École chercheurs MEXICO, La Rochelle, Mars 2018

Model Based Optimization

Victor Picheny, victor.picheny@inra.fr
Nicolas Durrande, nicolas@prowler.io

Introduction : global optimization

Optimisation pour l'aide à la conception / décision

- Réponse du modèle = performance ou coût
- Recherche des paramètres optimaux :

$$x^* = \arg\min cout(x) \text{ ou } \arg\max perf(x)$$

Optimisation pour la calibration

- Sorties du modèle $\Rightarrow$ comparaison à des observations
- On veut minimiser un écart quadratique (ou autre métrique)

$$x^* = \arg\min WLS(x)$$

Dans les 2 cas

- L'optimisation nécessite beaucoup d'appels au code
- Métamodèle : solution naturelle

Le compromis **exploration / intensification**

Est-ce qu'on souhaite

- améliorer une solution existante ?
- essayer "toutes" les valeurs possibles de paramètres pour trouver la meilleure

Dans la vraie vie : tout essayer n'est pas possible

On va chercher un compromis entre

- améliorer l'existant par de petites modifications
- chercher une meilleure solution radicalement différente

Le compromis **exploration / intensification**

Optimisation locale

Amélioration depuis un point initial

Optimisation globale : on cherche un compromis entre

- Exploration : recherche partout dans l'espace pour ne pas rater la zone optimale
- Intensification : une fois une zone identifiée : on recherche le minimum local

Dans un contexte de planification d'expériences

- Exploration : remplissage d'espace
- Intensification : "ciblage"

Introduction à l'optimisation globale : l'algorithme DIRECT

Garanti sans métamodèle !

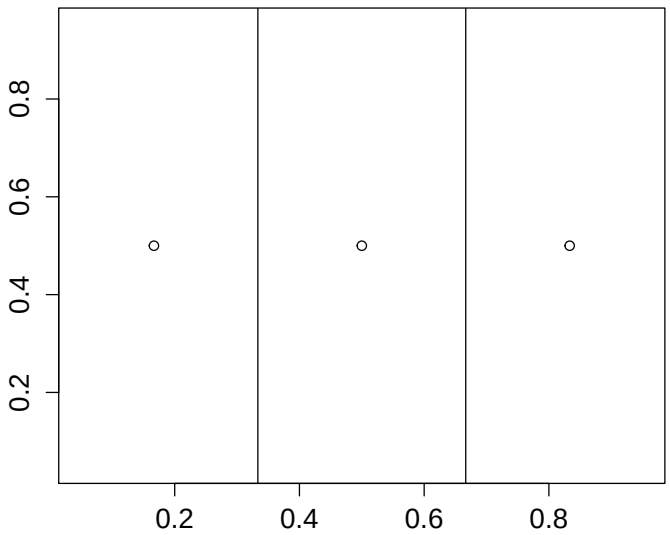
DIRECT : DIviding RECTangles

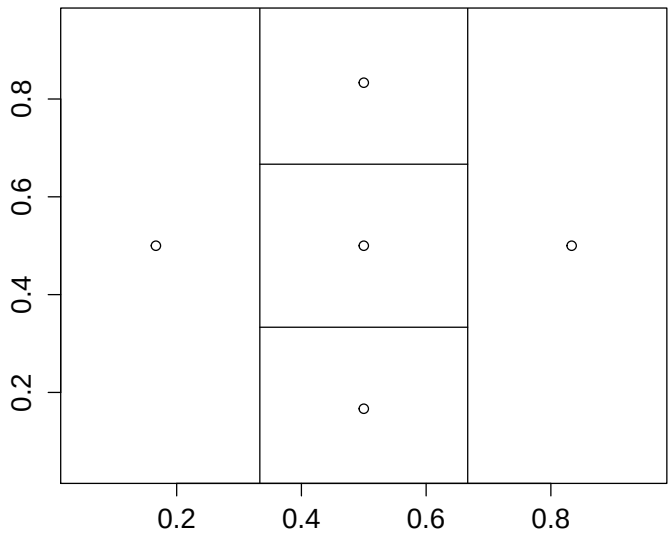
- Découpage de l'espace en (hyper)rectangles
- Un échantillon au centre de chaque rectangle
- On divise les rectangles les plus "intéressants" :
  - soit les plus grands (exploration)
  - soit ceux qui ont une valeur au centre basse (intensification)
- Pour diviser : ajout de 2 points, division en 3

D. Jones, C. Perttunen, B. Stuckman (1993)
Lipschitzian optimization without the Lipschitz constant
Journal of Optimization Theory and Applications 79(1), 157-181

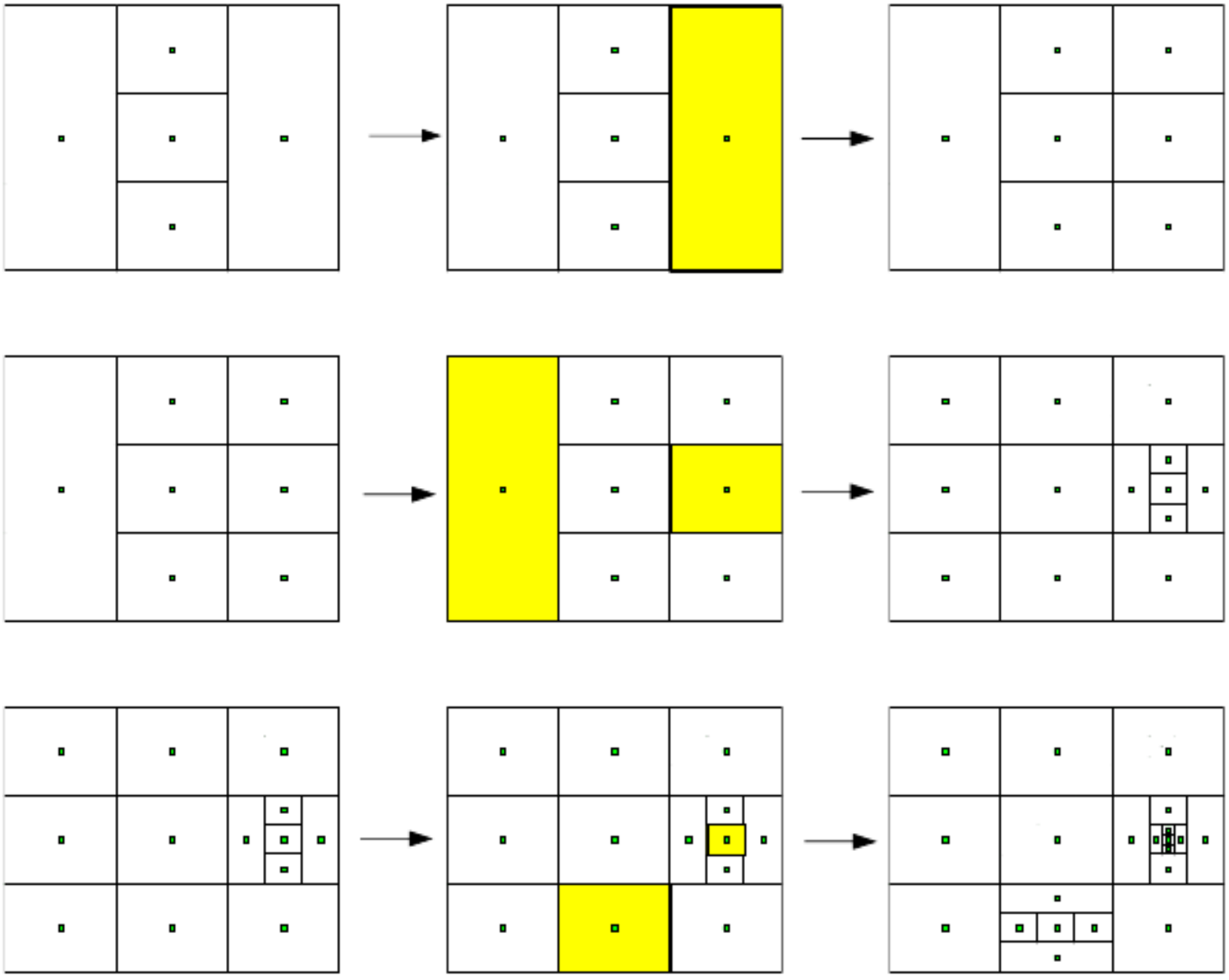
Exemple en dimension 2

- Départ : 3 points équirépartis dans une direction aléatoire
- On divise le rectangle ayant la meilleure observations



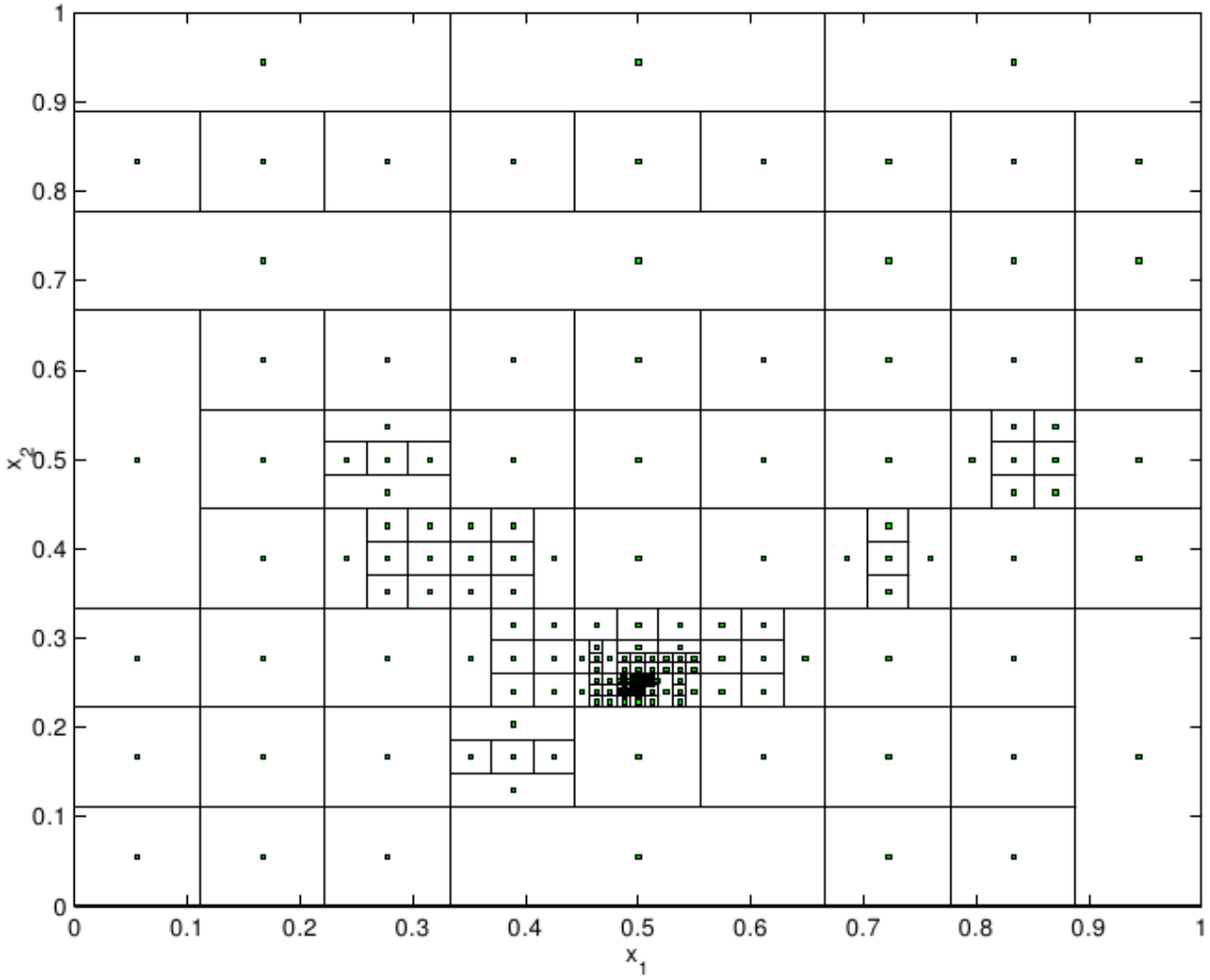


Exemple en dimension 2



Après 191 évaluations

- Echantillonnage intense dans la zone de l'optimum
- Bonne exploration



Source figures :

D. E. Finkel
DIRECT Optimization Algorithm User Guide (2003)

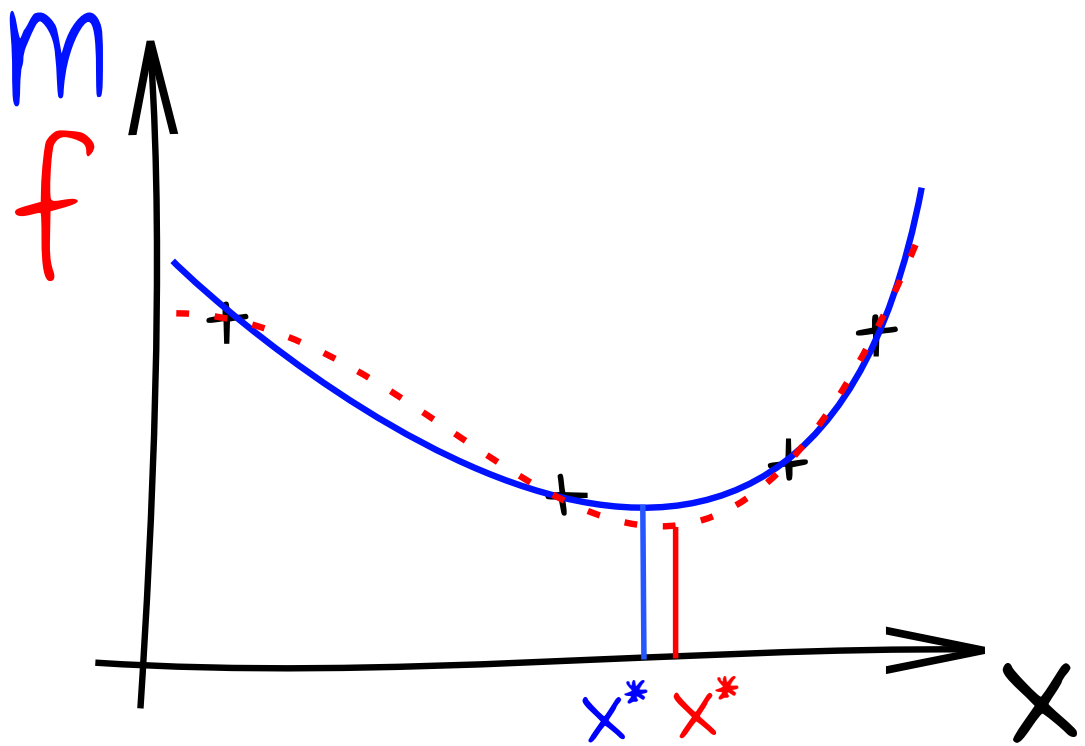
Intêret et limites

+ Exploration de tout l'espace de recherche
+ Stratégie robuste
– Limité aux petites dimensions
– Exploitation limitée de l'information

$\Rightarrow$ même principe général, avec un métamodèle ?

Trust regions (quadratic metamodels)

Si le nombre d'appels au modèle est trop limité, on peut optimiser le métamodèle à la place :

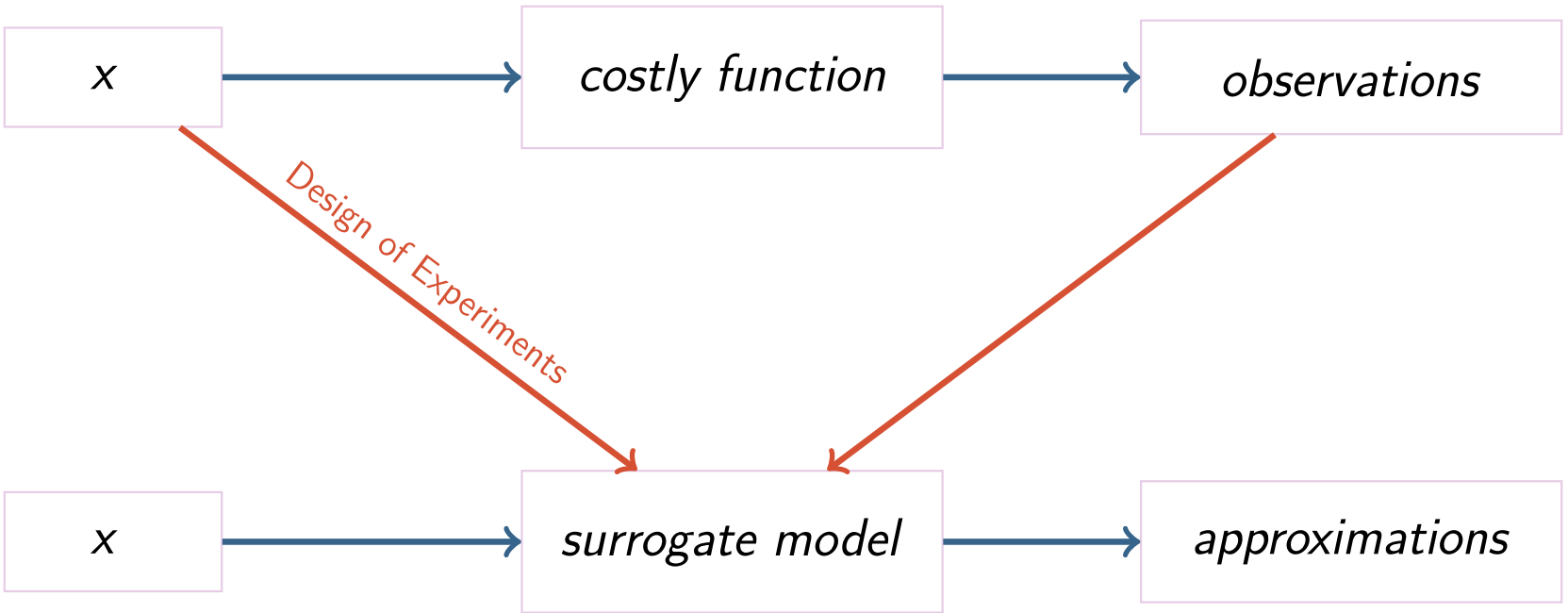


Et on espère qu'à la fin on ait :

$$\text{argmin}(m) \approx \text{argmin}(f)$$

$$\min(m) \approx \min(f)$$

Schéma global



Optimisation et métamodèle : ce qu'on est tenté de faire...

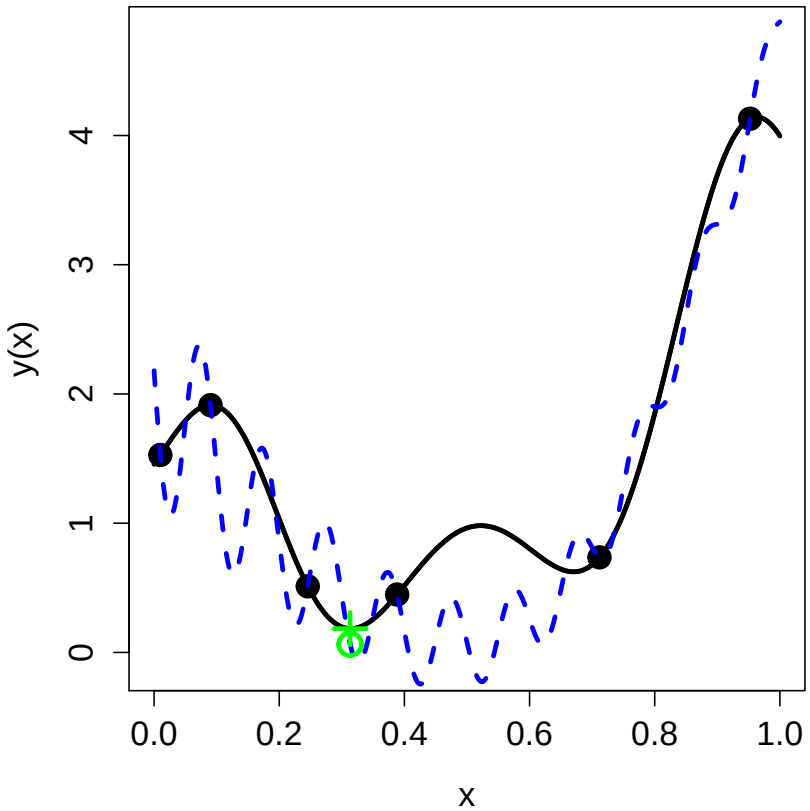
"Le métamodèle donne l'optimum"

- On cherche le minimum x^* sur le métamodèle
- On évalue le vrai $y(x^*)$ sur le simulateur

$\Rightarrow$ C'est fini !

Répartition de l'effort

- Plan initial : 49 expériences
- 98% exploration, 2% exploitation



Que faire si x^* n'est pas bon ?

Optimisation et métamodèle : ce qu'il faut faire

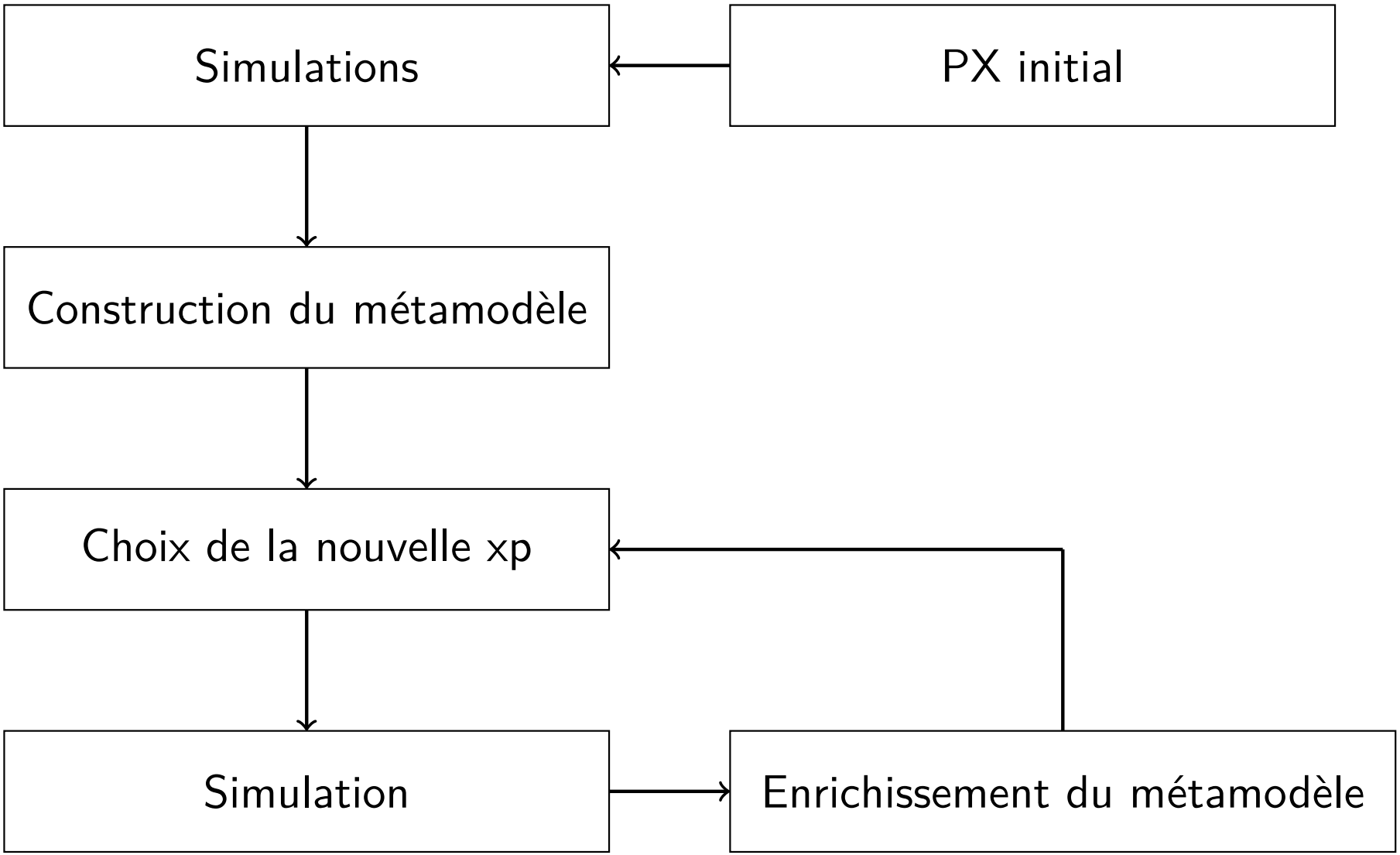
Si le budget est fixe

- On divise le budget en 2
- Budget 1 : plan initial (LHS)
- Budget 2 : optimisation

Utilisation **séquentielle** du métamodèle

- Métamodèle initial : *a priori* peu précis
- Le métamodèle sert à **choisir** pour les nouvelles observations
- A chaque nouvelle observation : amélioration du métamodèle

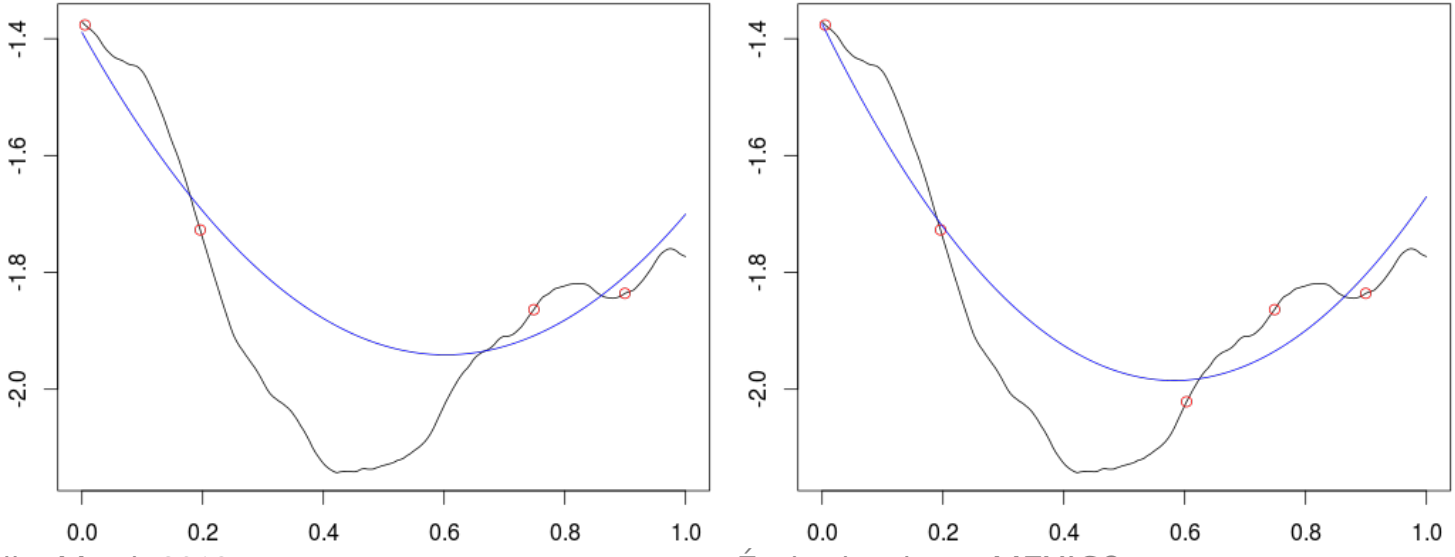
Schéma général : métamodèle = guide



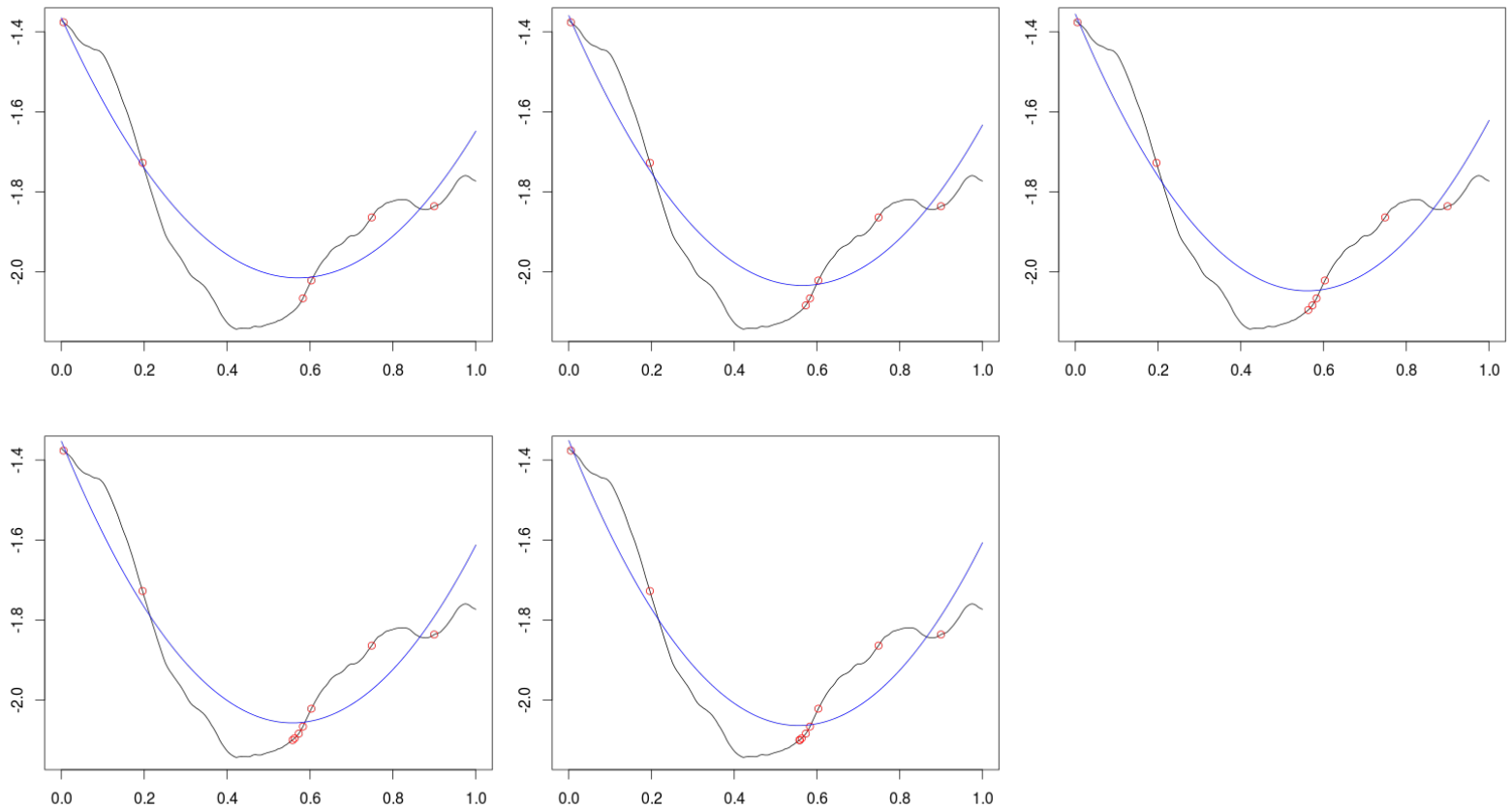
Optimisation basée sur les modèles polynomiaux

Principe

- On construit une surface de réponse $y = \beta_0 + \beta_1 x + \beta_2 x^2$
- On cherche le point qui minimise la surface de réponse
- On ajoute ce point
- On met à jour la surface de réponse
- On recommence...



Itérations 3 à 7



Optimisation basée sur les modèles polynomiaux

Problème : modèle “rigide”

Le modèle ne s'ajuste pas aux données : $Y = \mathbf{X}\beta + \epsilon$

Pas de convergence vers un modèle précis, même localement

Solutions

1. Augmenter le dégré du polynôme
   $\Rightarrow$ risque de surapprentissage & d'instabilité !
2. Supprimer des points
   $\Rightarrow$ méthode **de région de confiance**

Régions de confiance : principe

Modèle quadratique "creux"

- Valide à l'intérieur d'une région de confiance (petite)
- Construit uniquement avec les points à l'intérieur de la région
- Selon les valeurs des simulations, on modifie la taille de la région

Gestion de la région de confiance

A chaque itération :

- $\hat{y}(x^*)$ bon $\Rightarrow$ confiance dans le modèle : on augmente la taille
- $\hat{y}(x^*)$ mauvais $\Rightarrow$ modèle peu fiable : on diminue la taille

+ beaucoup de règles pour sélectionner les points et enrichir le plan d'expériences

Illustration (source : F. Vanden Berghen)

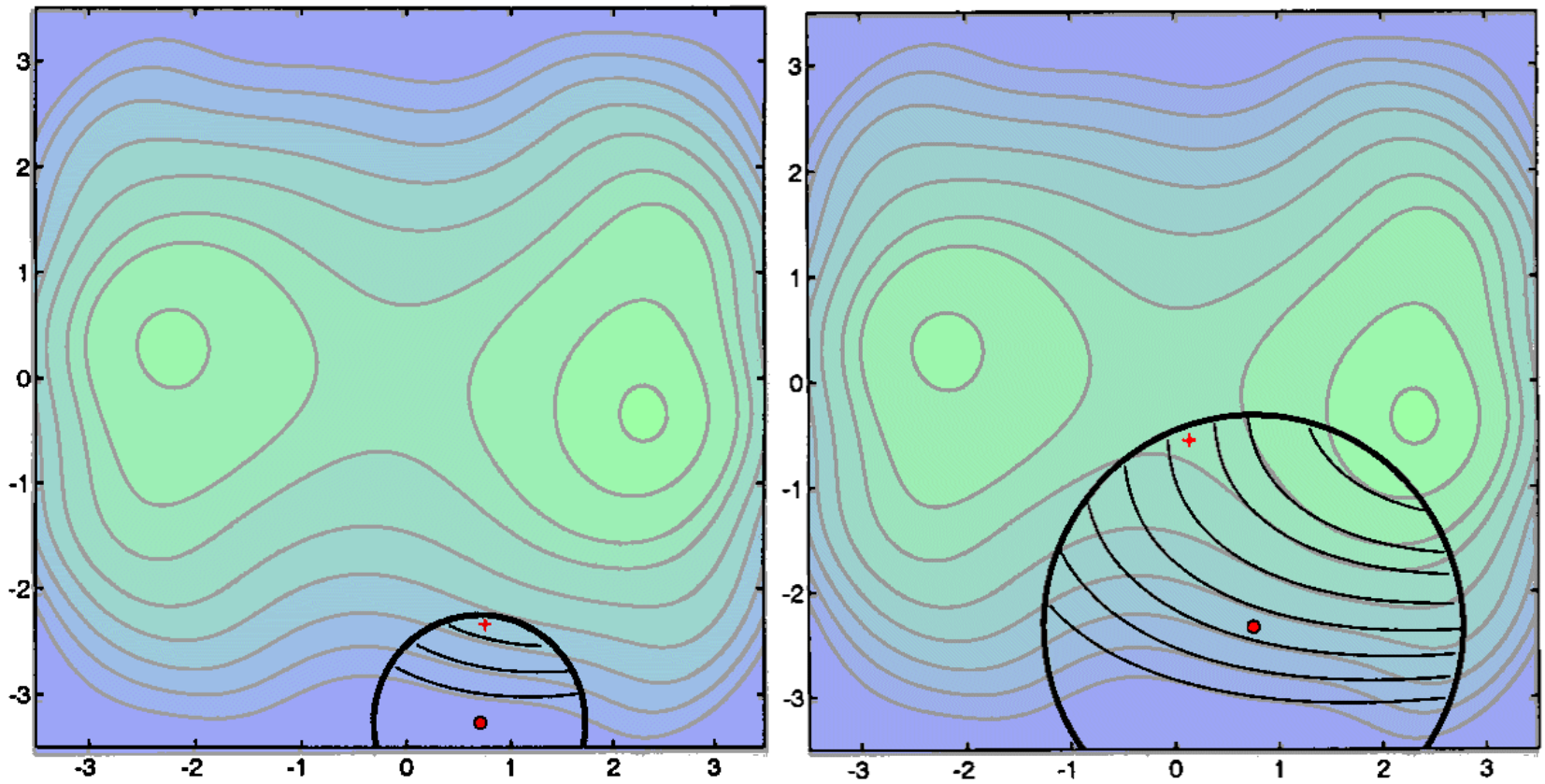


Illustration (source : F. Vanden Berghen)

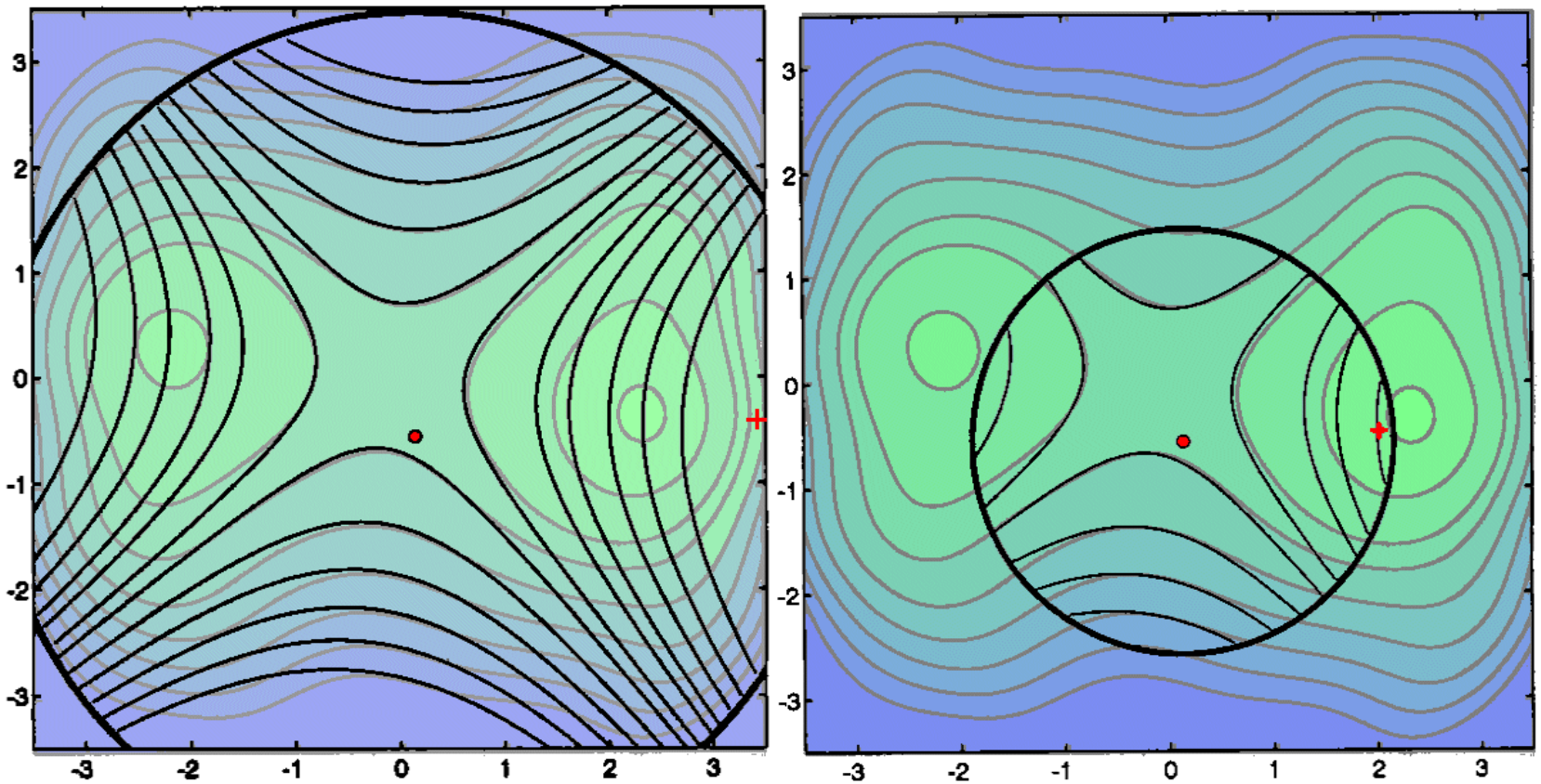
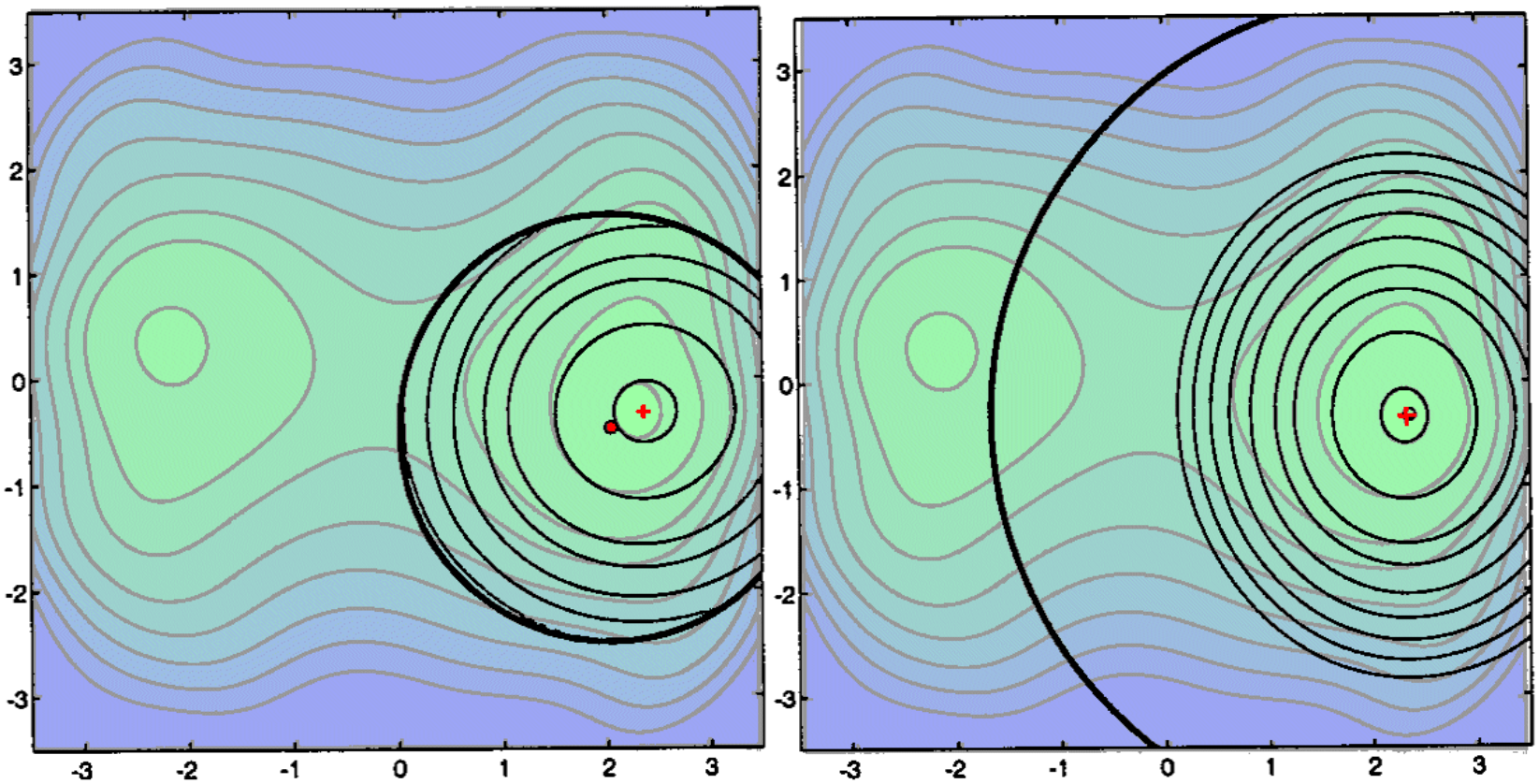


Illustration (source : F. Vanden Berghen)



Avantages

- Garantie de convergence
- Méthodes assez parcimonieuses
- Robuste
- Accepte un très grand nombre de variables

Conn, Scheinberg, and Vicente
Introduction to derivative-free optimization
MPS-SIAM Series on Optimization (2009)

Powell
The NEWUOA software for unconstrained optimization without derivatives
Large-scale nonlinear optimization (2006)

Méthode locale (proche méthode de gradient)

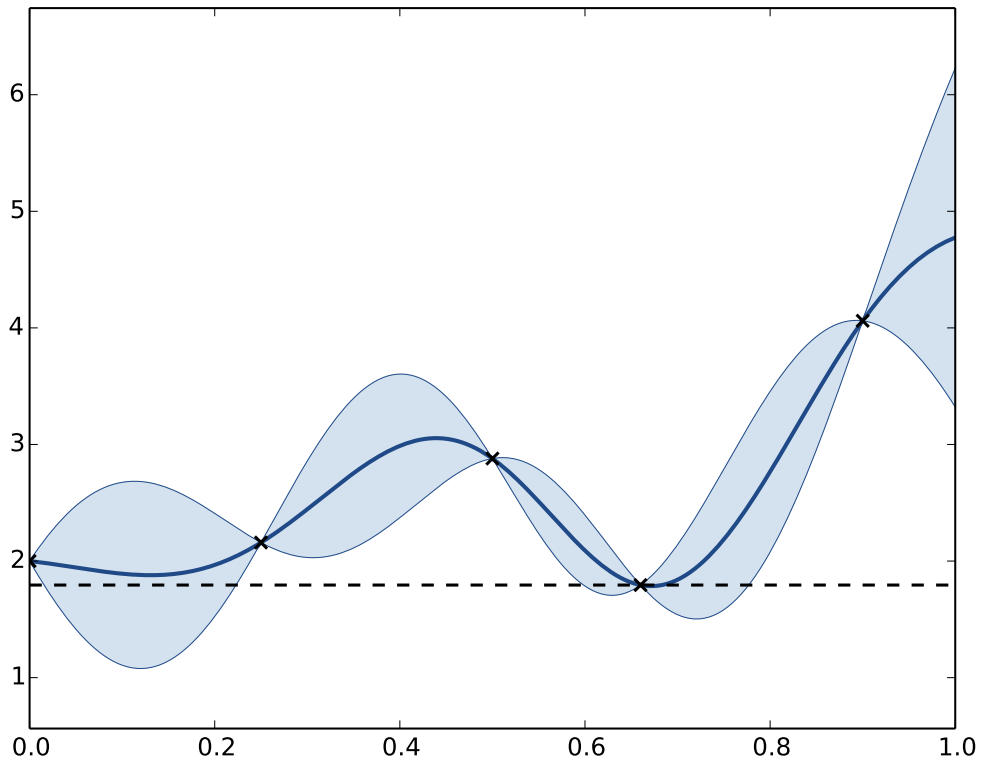
Pas de métamodèle final utilisable globalement - mais gradient + hessien !

Kriging-based optimization (EGO)

Global optimization methods are a trade-off between

- Exploitation of past good results
- Exploration of the space

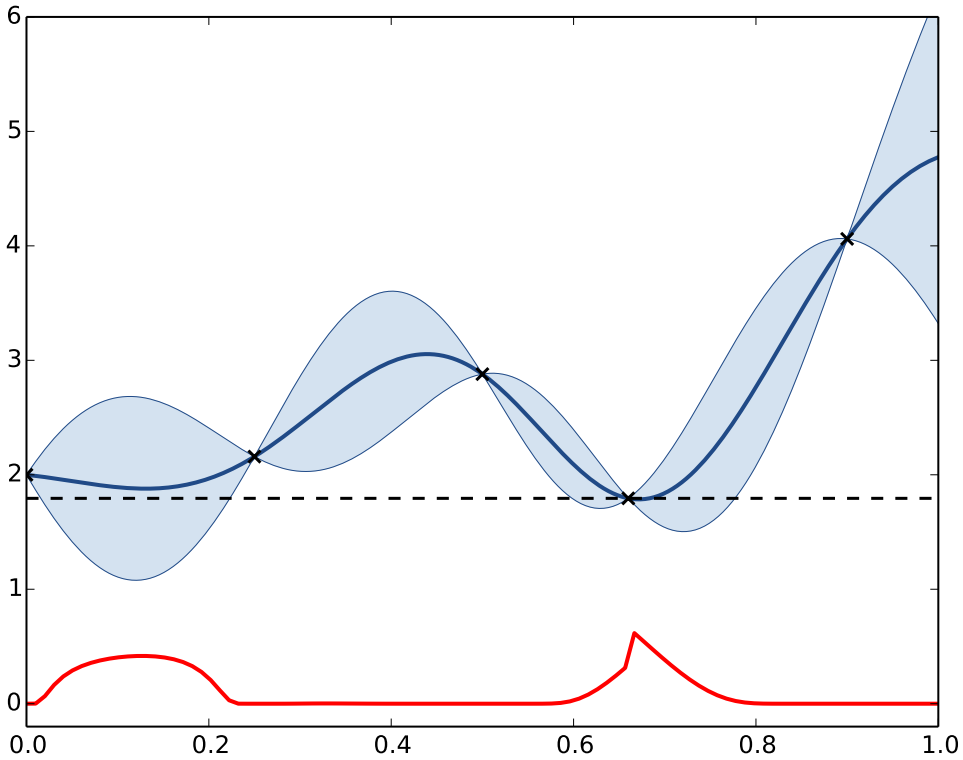
How can GPR models be helpful?



For now the best observed value is 1.79.

Probability of Improvement:

$$PI(x) = cdf\left(\frac{\min(F) - m(x)}{\sqrt{(c(x,x))}}\right)$$



The point with the highest PI is often very close to the best observed value. We can show that there is a x in the neighbourhood of x^* such that $PI(x) \geq 0.5$.

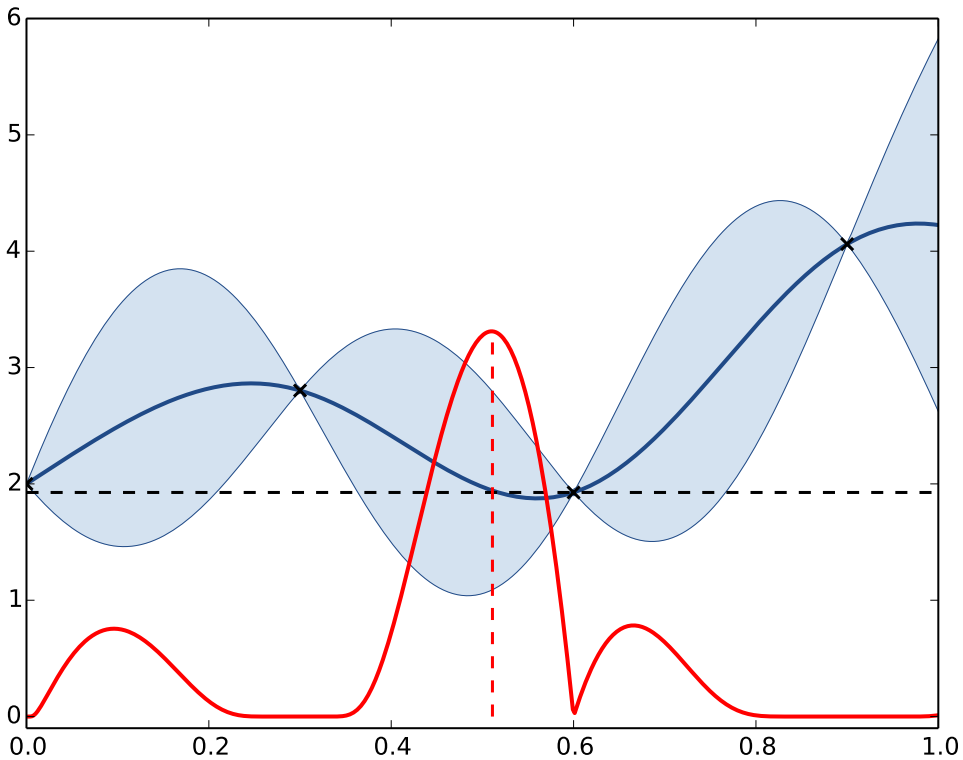
For such points, the improvement cannot be large...

Can we find another criterion?

Expected Improvement:

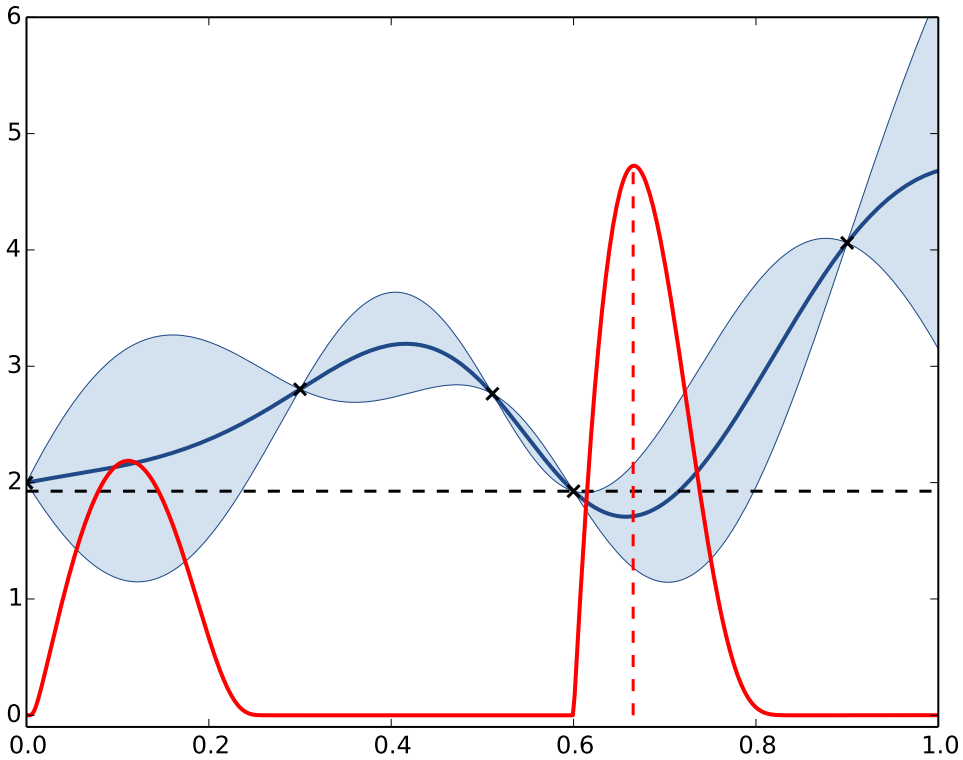
$$EI(x) = \int_{-\infty}^{\min(F)} \max(0, Y(x)) \; dy(x) = \cdots =$$

$$\sqrt{c(x,x)}(u(x)cdf(u(x)) + pdf(u(x))) \quad \text{with } u(x) = \frac{\min(F) - m(x)}{\sqrt{(c(x,x))}}$$



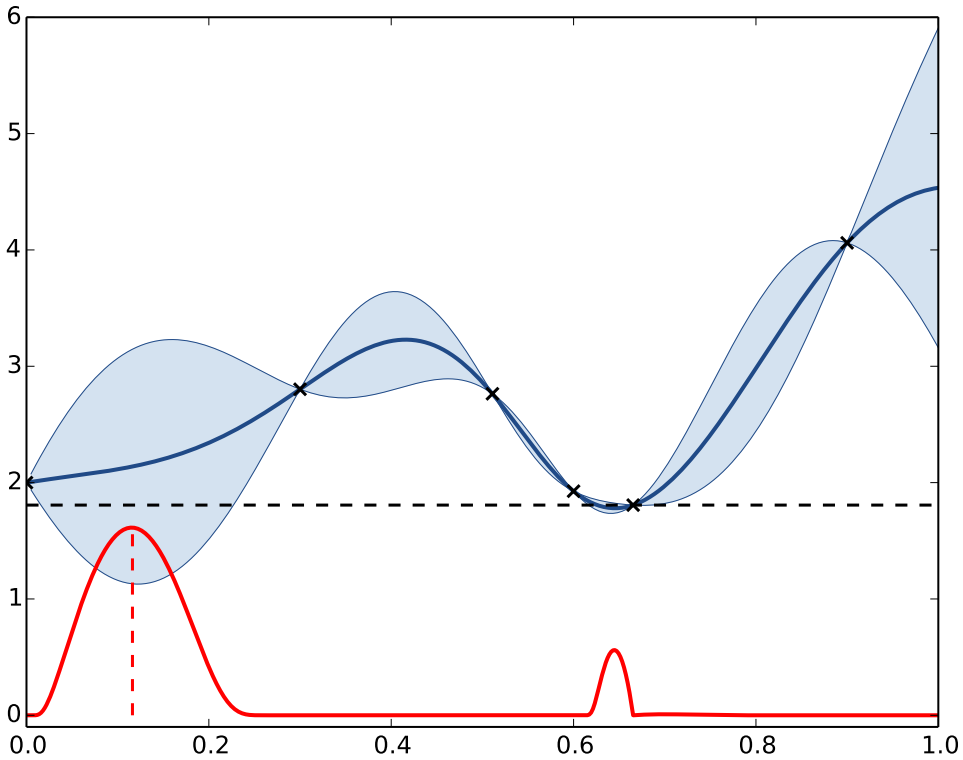
Expected Improvement

Let's see how it works... iteration 1



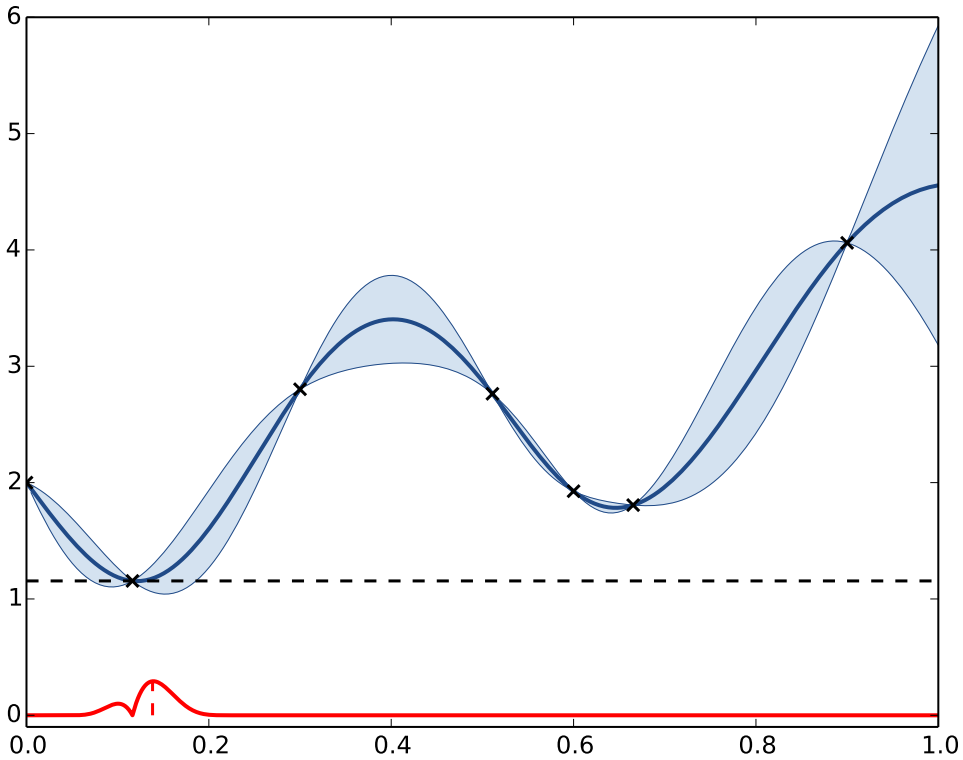
Expected Improvement

Let's see how it works... iteration 2



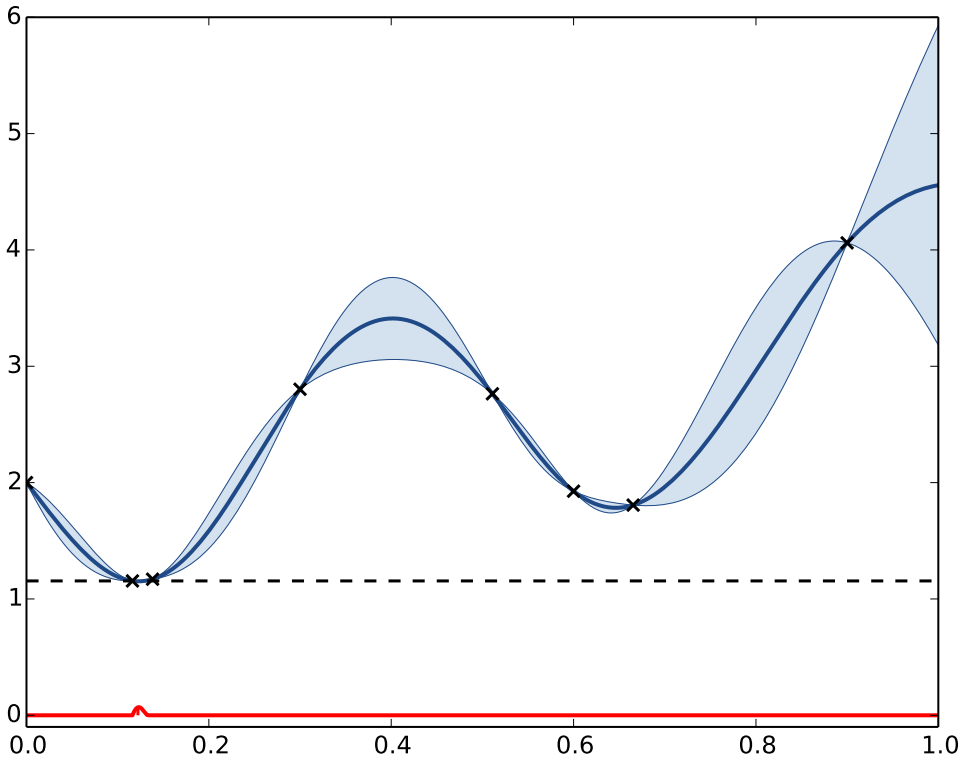
Expected Improvement

Let's see how it works... iteration 3



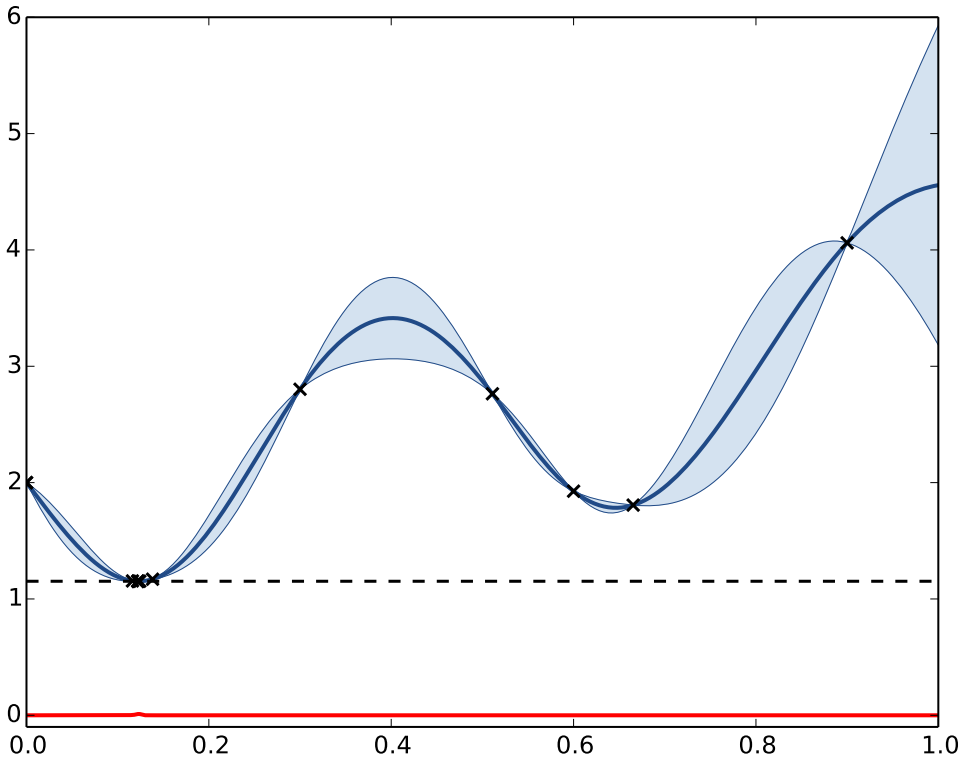
Expected Improvement

Let's see how it works... iteration 4



Expected Improvement

Let's see how it works... iteration 5



Expected Improvement

This algorithm is called **Efficient Global Optimization** (EGO, Jones et al., 1998):

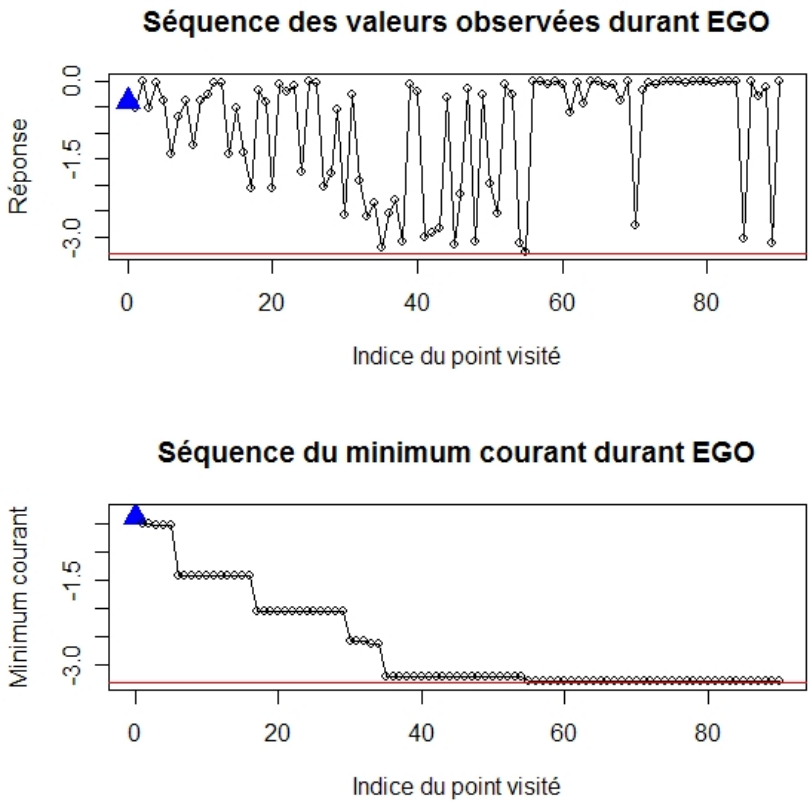
1. make an initial design of experiments X and calculate the associated F, $t = \texttt{length}(F)$
2. built a GP from (X, F) (max. log-likelihood on σ and θ_i's)
3. $X_{t+1} = \arg\max_x EI(x)$
4. calculate $F_{t+1} = f(X_{t+1})$, increment t
5. stop ($t > t^{\max}$) or go to 2.

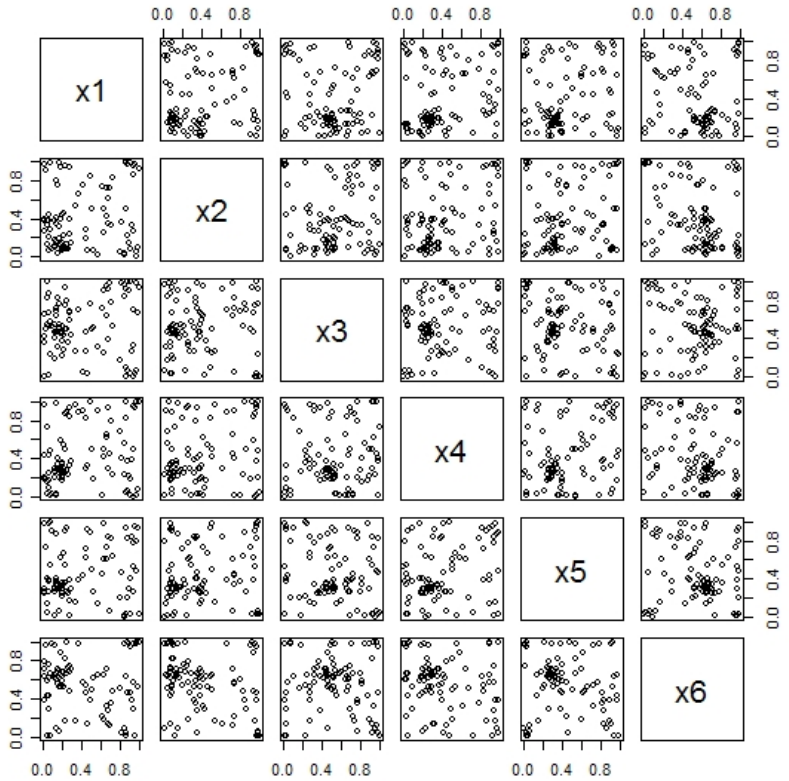
\+ EGO provides a good trade-off between exploitation and exploration without arbitrary parameters.

\+ It requires few function observations (10 in the example) to get close to optimal regions.

Illustration for $d = 6$ (Hartman)

Illustration in higher dimension





Source: *DiceOptim*, Roustant, Ginsbourger and Deville, 2009.

Difficulties and challenges with EGO

- Standard GPs are limited to $n \approx 1000$ points (covariance matrix inversion).
- EGO clusters points in good regions, the covariance matrix may become ill-conditionned if length scales θ_i are too large w.r.t. X.
- Although the method perfectly applies to large dimensional spaces ($d > 100$), larger d may require larger n, back 2 lines above.
- EGO does not converge in the traditional sense: it creates dense samples in the volume of S. The efficiency comes from the order in which points are sampled.

$\Rightarrow$ these are the topics of current research. Let's mention a few extensions next.

EGO continuations

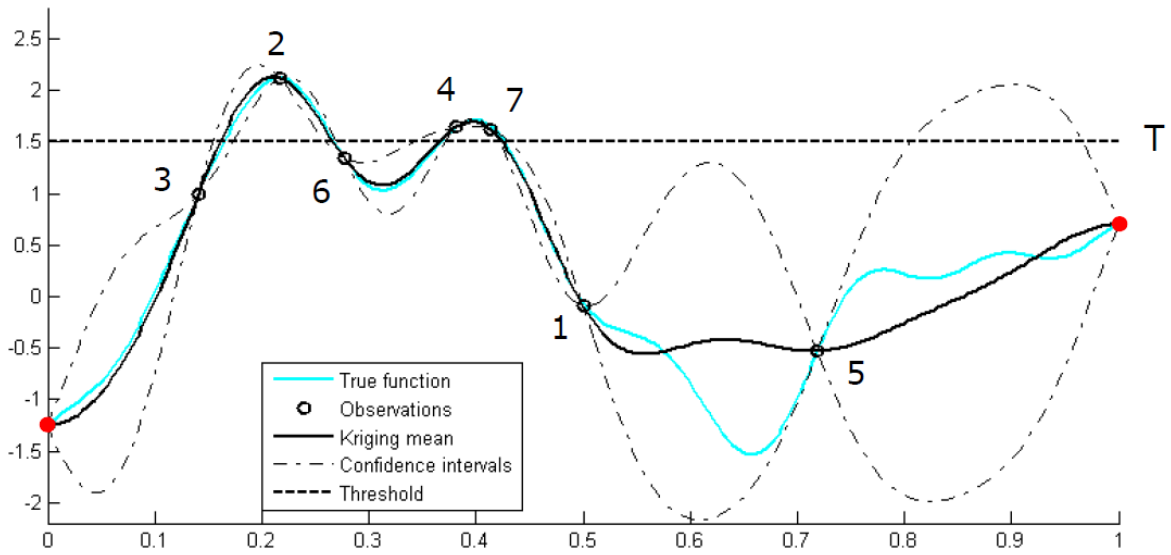
- Parallelized EGO: estimate the *EI* of groups of points, cf. Ginsbourger et al.
- Finite budget: *EI* of a single x is only optimal at the last iteration. Theory of dynamic *EI*, cf. Ginsbourger et al.
- EGO and bad covariance matrix conditioning: replace points that are close-by by one point and the associated derivatives (cf. M. Osborn, L. Laurent), regularizations (cf. Le Riche et al.)
- SUR strategies: (Step-wise Uncertainty Reduction), reduce the entropy of the optimum (cf. Vasquez et al.), or the average probability of incursions below $min(F)$ (cf. Picheny).

Related problems addressed with GPs

- EGO with constraints: $\min_x f(x)$ s.t. $g(x) \leq 0$, multiply the EI by the probability of constraints satisfaction.
- EGO with fixed budget or parallel EGO, see work from D. Ginsbourger et al.
- GP for multi-objective optimization: $\min_x \{f_1(x), \dots f_m(x)\}$, cf. Binois et al.
- GP for target attainment: find the set of x s.t. $f(x) = T$, change the EI into $c(x,x) \times \text{pdf}\left((T - m(x))/sqrt(c(x,x))\right)$, cf. Picheny et al.
- GP for probability estimation: find $\mathbb{P}(f(x,U) \leq T)$ where U is a random vector.

Example:

There are algorithms in the same spirit as EGO for inverse problems (calibration) and/or probability estimation:



See work from V. Picheny et al...

Kernel Design

Making new from old: Many operations can be applied to psd functions while retaining this property

Kernels can be:

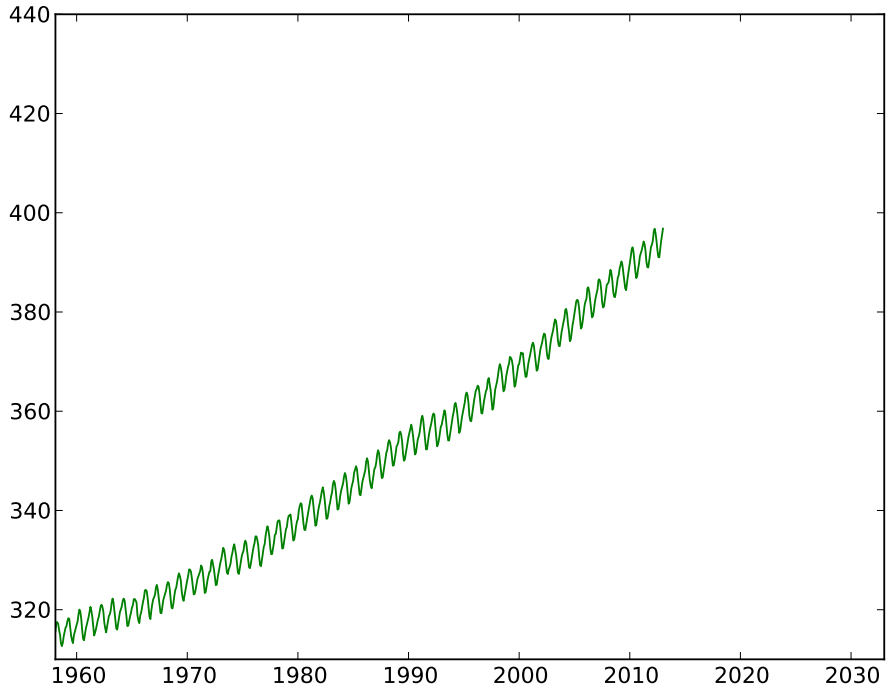
- Summed together
  - On the same space $k(x, y) = k_1(x, y) + k_2(x, y)$
  - On the tensor space $k(x, y) = k_1(x_1, y_1) + k_2(x_2, y_2)$
- Multiplied together
  - On the same space $k(x, y) = k_1(x, y) \times k_2(x, y)$
  - On the tensor space $k(x, y) = k_1(x_1, y_1) \times k_2(x_2, y_2)$
- Composed with a function
  - $k(x, y) = k_1(f(x), f(y))$

How can this be useful?

Sum of kernels over the same space

Example (The Mauna Loa observatory dataset)

This famous dataset compiles the monthly CO_2 concentration in Hawaii since 1958.

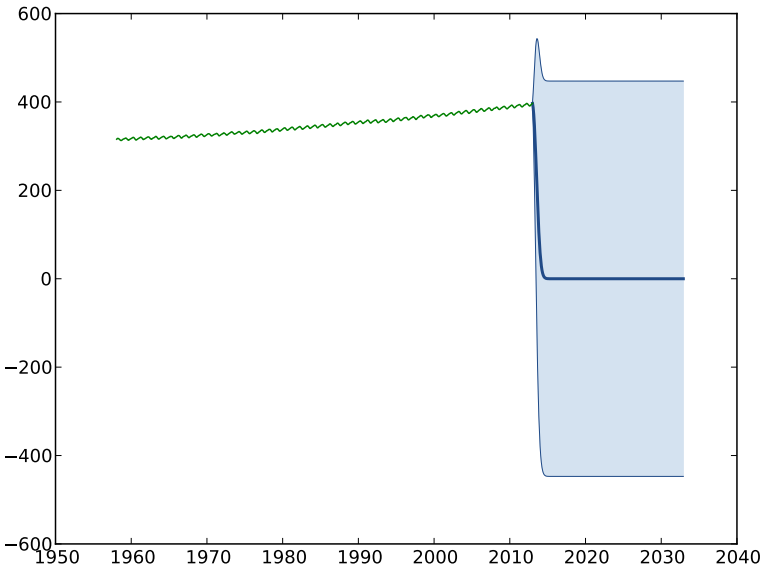


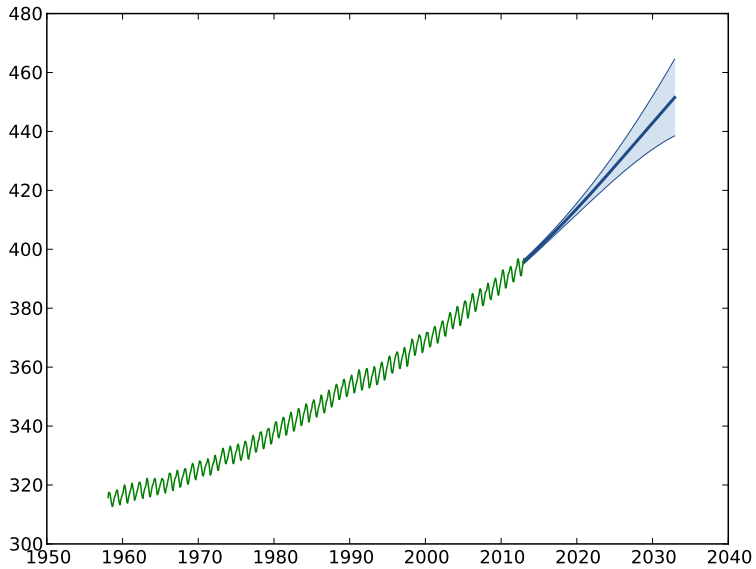
Let's try to predict the concentration for the next 20 years.

Sum of kernels over the same space

We first consider a squared-exponential kernel:

$$k_{se}(x, y) = \sigma^2 \exp\left(-\frac{(x-y)^2}{\theta^2}\right)$$





The results are terrible!

Sum of kernels over the same space

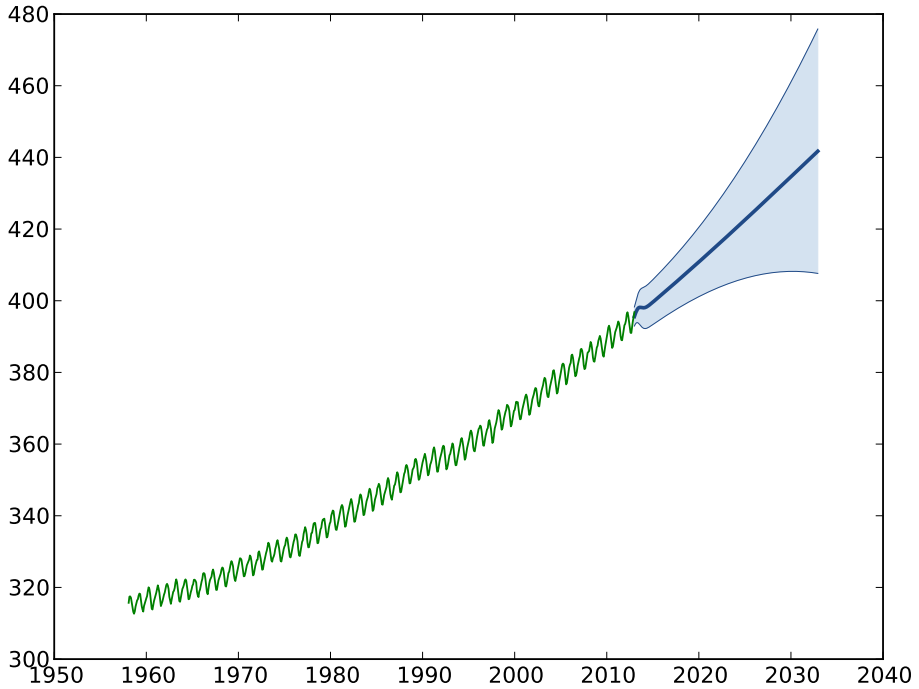
What happen if we sum both kernels?

$$k(x, y) = k_{se1}(x, y) + k_{se2}(x, y)$$

Sum of kernels over the same space

What happen if we sum both kernels?

$$k(x, y) = k_{se1}(x, y) + k_{se2}(x, y)$$



The model is drastically improved!

Sum of kernels over the same space

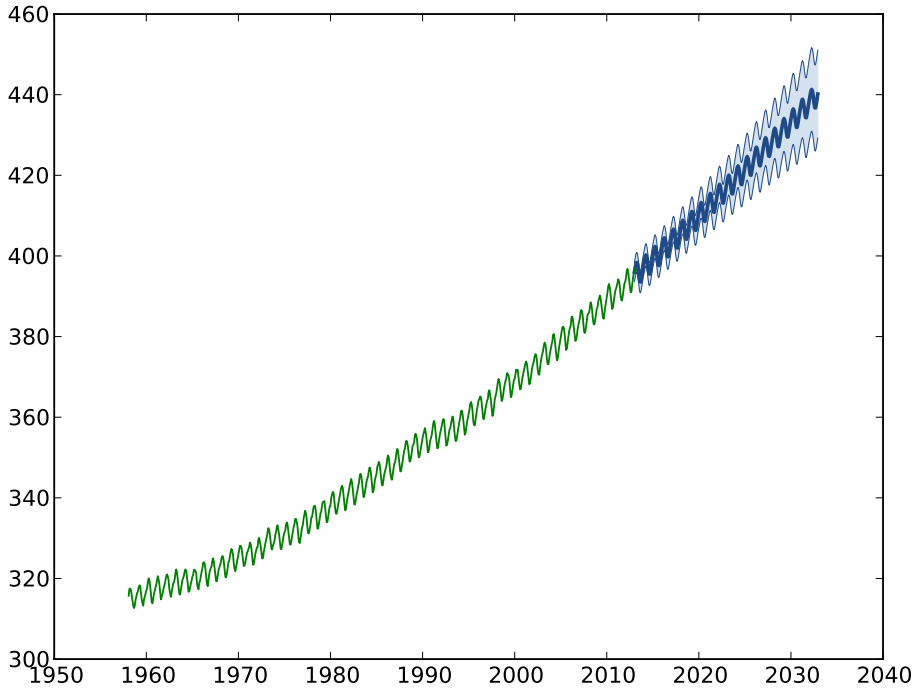
We can try the following kernel:

$$k(x, y) = \sigma_0^2 x^2 y^2 + k_{se1}(x, y) + k_{se2}(x, y) + k_{per}(x, y)$$

Sum of kernels over the same space

We can try the following kernel:

$$k(x, y) = \sigma_0^2 x^2 y^2 + k_{se1}(x, y) + k_{se2}(x, y) + k_{per}(x, y)$$



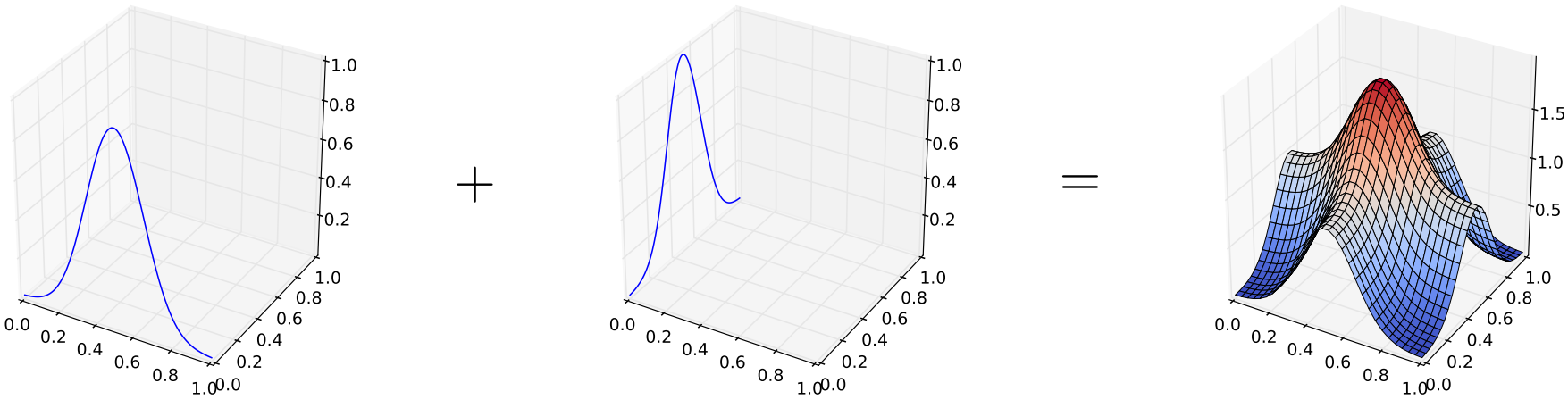
Once again, the model is significantly improved.

Sum of kernels over tensor space

Property

$$k(x, y) = k_1(x_1, y_1) + k_2(x_2, y_2)$$

is valid covariance structure.

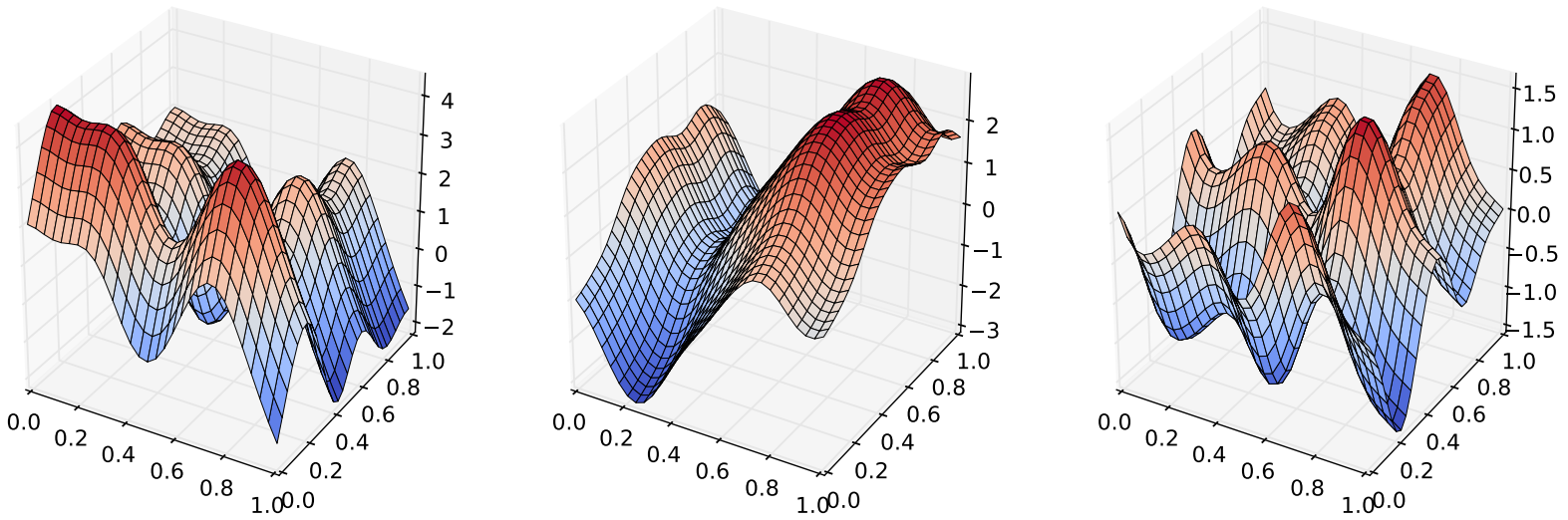


Remark: From a GP point of view, k is the kernel of

$$Z(x) = Z_1(x_1) + Z_2(x_2)$$

Sum of kernels over tensor space

We can have a look at a few sample paths from Z:



$\Rightarrow$ They are additive (up to a modification)

Tensor Additive kernels are very useful for

- Approximating additive functions
- Building models over high dimensional inputs spaces

Product over the same space

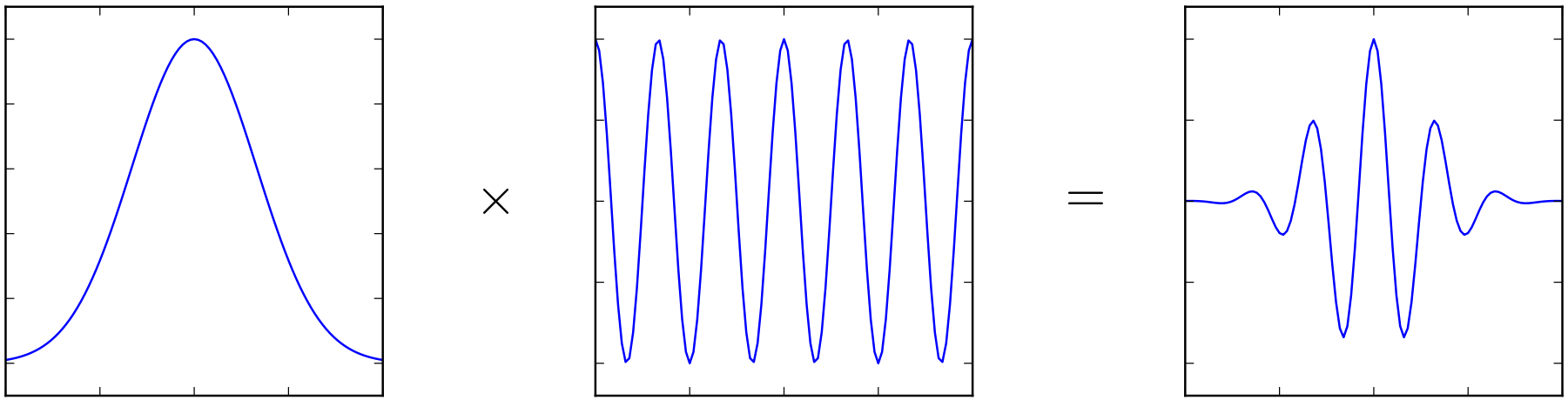
Property

$$k(x, y) = k_1(x, y) \times k_2(x, y)$$

is valid covariance structure.

Example

We consider the product of a squared exponential with a cosine:



Product over the tensor space

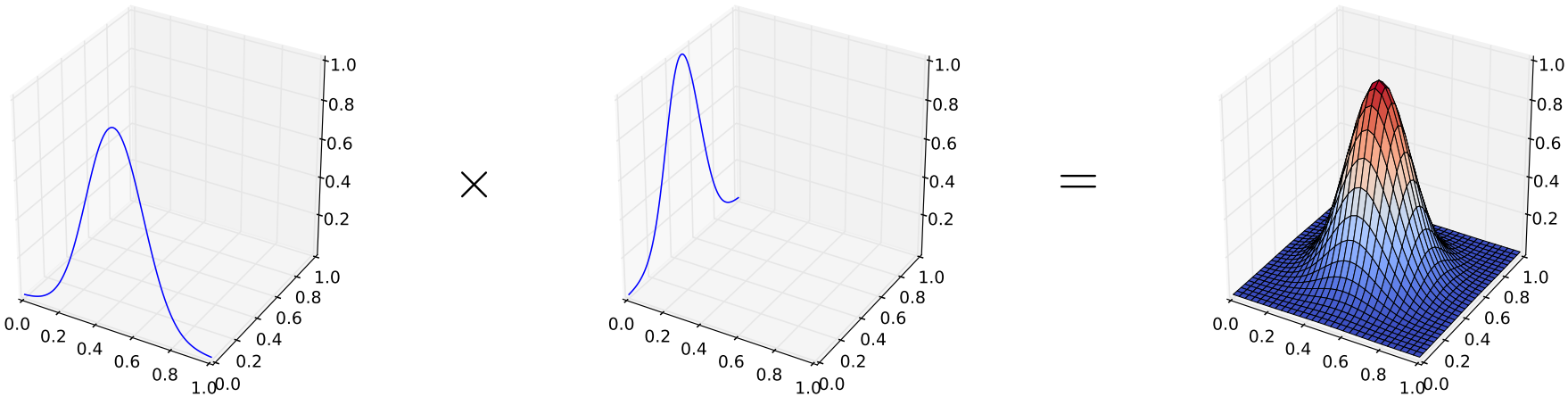
Property

$$k(x, y) = k_1(x_1, y_1) \times k_2(x_2, y_2)$$

is valid covariance structure.

Example

We multiply 2 squared exponential kernel



Calculation shows this is the usual 2D squared exponential kernel.

Composition with a function

Property

Let k_1 be a kernel over $D_1 \times D_1$ and f be an arbitrary function $D \to D_1$, then

$$k(x, y) = k_1(f(x), f(y))$$

is a kernel over $D \times D$.

proof

$$\sum\sum a_i a_j k(x_i, x_j) = \sum\sum a_i a_j k_1(\underbrace{f(x_i)}_{y_i}, \underbrace{f(x_j)}_{y_j}) \geq 0$$

Remarks:

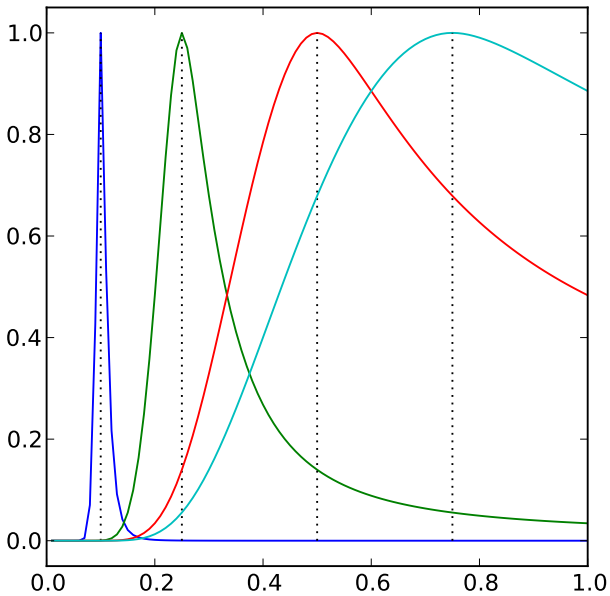
- k corresponds to the covariance of $Z(x) = Z_1(f(x))$
- This can be seen as a (non-linear) rescaling of the input space

Example

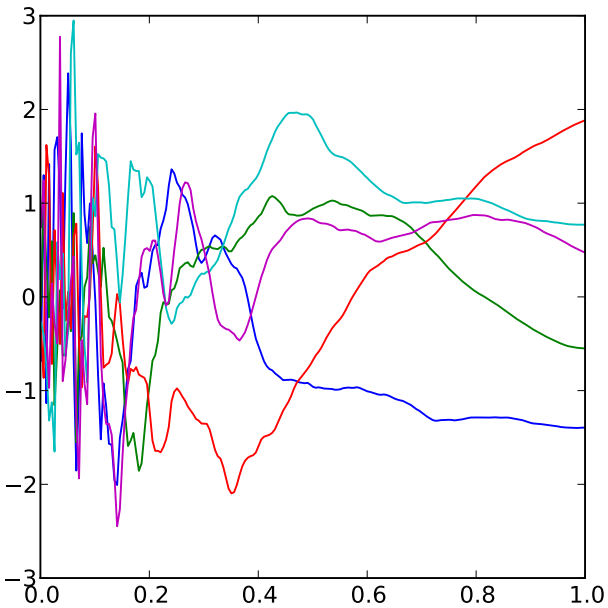
We consider $f(x) = \frac{1}{x}$ and a Matérn 3/2 kernel $k_1(x, y) = (1 + |x - y|)e^{-|x-y|}$.

We obtain:

Kernel



Sample paths



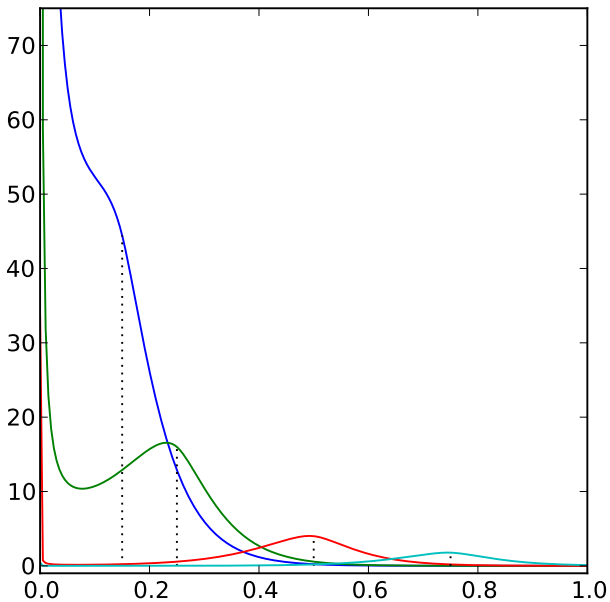
All these transformations can be combined!

Example

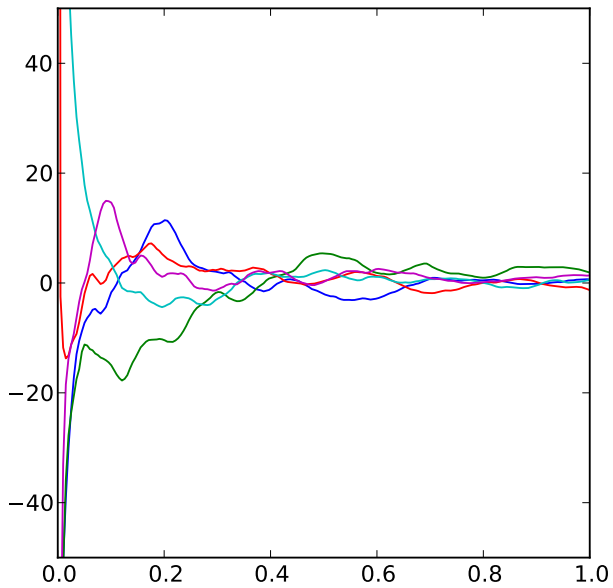
$k(x, y) = f(x)f(y)k_1(x, y)$ is a valid kernel.

This can be illustrated with $f(x) = \frac{1}{x}$ and $k_1(x, y) = (1 + |x - y|)e^{-|x-y|}$:

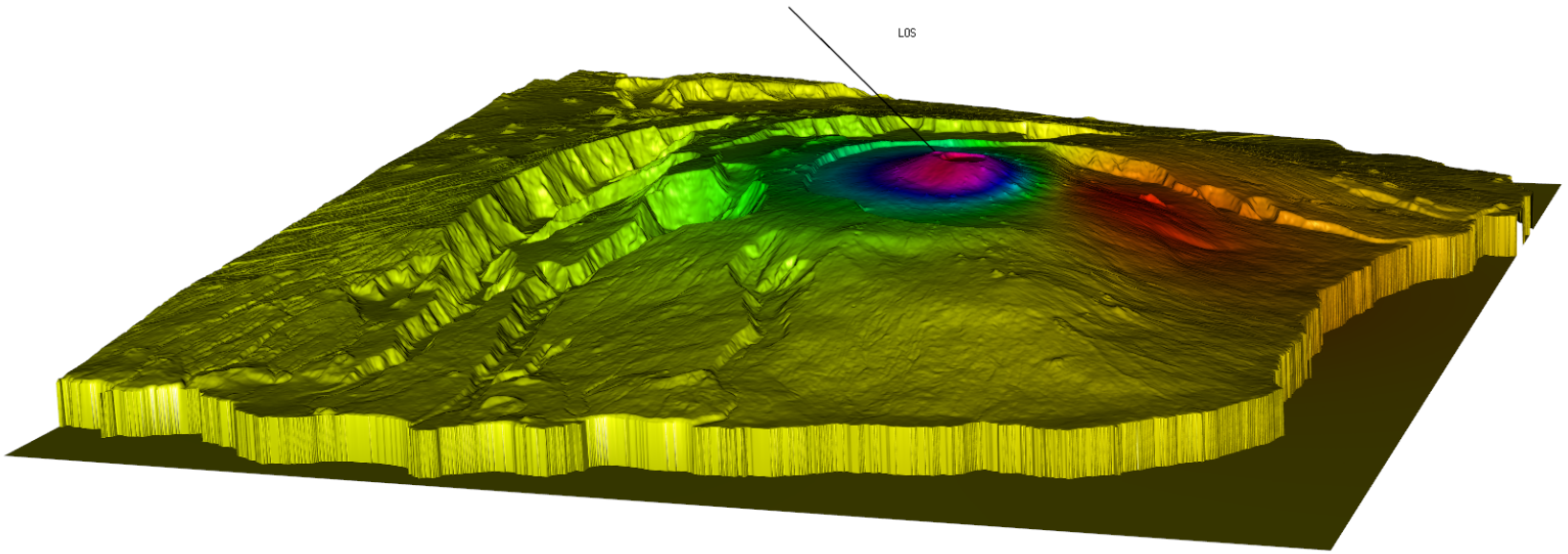
Kernel



Sample paths



Final example:



$\Rightarrow$ R demo