilité d'échapper d'un minimum local ;
- Agitation (taux d'acceptation) ajustable avec la température : utilisation de paliers de température introduite sous la forme d'une série, …

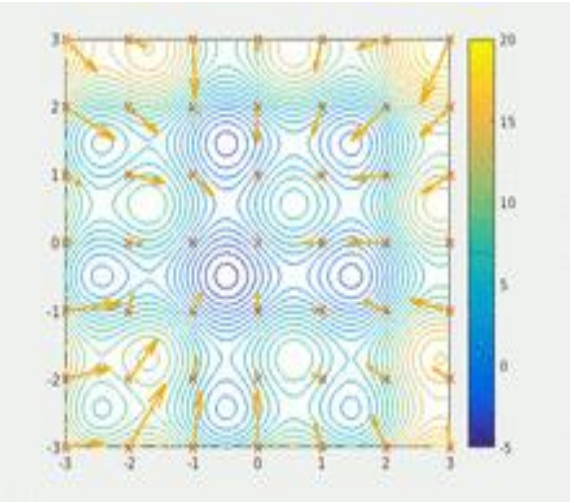
# Les algorithmes génétiques

- Algorithmes de type évolutionnaire
- Nombreuses variantes existent
- Basés sur la théorie de Darwin de l'évolution
- Ces algorithmes nécessitent beaucoup de paramètres à définir



# Les algorithmes par essaims de particules

- Algorithme de type comportemental
- Beaucoup de variantes existent
- Basé sur le comportement en groupe (abeilles)
- Nécessite peu de paramètres de réglage

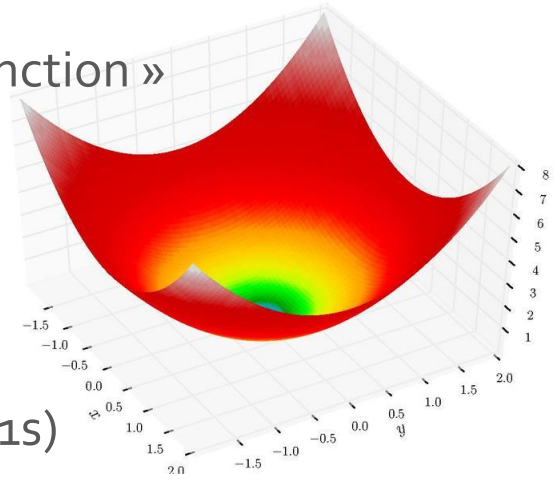


- Algorithme résout l'équation de la position et du mouvement

$$\begin{cases} \vec{v}_i(it) = w.\vec{v}_i(it-1) + \phi_1\left(\vec{p}_i - \vec{x}_i(it-1)\right) + \phi_2\left(\vec{p}_g - \vec{x}_i(it-1)\right) \\ \vec{x}_i(it) = \vec{x}_i(it-1) + \vec{v}_i(it-1) \end{cases}$$

# PSO

**Fonction Test pour l'optimisation :** « Sphere function »



Nb de paramètres : **4**

Nb d'itérations : 50 (3000 jeux testés en moins d'1s)

Couple optimum : $\boldsymbol{x}^* = (0{,}00; 0{,}00; 0{,}00; 0{,}00)$

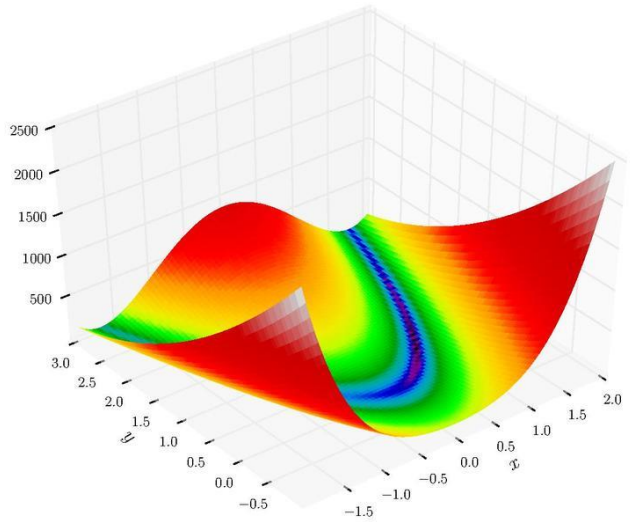
Fonction coût : $f(\boldsymbol{x}^*) = 3.10^{-6}$

# PSO

**Fonction Test pour l'optimisation :** « Rosenbrock function »

Nb de paramètres : 4

Nb d'itérations : 80 (4800 jeux testés en 1s)




Couple optimum : $\boldsymbol{x}^* = (1; 1; 1; 1)$

Fonction coût : $f(\boldsymbol{x}^*) = 0$

# Paysage des métaheuristiques

- Beaucoup d'algorithmes très différents
  - Optimisation par essaims de particules (PSO)
  - Algorithme génétique (AG)
  - shuffle complex evolution (SCE)
  - Ant colony (AC)
  - Algorithme par mémétique (MA)
  - Shuffle Frog Leaping (SFL)

- Etude de Elbeltagi et al* montre que globalement PSO fonctionne mieux que AG, MA, AC, SFL.
- Attention ces performances peuvent évoluer en fonction du problème mathématique à résoudre

**E. Elbeltagi, T. Hegazy, D. Grierson. Comparison among five evolutionary-based optimization algorithms. Adavcned engineering informatics, Elsevier, 19, 43-53, 2005.*