MDIS Fall School 2017

Introduction to Gaussian Process Surrogate models

Clermont-Ferrand, Besse-en-Chandesse, 16th of October 2017

Nicolas Durrande, PROWLER.io (nicolas@prowler.io)
Rodolphe Le Riche, CNRS LIMOS – Mines St-Étienne (leriche@emse.fr)

Statistical models in engineering and physics?

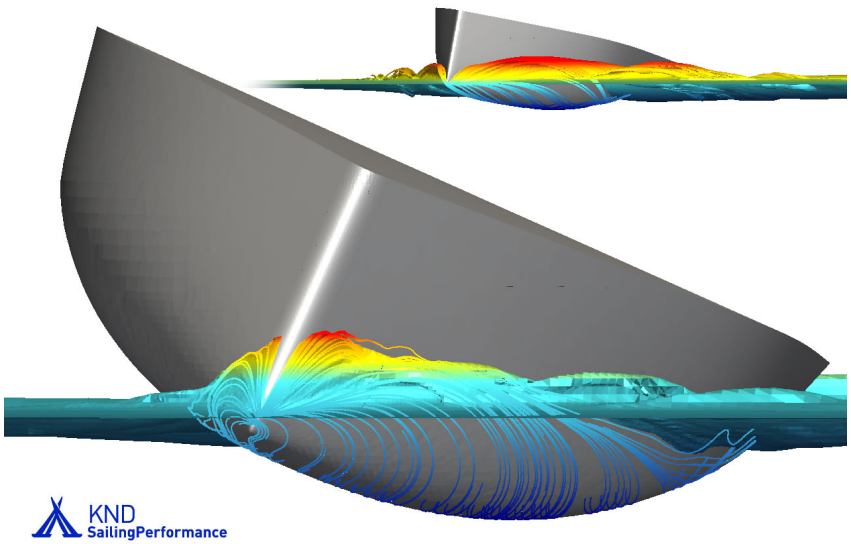
There is a wide variety of situations where getting data about a system performance can be extremely expensive:

- real world experiments in particular
  - destructive tests
  - prototyping
- and even numerical experiments (e.g. non linear, with uncertainties, ...): Numerical experiments are less expensive but can be very time consuming! (expl.: 15 min / execution, 5 parameters, a grid with 10 levels, total time $= 10^5 \times 15$ min $>$ 2 years and 10 months)

Example: boat hull design

real

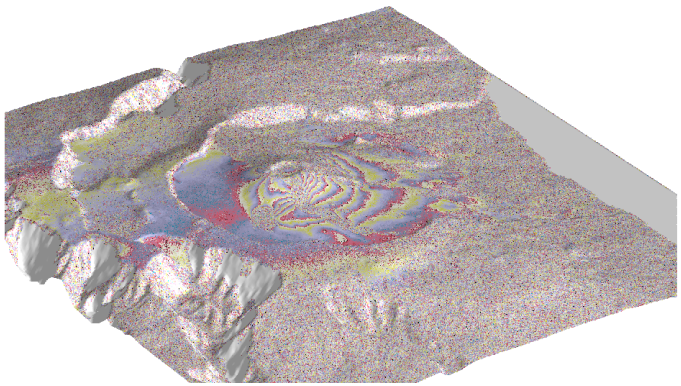
numerical

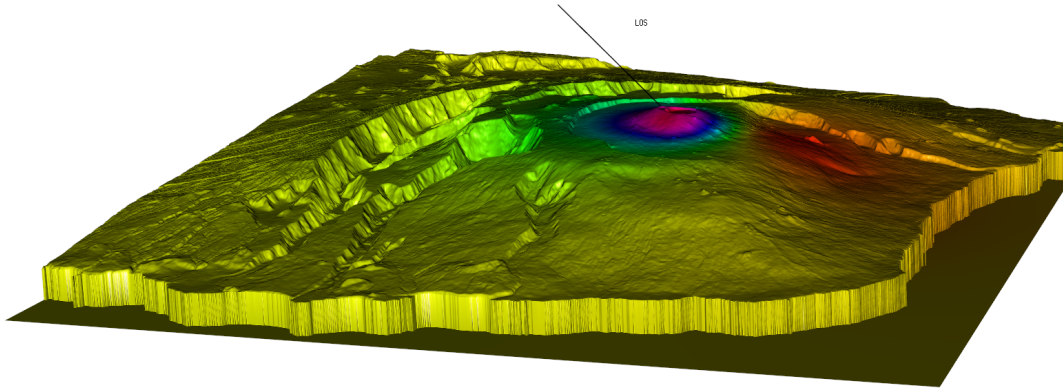


Example: volcano internal structure identification

real

numerical





displacements resulting from 5 dike intrusions

in the Piton de la Fournaise between 1998

and 2000

In all these cases, the quantity of interest (drag, misfit ...) can be seen as a function of the input parameters

$$y = f(x).$$

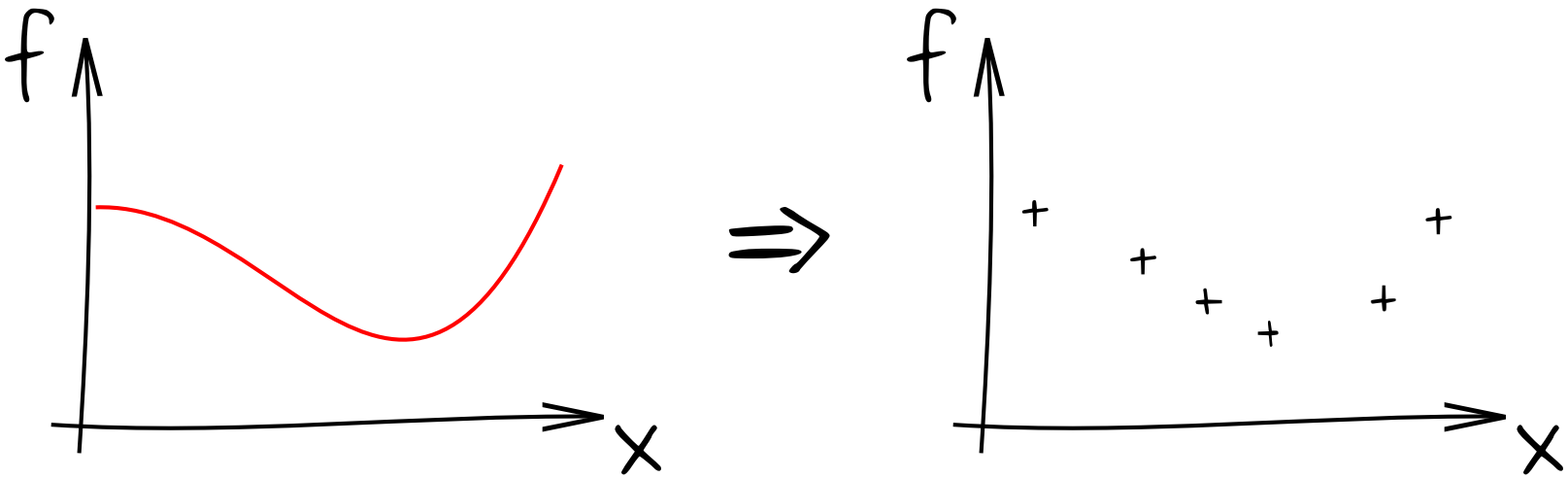
where f is a **costly to evaluate function**.

In the following, we will assume that

- $x \in \mathbb{R}^d$: There are d input variables,
- $y \in \mathbb{R}$: The output is a scalar.

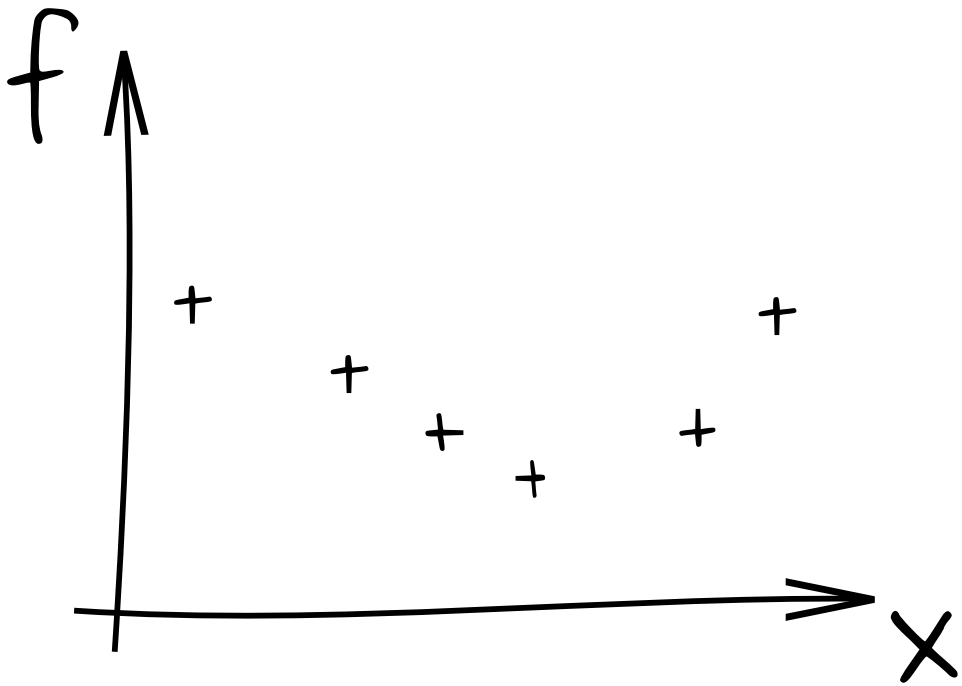
The fact that f is **costly to evaluate** changes a lot of things...

1. Representing the function is not possible...



The fact that f is **costly to evaluate** changes a lot of things...

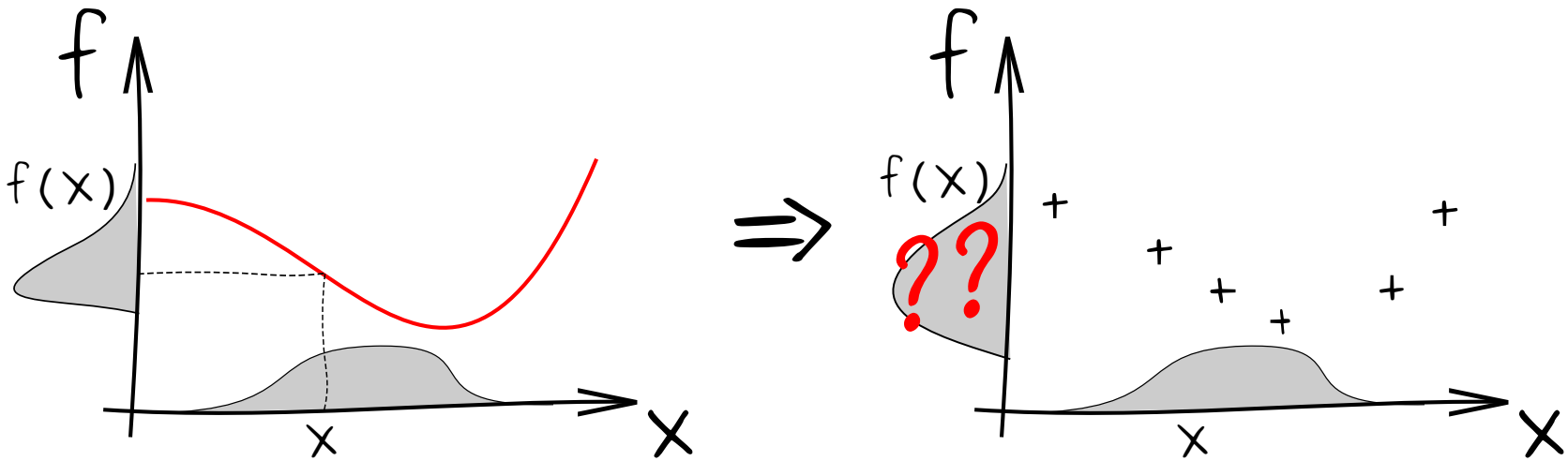
2. Computing integrals is not possible...



What is the mean value of f?

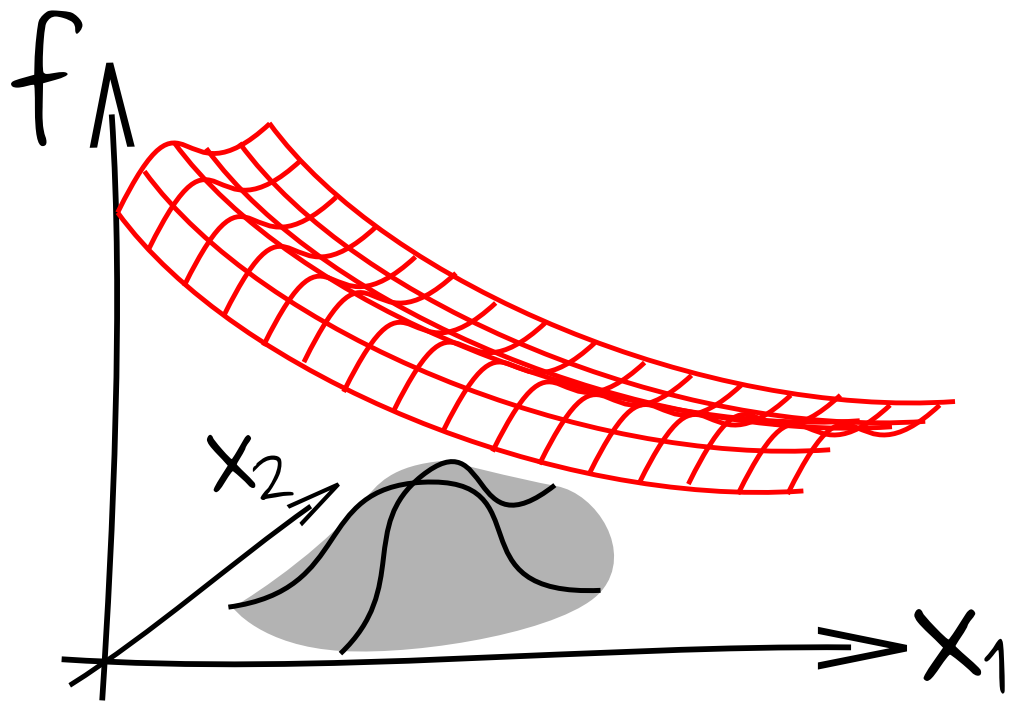
The fact that f is **costly to evaluate** changes a lot of things...

3. Uncertainty propagation is not possible...



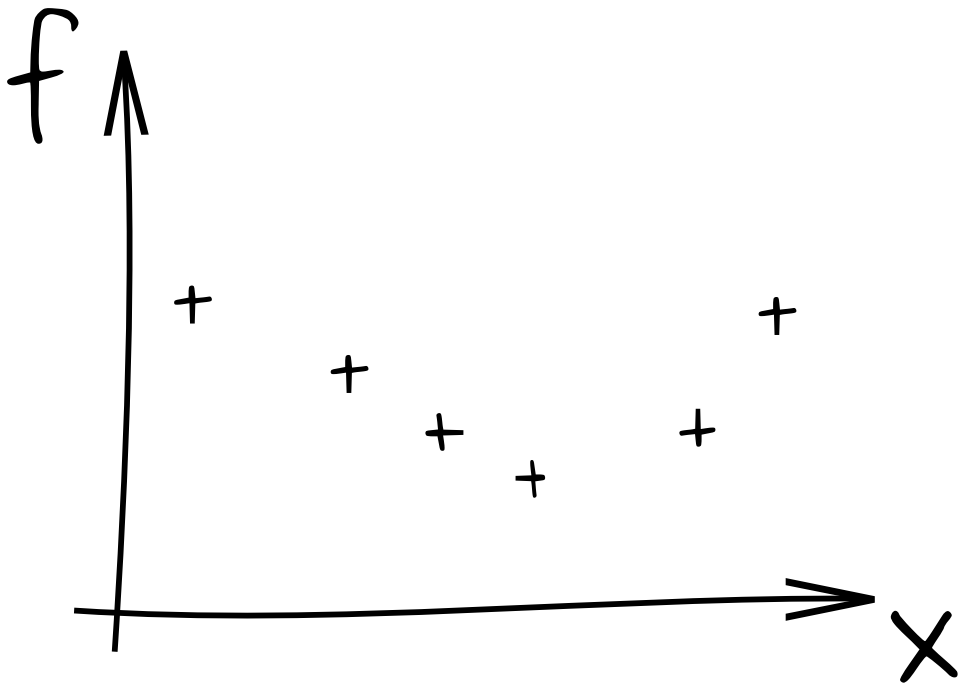
The fact that f is **costly to evaluate** changes a lot of things...

4. Sensitivity analysis is not possible...

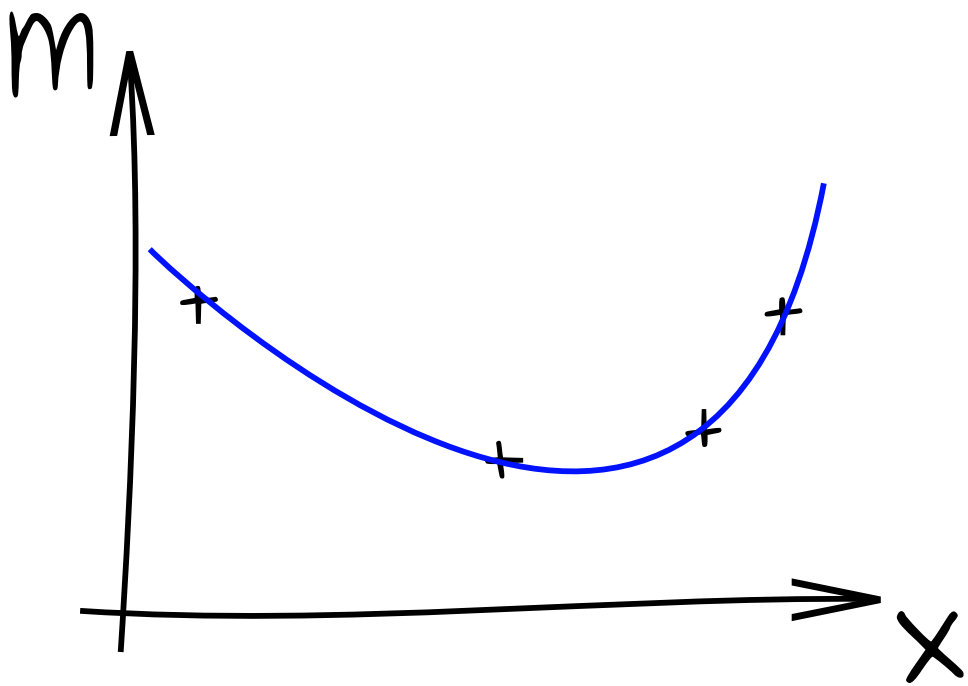


The fact that f is **costly to evaluate** changes a lot of things...

5. Optimisation is also tricky...

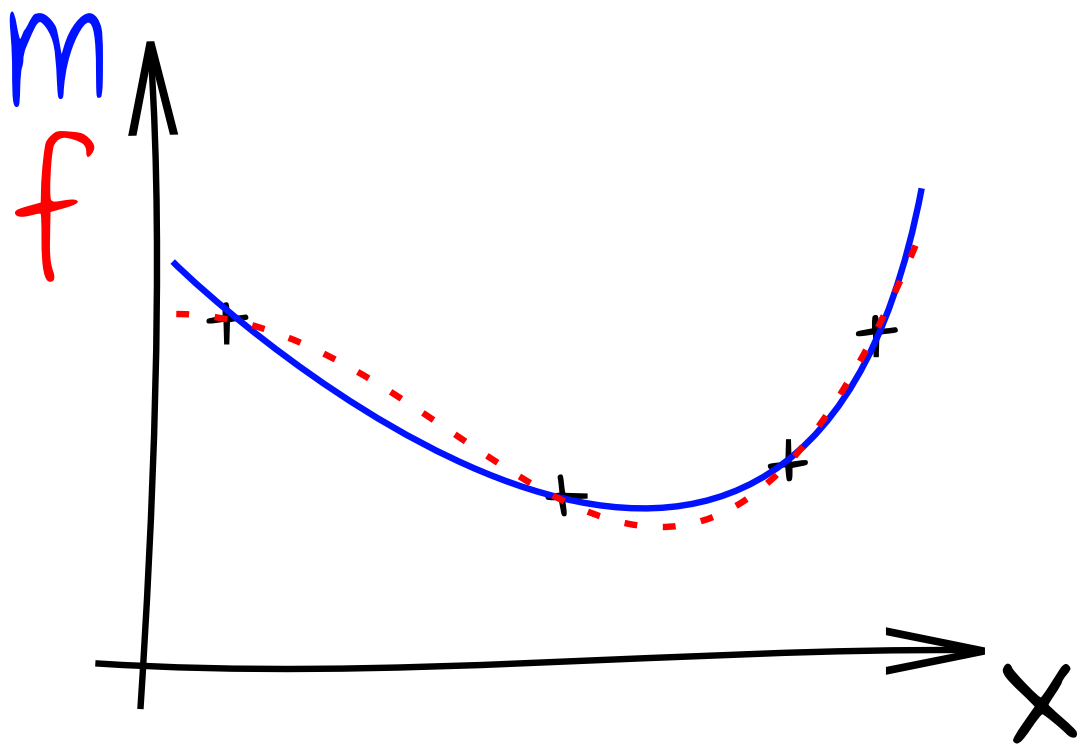


The principle of statistical modelling is to use the data to build a mathematical approximation of the function.

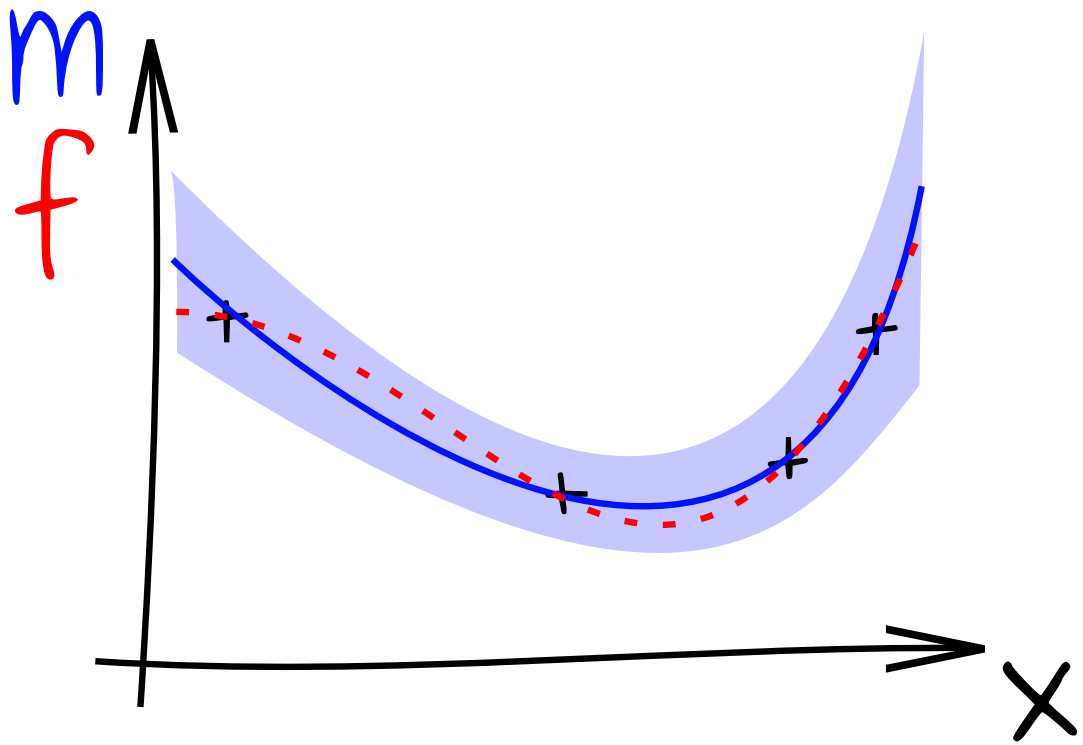


The model can then be used to answer all previous questions

Of course, there is a difference between f and m...



Why **statistical models**?
We want to be able to quantify the model error:



The confidence intervals can be used to obtain a **measure of uncertainty on the value of interest**.

In the sequel, we will use the following notations :

- The set of observation points will be represented by a $n \times d$ matrix $X = (X_1, ..., X_n)^t$
- The vector of observations will be denoted by F : $F_i = f(X_i)$ (or $F = f(X)$).

There are many surrogate methods available on the market

- Linear regression
- Smoothing splines
- Gaussian process regression
- Neural Networks
- ...

In this course we will focus on **Gaussian Process Regression**.

Our breadcrumb example: **identification of a spherical magma reservoir** from measured displacements along a satellite line of sight.

Gaussian Process Regression

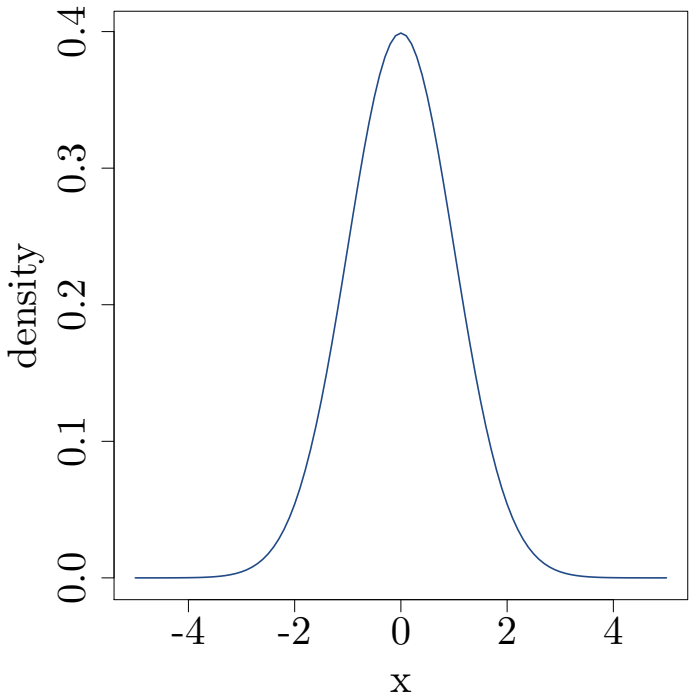
This section is be organised in 3 subsections:

1. Univariate and multivariate normal distributions
2. Gaussian processes
3. Gaussian process regression

1D normal distribution

We say that $X \sim \mathcal{N}(\mu, \sigma^2)$ if it has the following pdf:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



The distribution is characterised by

mean: $\mu = \mathrm{E}[X]$

variance: $\sigma^2 = \mathrm{E}[X^2] - \mathrm{E}[X]^2$

One fundamental property: a linear combination of independant normal distributed random variables is still normal distributed.

Multivariate normal distribution

Definition

We say that a vector $Y = (Y_1, \ldots, Y_d)^t$ follows a multivariate normal distribution if any linear combination of Y follows a normal distribution:

$$\forall \alpha \in \mathbb{R}^d,\ \alpha^t Y \sim \mathcal{N}$$

The distribution of a Gaussian vector is characterised by

- a **mean vector** $\mu = \mathrm{E}[Y]$
- a **covariance matrix** $\Sigma = \mathrm{E}[YY^t] - \mathrm{E}[Y]\,\mathrm{E}[Y]^t$

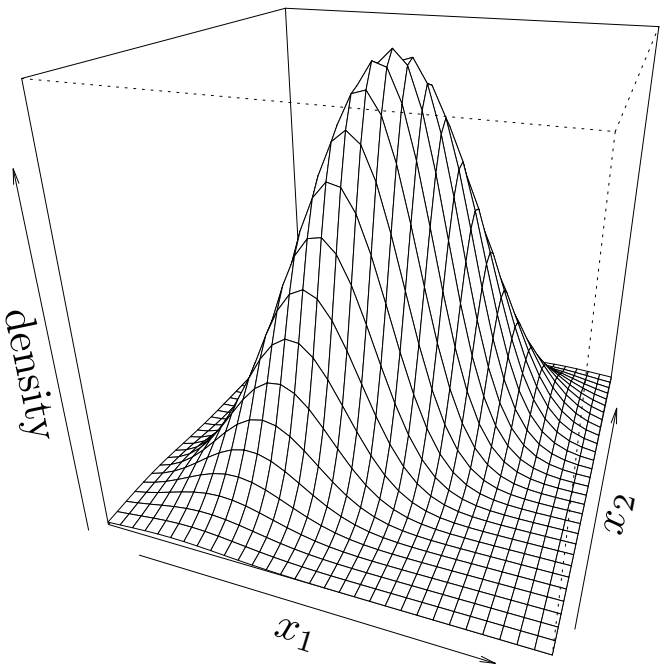
Property:

A covariance matrix is

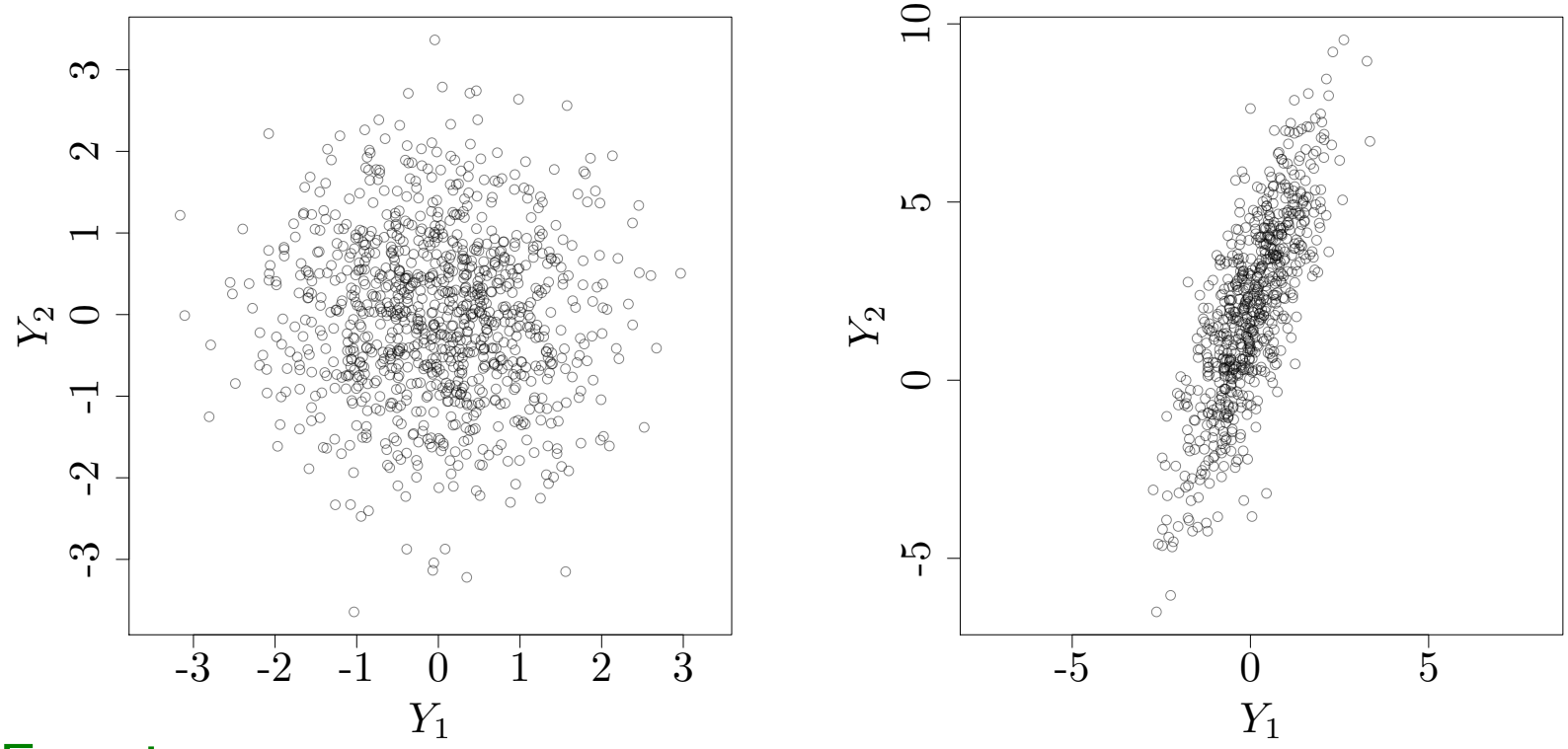
- symmetric $K_{i,j} = K_{j,i}$
- positive semi-definite $\forall \alpha \in \mathbb{R}^d, \alpha^t K \alpha \geq 0$.

The density of a multivariate Gaussian is:

$$f_Y(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)\right).$$



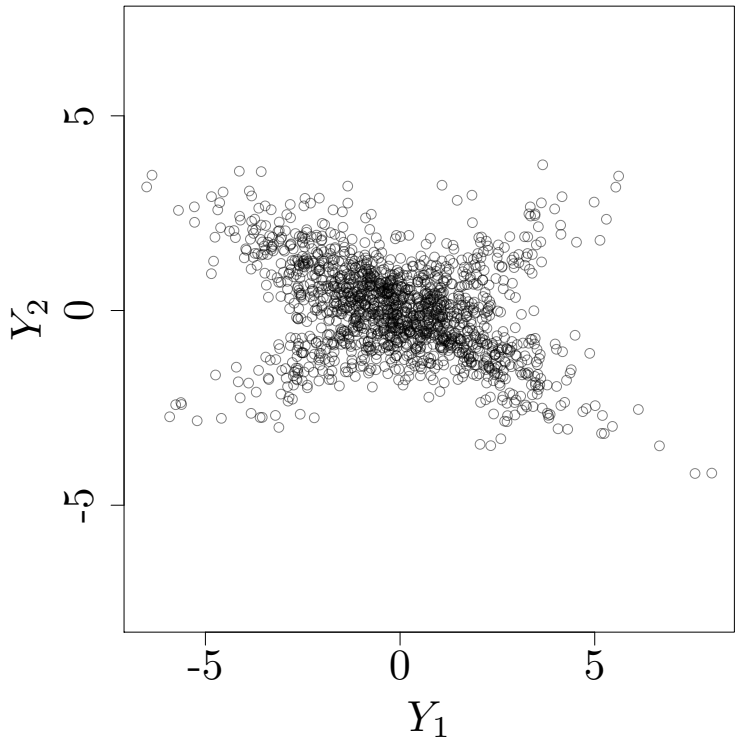
Samples from a multivariate normal



Exercise

- For $X = (X_1, \ldots, X_d)$ with X_i independant and $\mathcal{N}(0, 1)$, and a $d \times d$ matrix A, what is the distribution of AX?
- For a given covariance matrix K and independant $\mathcal{N}(0, 1)$ samples, how can we generate $\mathcal{N}(\mu, K)$ random samples?

Counter example



Y_1 and Y_2 are normally distributed but **the couple** (Y_1, Y_2) **is not**.

Conditional distribution

Let (Y, Z) be a Gaussian vector (Y and Z may both be vectors) with mean $(\mu_Y, \mu_Z)^t$ and covariance matrix

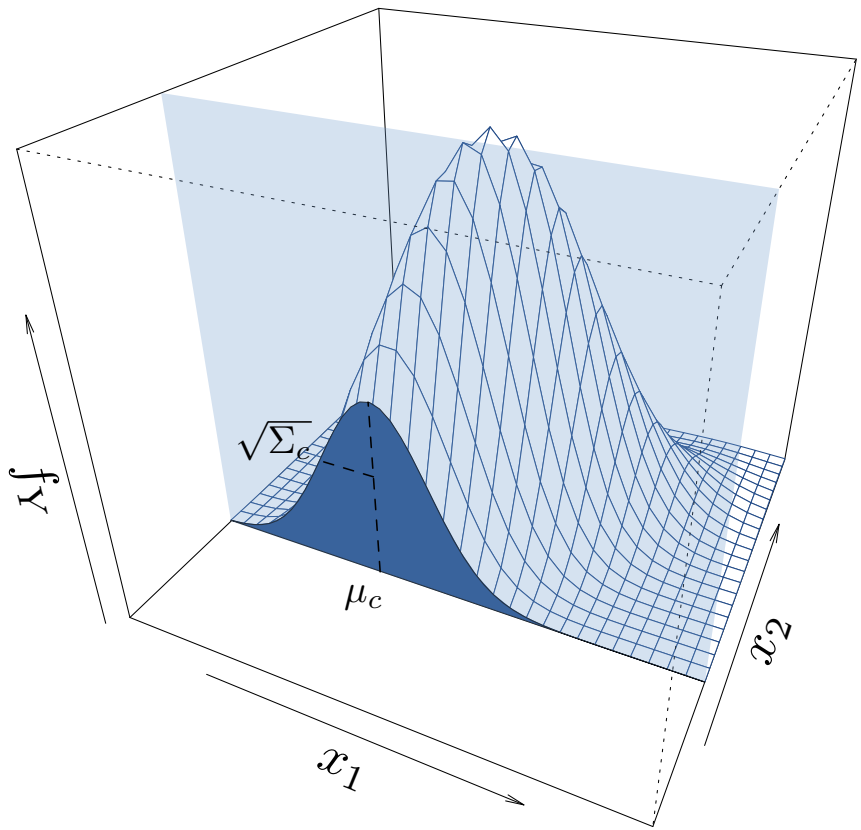
$$\begin{pmatrix} \text{cov}(Y,Y) & \text{cov}(Y,Z) \\ \text{cov}(Z,Y) & \text{cov}(Z,Z) \end{pmatrix}.$$

The conditional distribution of Y knowing Z is still multivariate normal $Y|Z \sim \mathcal{N}(\mu_{cond}, \Sigma_{cond})$ with

$$\mu_{cond} = \text{E}[Y|Z] = \mu_Y + \text{cov}(Y,Z)\,\text{cov}(Z,Z)^{-1}(Z - \mu_Z)$$
$$\Sigma_{cond} = \text{cov}[Y,Y|Z] = \text{cov}(Y,Y) - \text{cov}(Y,Z)\,\text{cov}(Z,Z)^{-1}\,\text{cov}(Z,Y)$$

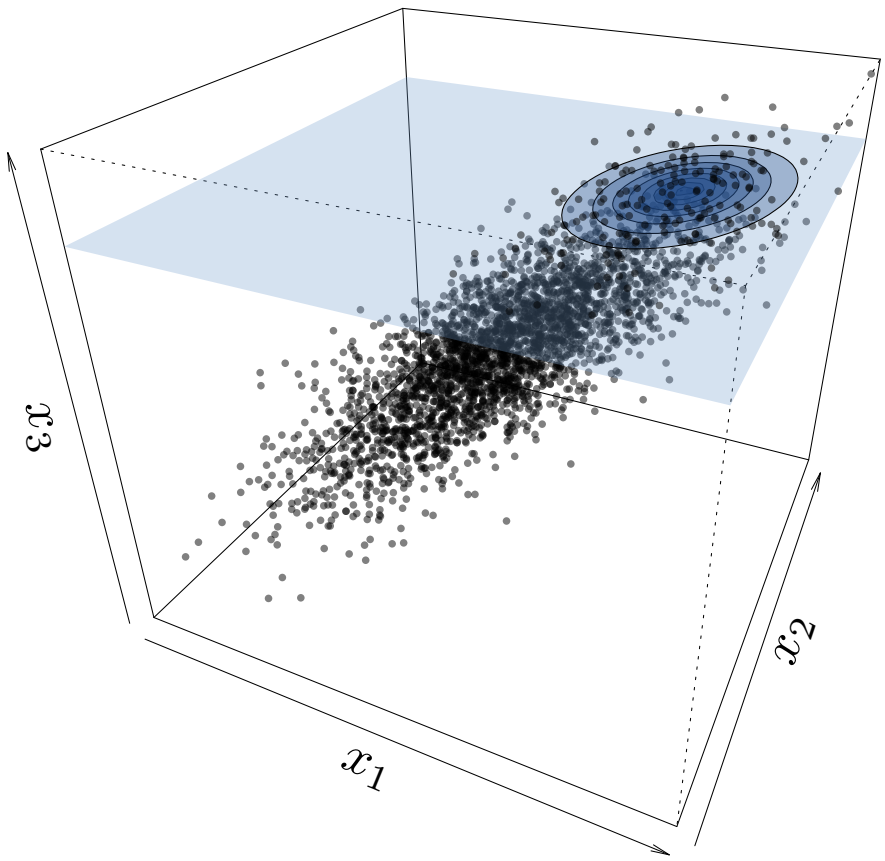
Example 1

2D multivariate Gaussian conditional distribution:



Example 2

3D multivariate Gaussian conditional distribution:



2. Gaussian processes

The multivariate Gaussian distribution can be generalised to random processes:

Definition

A random process Z over $D \subset \mathbb{R}^d$ is said to be Gaussian if

$$\forall n \in \mathbb{N}, \forall x_i \in D, (Z(x_1), \ldots, Z(x_n)) \text{ is a Gaussian vector.}$$

The distribution of a GP is fully characterised by:

- its mean function m defined over D
- its covariance function (or kernel) k defined over $D \times D$: $k(x, y) = \text{cov}(Z(x), Z(y))$

We will use the notation $Z \sim \mathcal{N}(m(.), k(.,.))$.

A kernel satisfies the following properties:

- It is symmetric: $k(x, y) = k(y, x)$
- It is positive semi-definite (psd):

$$\forall n \in \mathbb{N}, \forall x_i \in D, \forall \alpha \in \mathbb{R}^n, \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0$$

Furthermore any symmetric psd function can be seen as the covariance of a Gaussian process. This equivalence is known as the Loeve theorem.

There are a lot of functions that have already been proven psd:

constant	$k(x,y) = \sigma^2$
white noise	$k(x,y) = \sigma^2 \delta_{x,y}$
Brownian	$k(x,y) = \sigma^2 \min(x,y)$
exponential	$k(x,y) = \sigma^2 \exp\left(-\lvert x-y\rvert/\theta\right)$
Matern 3/2	$k(x,y) = \sigma^2 \left(1+\lvert x-y\rvert\right) \exp\left(-\lvert x-y\rvert/\theta\right)$
Matern 5/2	$k(x,y) = \sigma^2 \left(1+\lvert x-y\rvert/\theta + 1/3\lvert x-y\rvert^2/\theta^2\right) \exp\left(-\lvert x-y\rvert/\theta\right)$
squared exponential	$k(x,y) = \sigma^2 \exp\left(-(x-y)^2/\theta^2\right)$
$\vdots$	

When k is a function of $x - y$, the kernel is called **stationary**.
The parameter σ^2 is called the **variance** and θ the **length-scale**.

Can we look at the sample paths associated to these kernels?

In order to simulate sample paths from a GP $Z \sim \mathcal{N}(m(.), k(.,.))$, we will consider samples of the GP discretised on a fine grid.

Exercise: Simulating sample paths

Let X be a set 100 regularly spaced points over the input space of Z.

- What is the distribution of $Z(X)$?
- How to simulate samples from $Z(X)$?

$\Rightarrow$ Shiny App:
`https://github.com/NicolasDurrande/shinyApps`

In higher dimension one can introduce one length-scale parameter per dimension. The usual Euclidean distance between two points $||x - y|| = (\sum(x_i - y_i)^2)^{1/2}$ is thus replaced by

$$||x - y||_\theta = \left(\sum_{i=1}^{d} \frac{(x_i - y_i)^2}{\theta_i^2} \right)^{1/2}.$$

If the parameters θ_i are equal for all the dimensions, the covariance (or the process) is called **isotropic**.

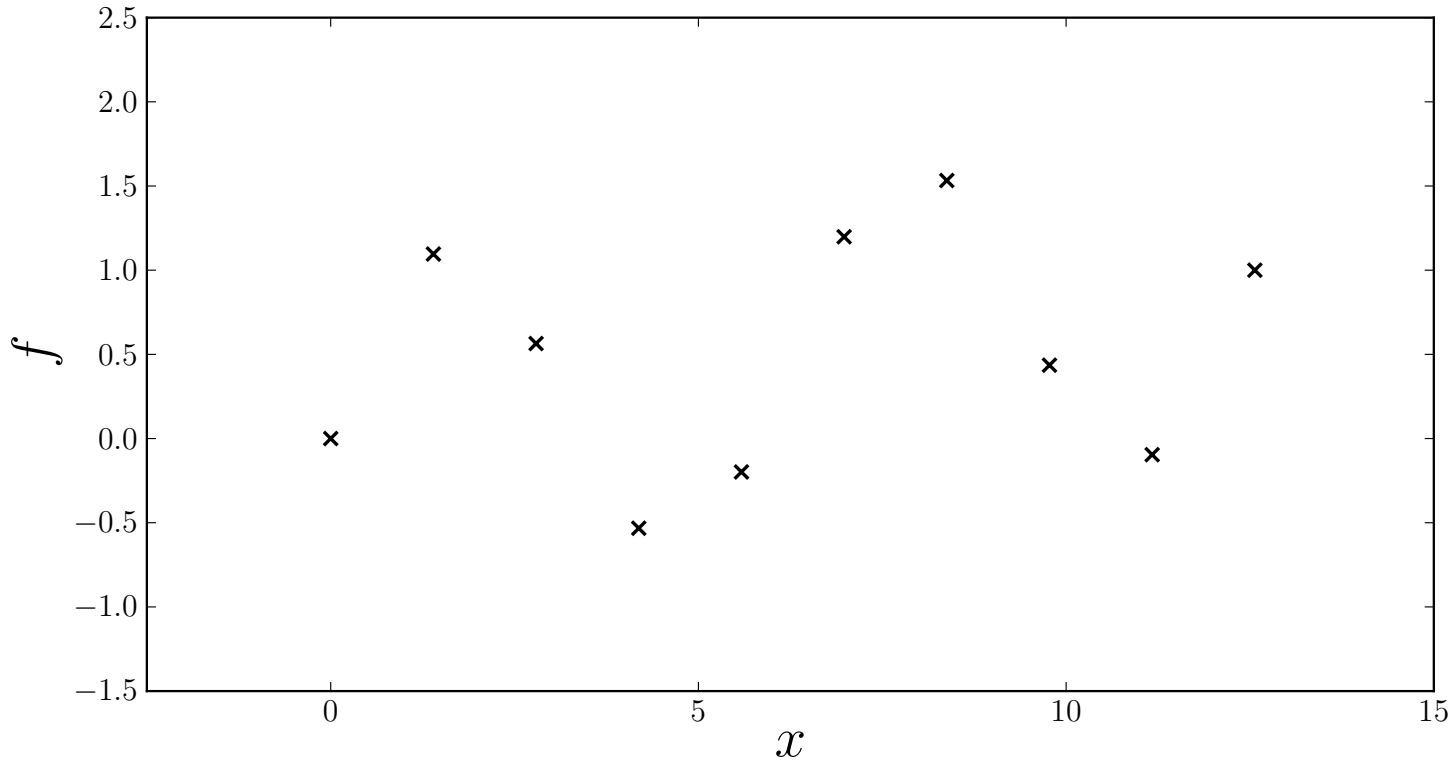
$\Rightarrow$ R demo

Here is a list of the most common kernels:

constant	$k(x,y) = \sigma^2$
white noise	$k(x,y) = \sigma^2 \delta_{x,y}$
exponential	$k(x,y) = \sigma^2 \exp\left(-\|\|x-y\|\|_\theta\right)$
Matern 3/2	$k(x,y) = \sigma^2 \left(1+\sqrt{3}\|\|x-y\|\|_\theta\right) \exp\left(-\sqrt{3}\|\|x-y\|\|_\theta\right)$
Matern 5/2	$k(x,y) = \sigma^2 \left(1+\sqrt{5}\|\|x-y\|\|_\theta + \frac{5}{3}\|\|x-y\|\|_\theta^2\right) \exp\left(-\sqrt{5}\|\|x-y\|\|_\theta\right)$
Gaussian	$k(x,y) = \sigma^2 \exp\left(-\frac{1}{2}\|\|x-y\|\|_\theta^2\right)$

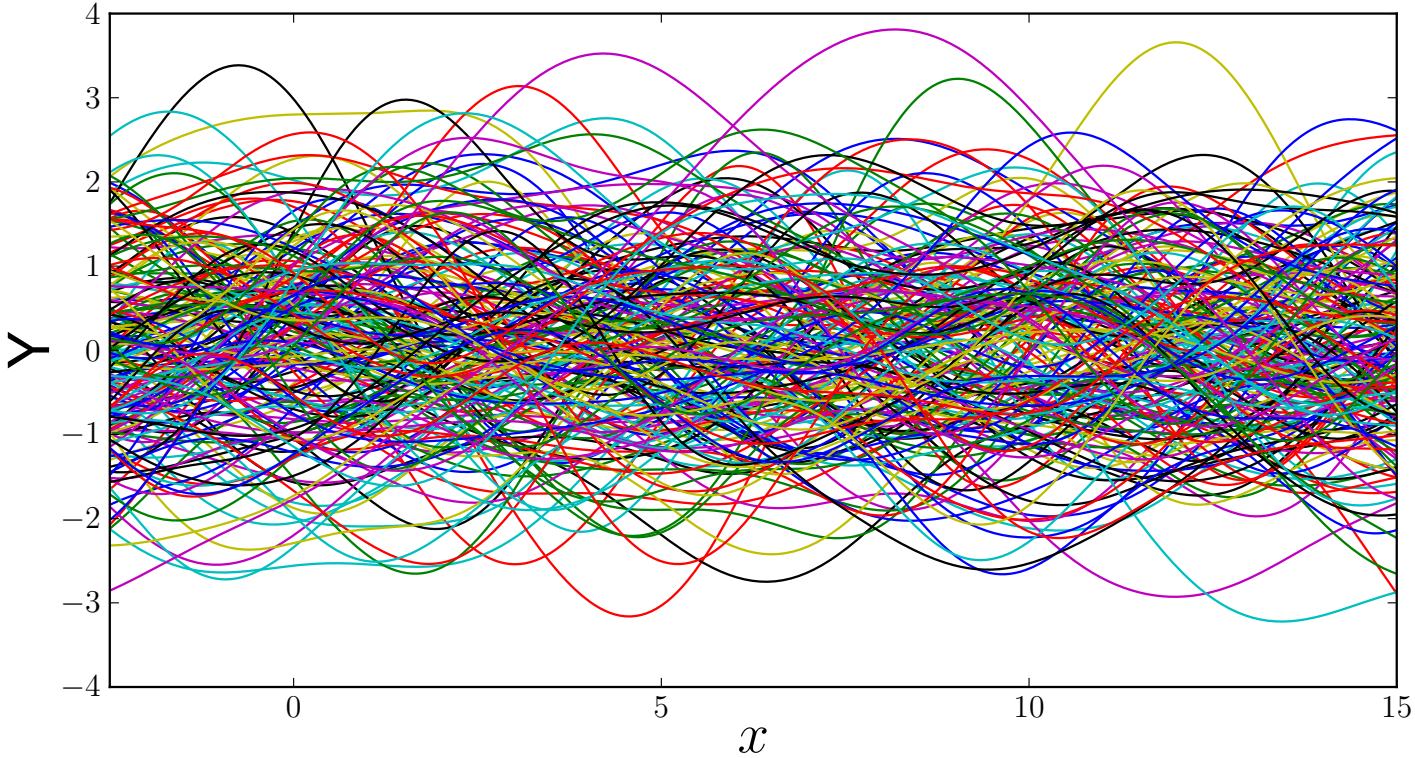
3. Gaussian process regression

We assume we have observed a function f for a set of points $X = (X_1, \ldots, X_n)$:



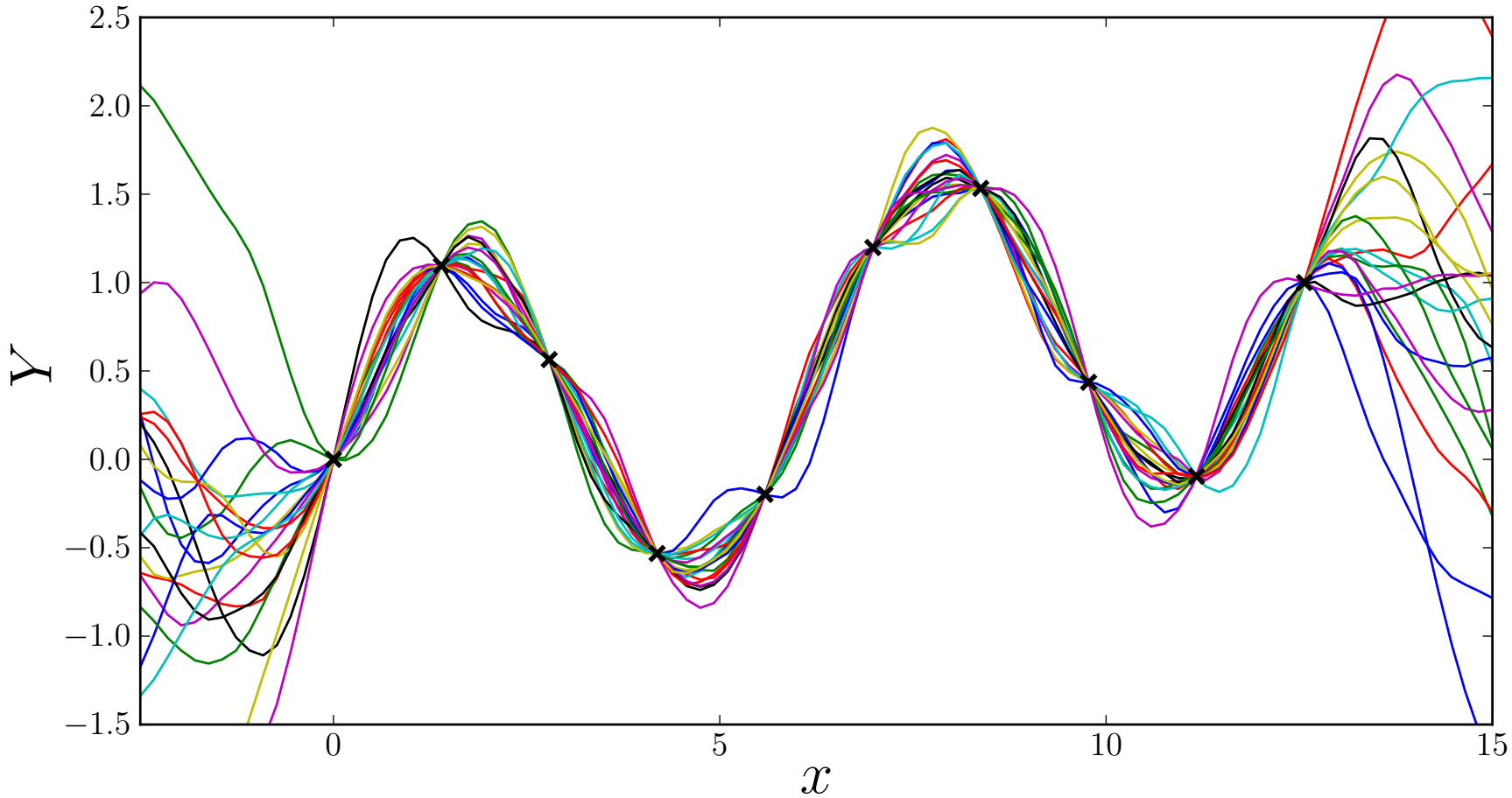
The vector of observations is $F = f(X)$ (ie $F_i = f(X_i)$).

Since f in unknown, we make the general assumption that it is the sample path of a Gaussian process $Z \sim \mathcal{N}(0, k)$:

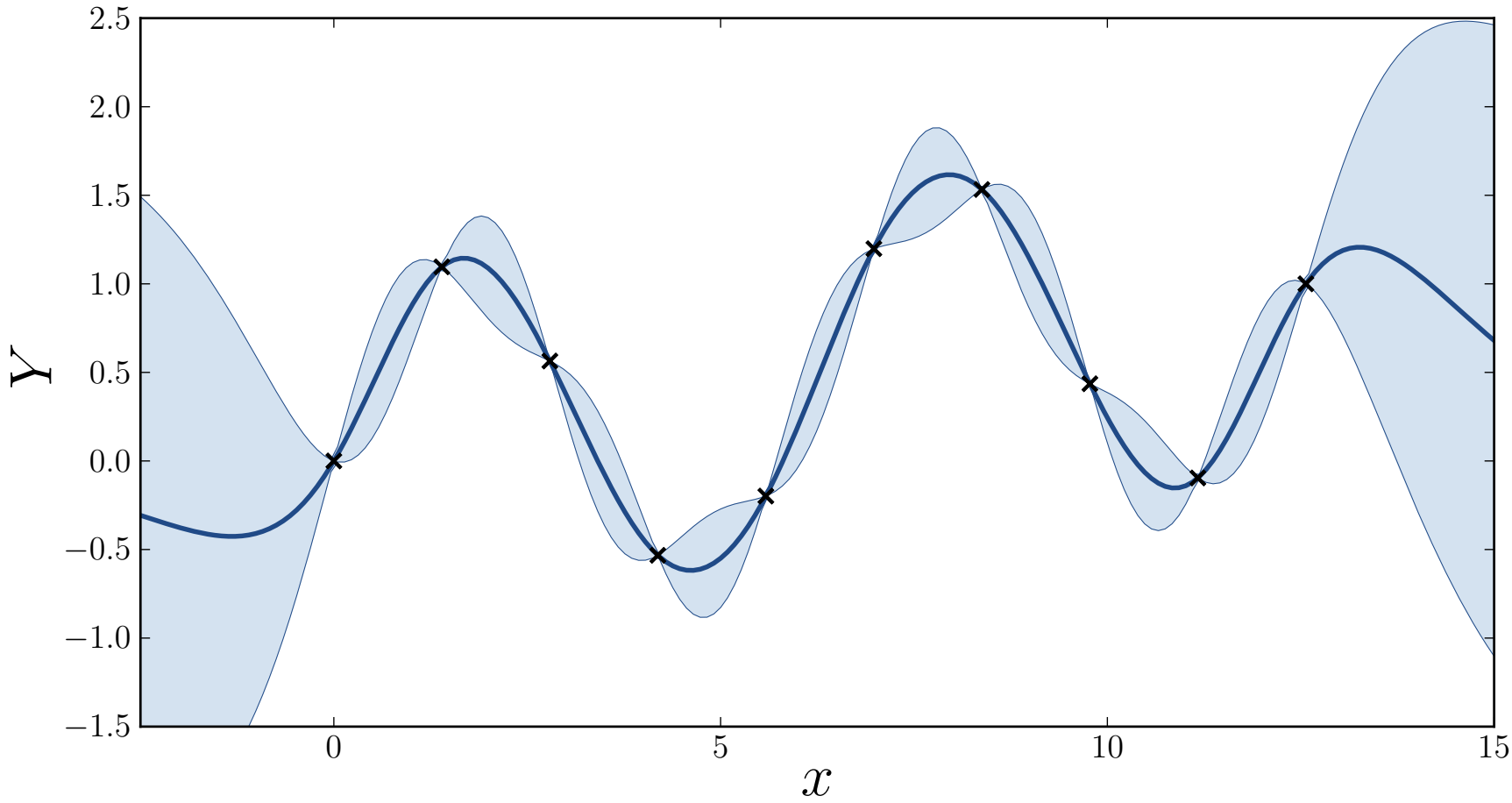


What would be the next step?

If we remove all the samples that do not interpolate we obtain:



It can summarized by a mean function and 95% confidence intervals.



In practice, the conditional distribution can be obtained analyticaly:

By definition, $(Z(x), Z(X))$ is multivariate normal so we know the distribution of $Z(x)|Z(X) = F$ is $\mathcal{N}(m(.), c(.,.))$ with:

$$\begin{aligned} m(x) &= \mathrm{E}[Z(x)|Z(X)=F] \\ &= k(x,X)k(X,X)^{-1}F \\ c(x,y) &= \mathrm{cov}[Z(x), Z(y)|Z(X)=F] \\ &= k(x,y) - k(x,X)k(X,X)^{-1}k(X,y) \end{aligned}$$

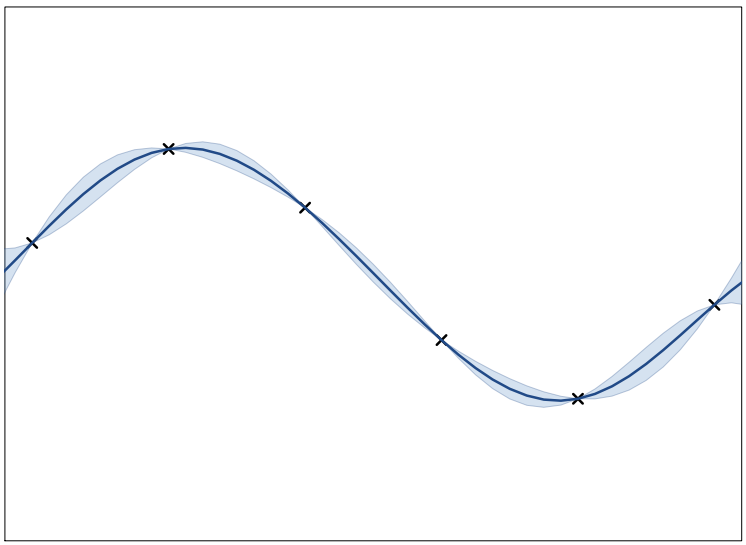
A few remarkable properties of GPR models

- They (can) interpolate the data-points
- The prediction variance does not depend on the observations
- The mean predictor does not depend on the variance parameter
- They (usually) come back to zero when we are far away from the observations.

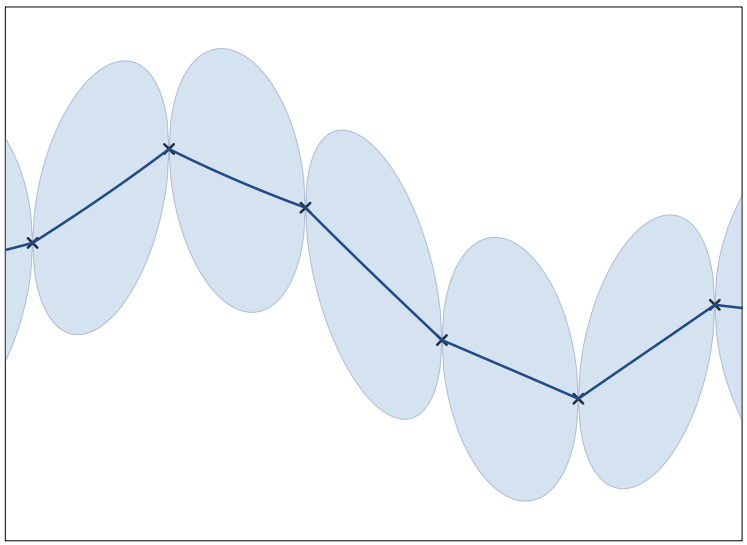
Can we prove them?

Changing the kernel has a huge impact on the model:

Gaussian kernel:

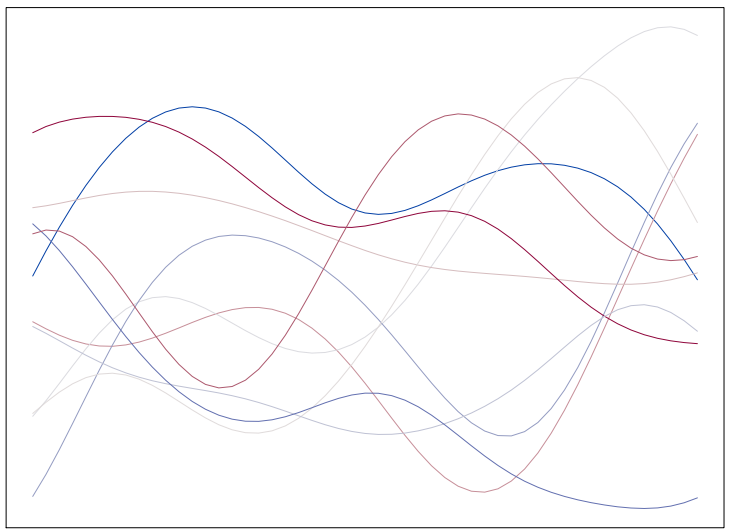


Exponential kernel:

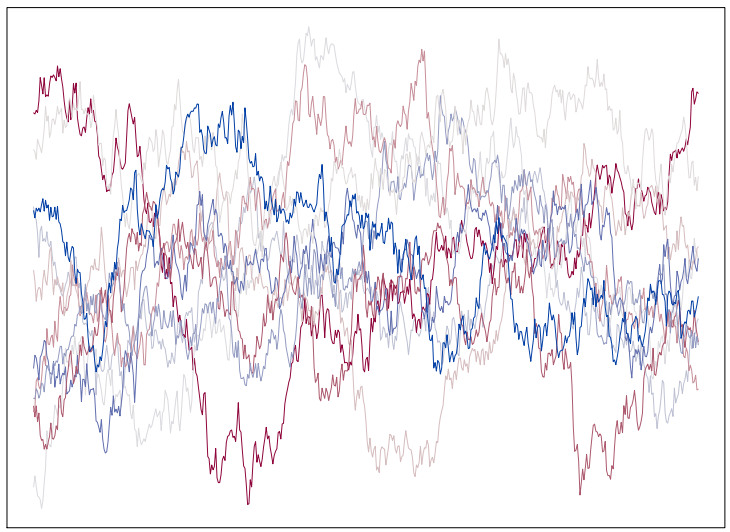


This is because changing the kernel means changing the prior on f

Gaussian kernel:

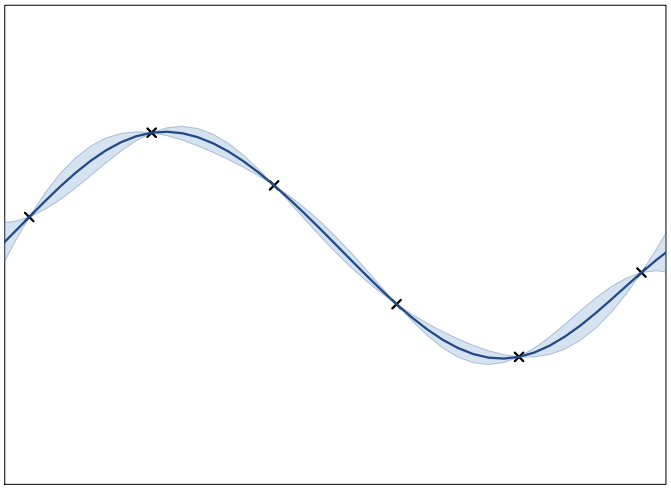


Exponential kernel:

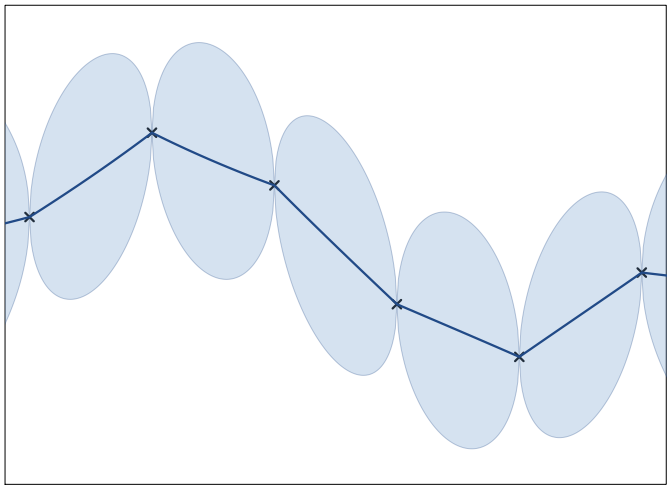


There is no model/kernel that is intrinsically better... it depends on the data!

Gaussian kernel:



Exponential kernel:



The kernel has to be chosen accordingly to our prior belief on the behaviour of the function to study:

- is it continuous, differentiable, how many times?
- is it stationary ?
- ...

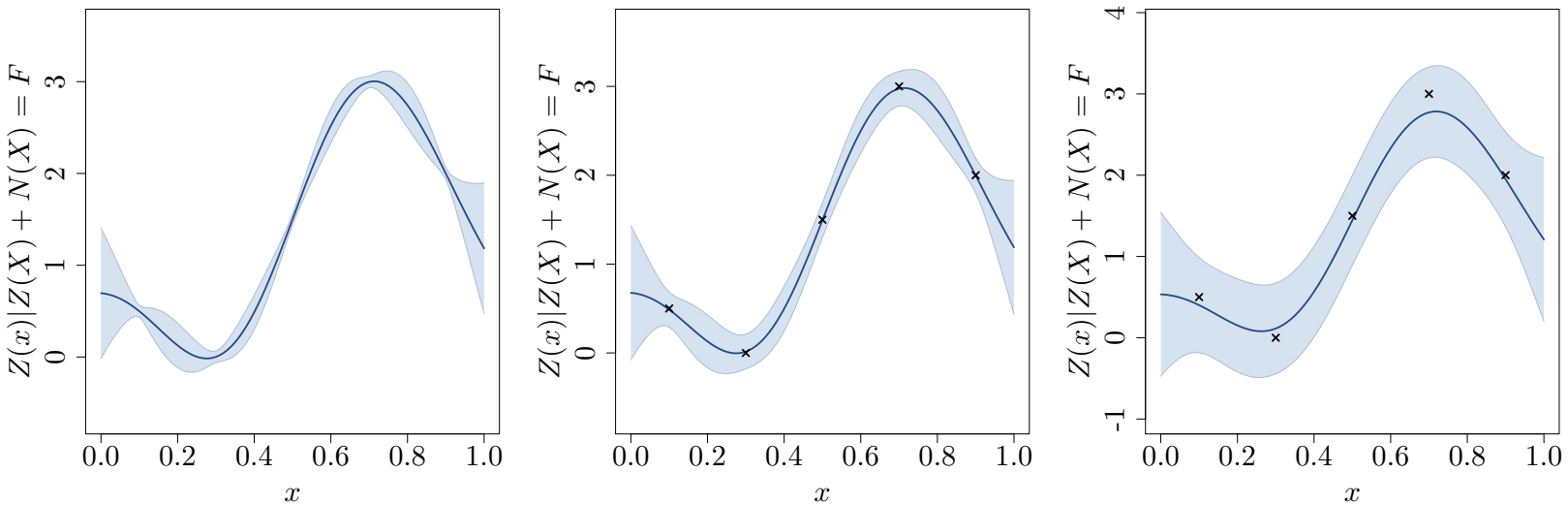
We are not always interested in models that interpolate the data. For example, if there is some observation noise: $F = f(X) + \varepsilon$. Let

N be a process $\mathcal{N}(0, n(.,.))$ that represent the observation noise. The expressions of GPR with noise are

$$\begin{aligned} m(x) &= \mathrm{E}[Z(x)|Z(X) + N(X)=F] \\ &= k(x,X)(k(X,X) + n(X,X))^{-1}F \end{aligned}$$

$$\begin{aligned} c(x,y) &= \mathrm{cov}[Z(x), Z(y)|Z(X) + N(X)=F] \\ &= k(x,y) - k(x,X)(k(X,X) + n(X,X))^{-1}k(X,y) \end{aligned}$$

Examples of models with observation noise for $n(x, y) = \tau^2 \delta_{x,y}$:



The values of τ^2 are respectively 0.001, 0.01 and 0.1.

$\Rightarrow$ R demo

Parameter estimation

We have seen previously that the choice of the kernel and its parameters have a great influence on the model.

In order to choose a prior that is suited to the data at hand, we can consider:

- minimising the model error
- Using maximum likelihood estimation

We will now detail the second one.

Definition

The **likelihood** of a distribution with a density f_X given some observations $X_1, \ldots, X_p$ is:

$$L = \prod_{i=1}^{p} f_X(X_i)$$

This quantity can be used to measure the adequacy between observations and a distribution.

In the GPR context, we often have only **one observation** of the vector F. The likelihood is then:

$$L = f_{Z(X)}(F) = \frac{1}{(2\pi)^{n/2}|k(X,X)|^{1/2}} \exp\left(-\frac{1}{2}F^t k(X,X)^{-1}F\right).$$

It is thus possible to maximise L – or $\log(L)$ – with respect to the kernel's parameters in order to find a well suited prior.

$\Rightarrow$ R demo

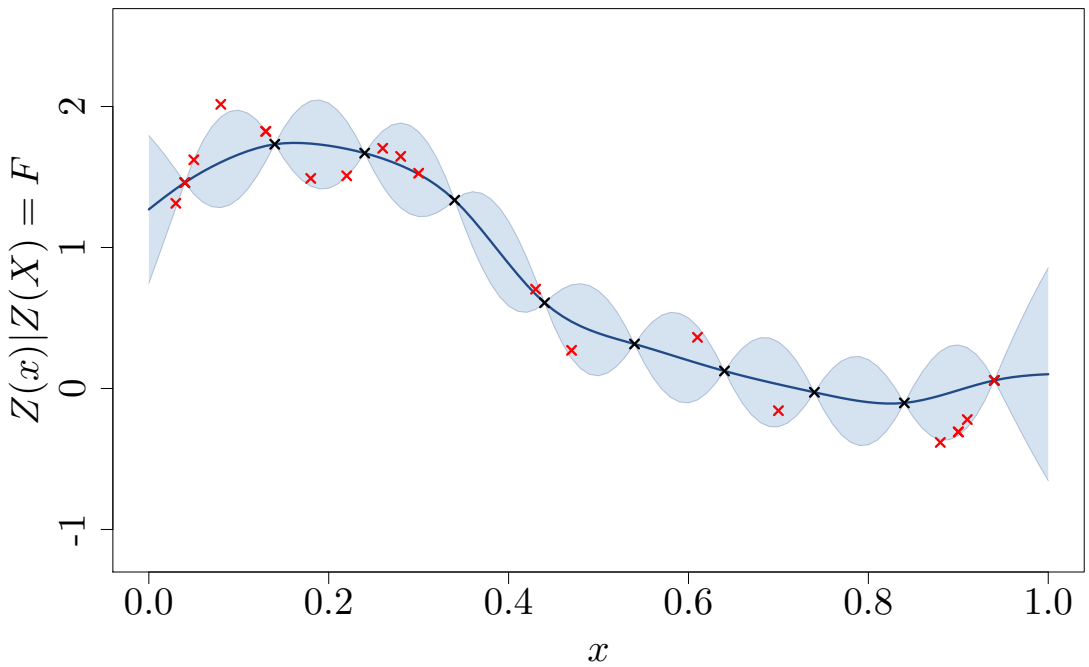
Model validation

We have seen that given some observations $F = f(X)$, it is very easy to build lots of models, either by changing the kernel parameters or the kernel itself.

The interesting question now is to know how to get a good model. To do so, we will need to answer the following questions:

- What is a good model?
- How to measure it?

The idea is to introduce new data and to compare the model prediction with reality



Since GPR models provide a mean and a covariance structure for the error they both have to be assessed.

Let X_t be the test set and $F_t = f(X_t)$ be the associated observations.

The accuracy of the mean can be measured by computing:

$$\text{Mean Square Error} \qquad MSE = \text{mean}((F_t - m(X_t))^2)$$

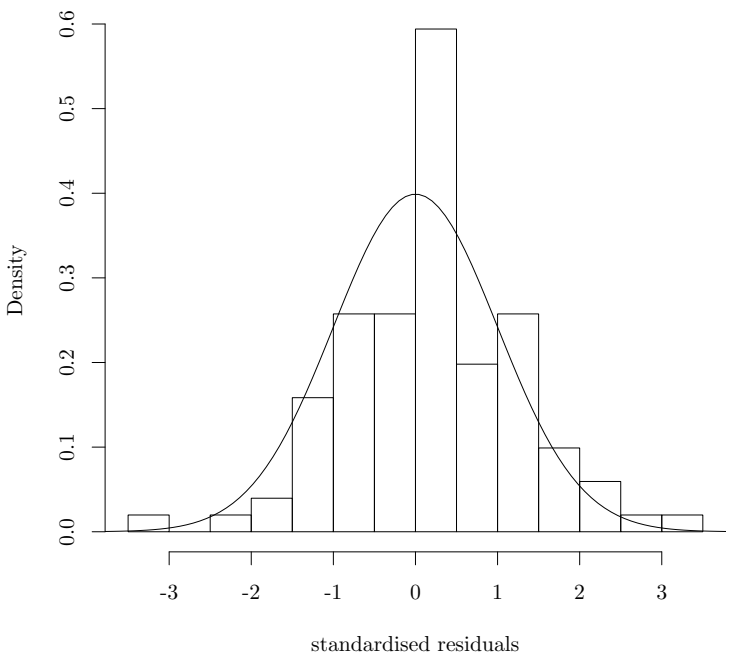
$$\text{A "normalised" criterion} \qquad Q_2 = 1 - \frac{\sum (F_t - m(X_t))^2}{\sum (F_t - \text{mean}(F_t))^2}$$

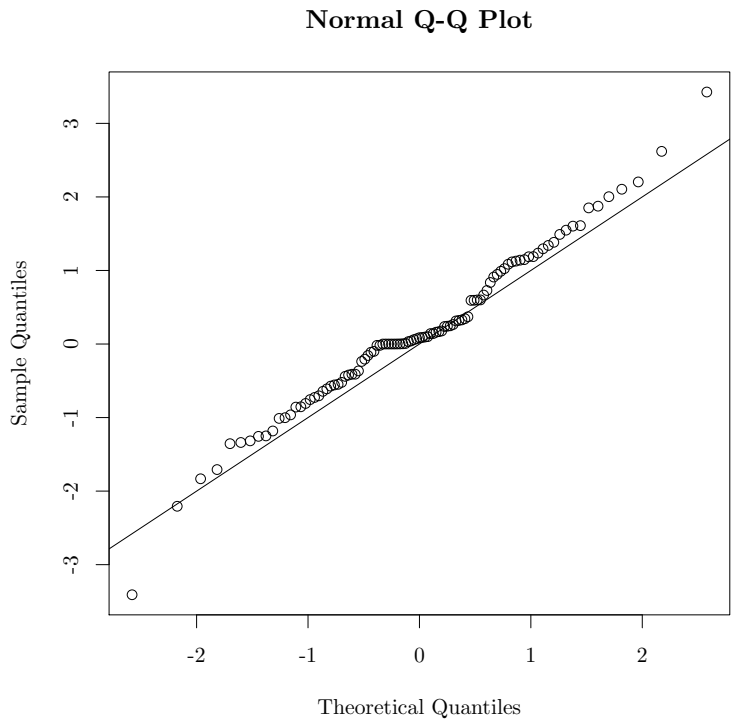
On the above example we get $MSE = 0.038$ and $Q_2 = 0.95$.

The predicted distribution can be tested by normalising the residuals.

According to the model, $F_t \sim \mathcal{N}(m(X_t), c(X_t, X_t))$.

$c(X_t, X_t)^{-1/2}(F_t - m(X_t))$ should thus be independents $\mathcal{N}(0, 1)$:





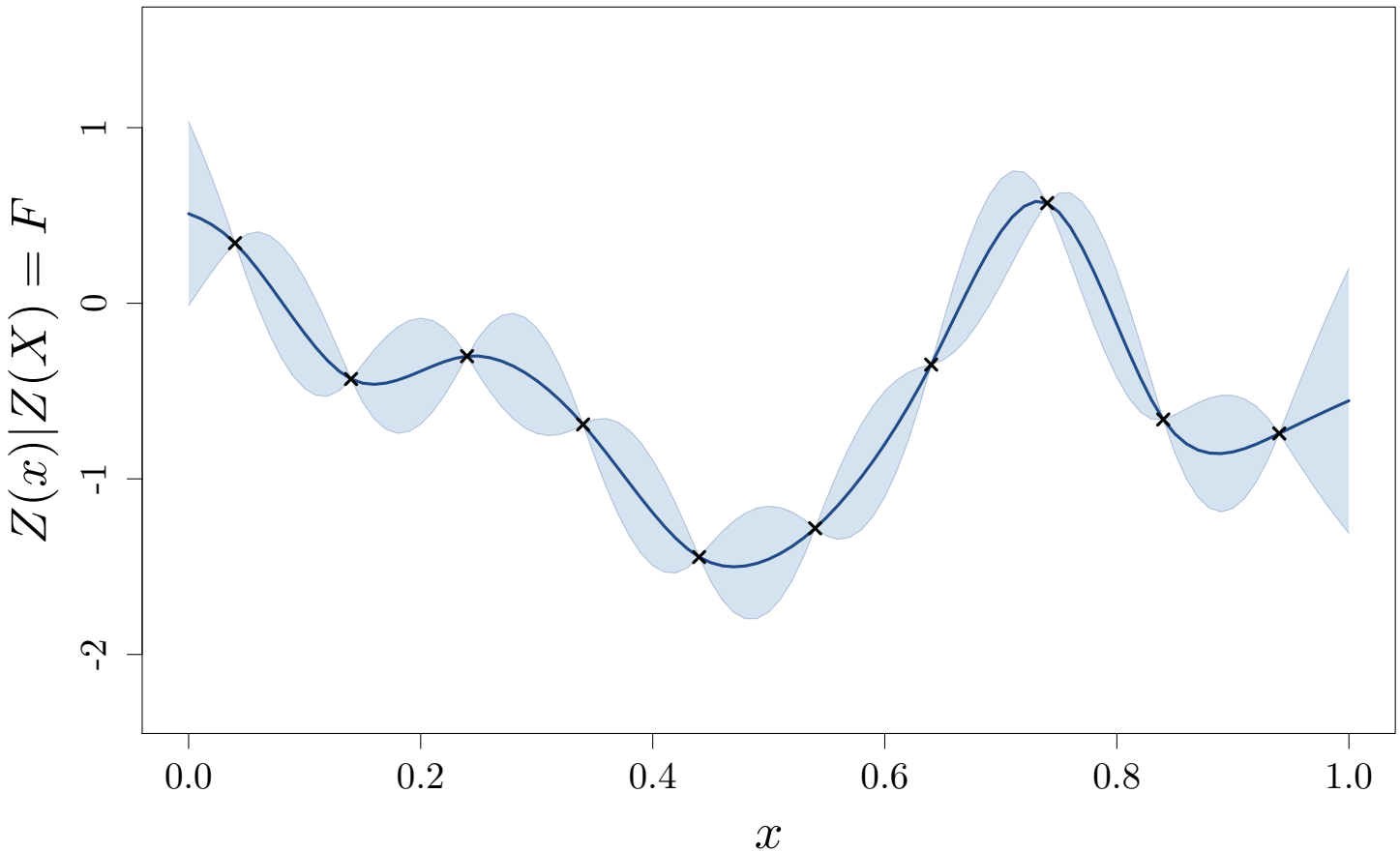
When no test set is available, another option is to consider cross validation methods such as leave-one-out.

The steps are:

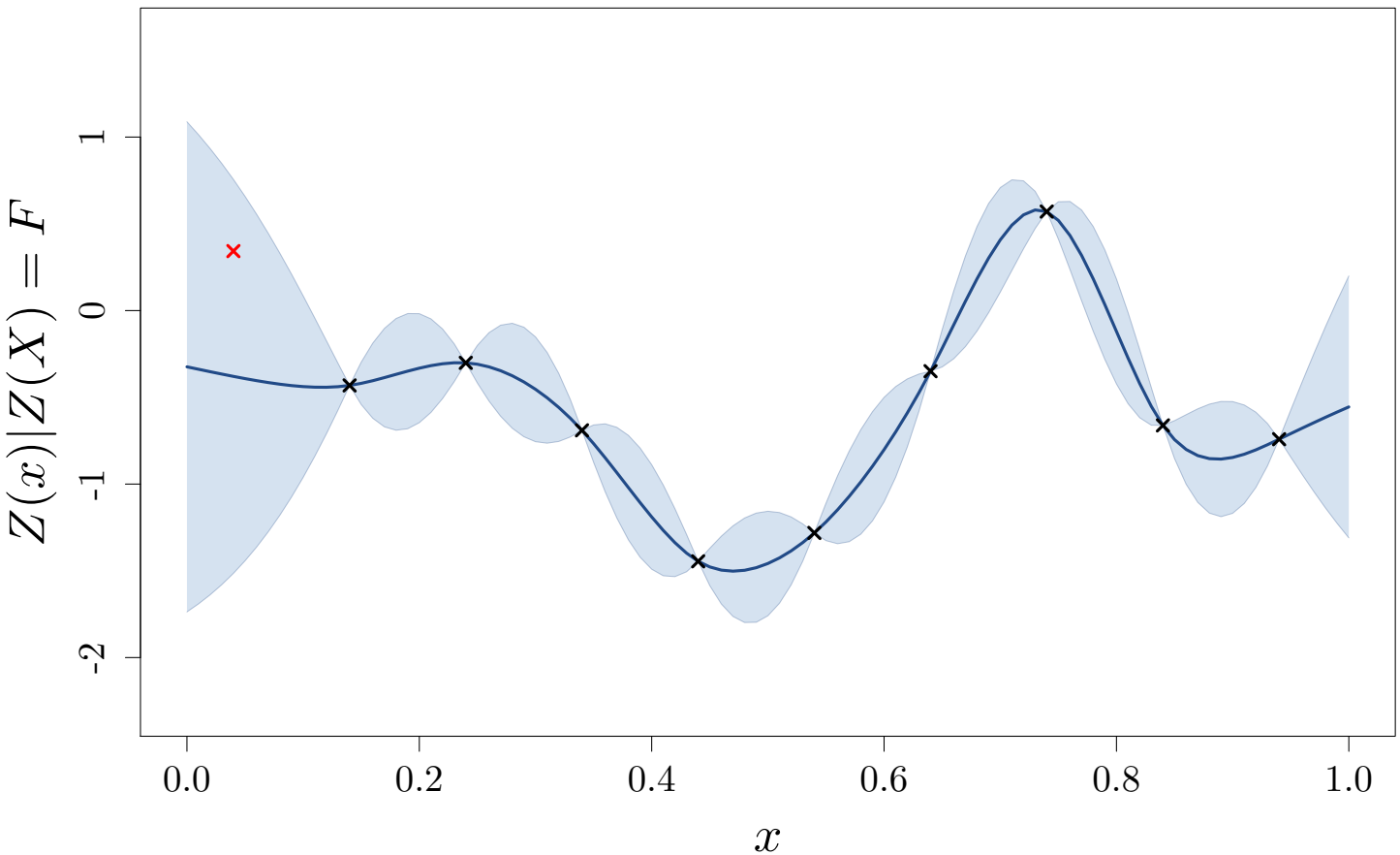
1. build a model based on all observations except one
2. compute the model error at this point

This procedure can be repeated for all the design points in order to get a vector of error.

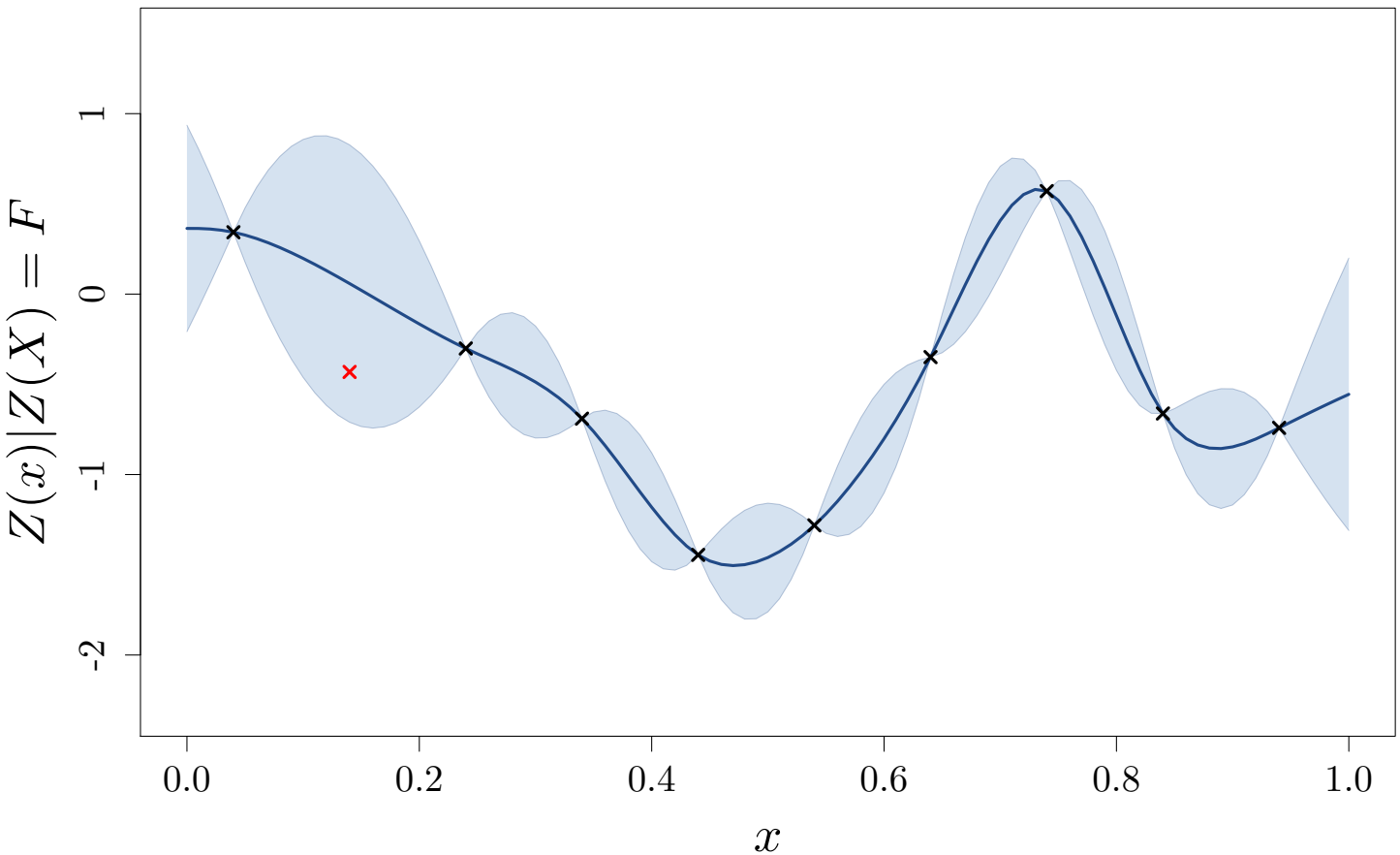
Model to be tested:



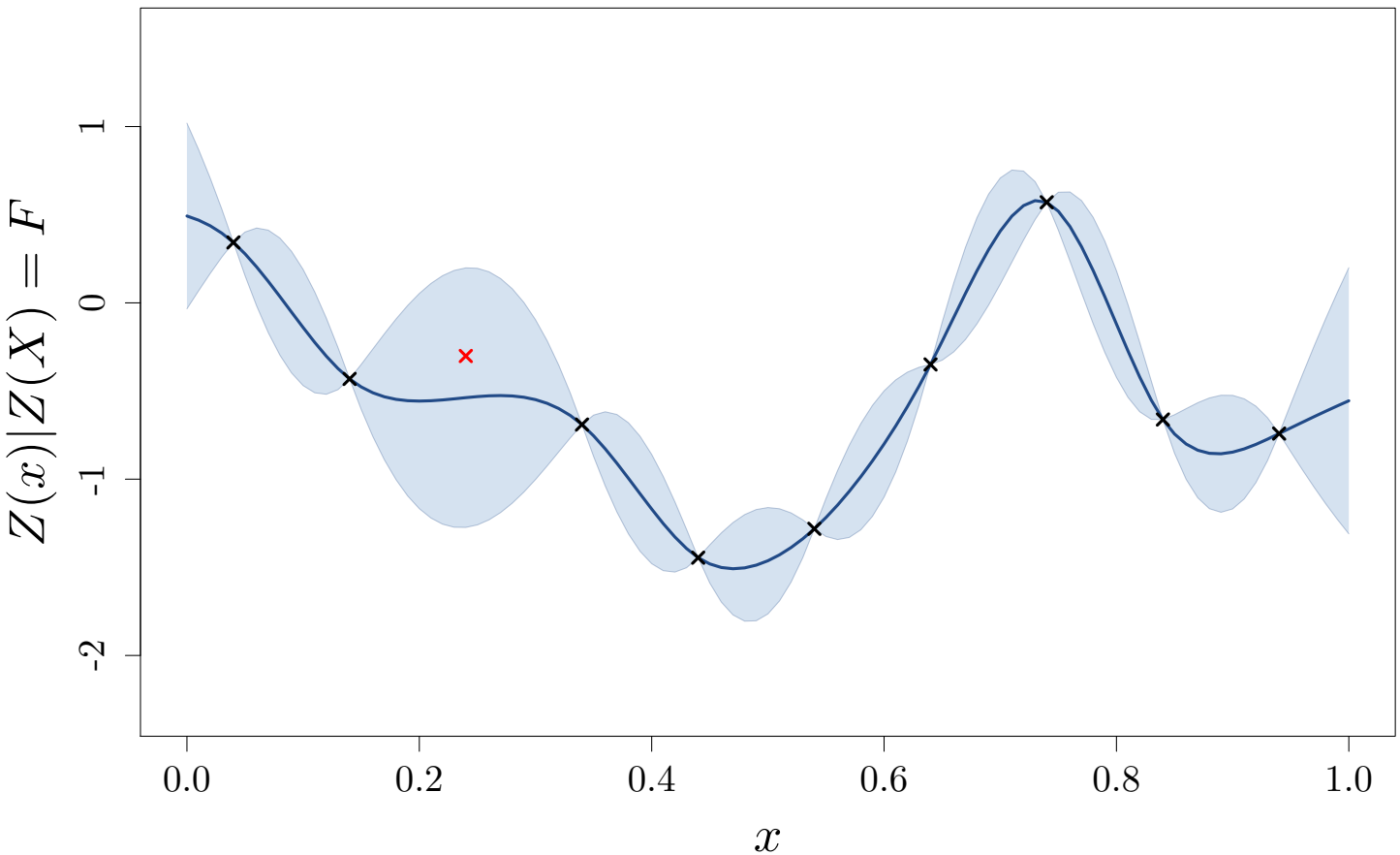
Step 1:



Step 2:



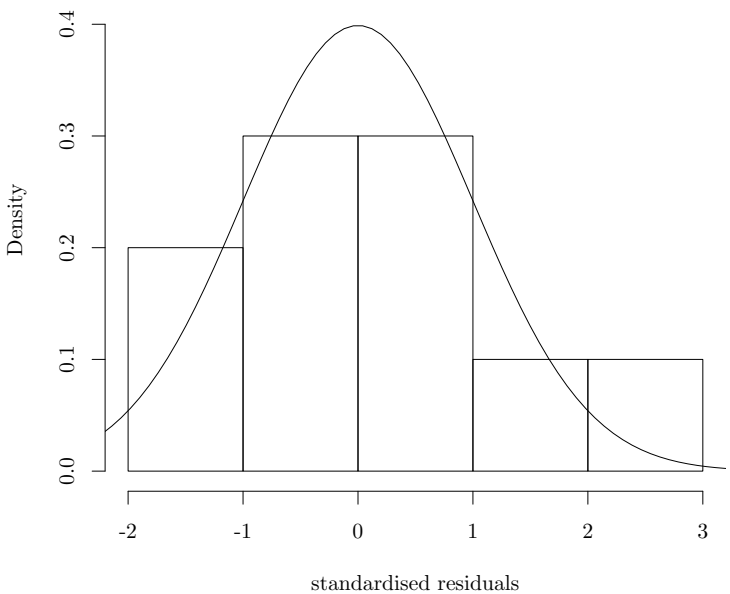
Step 3:

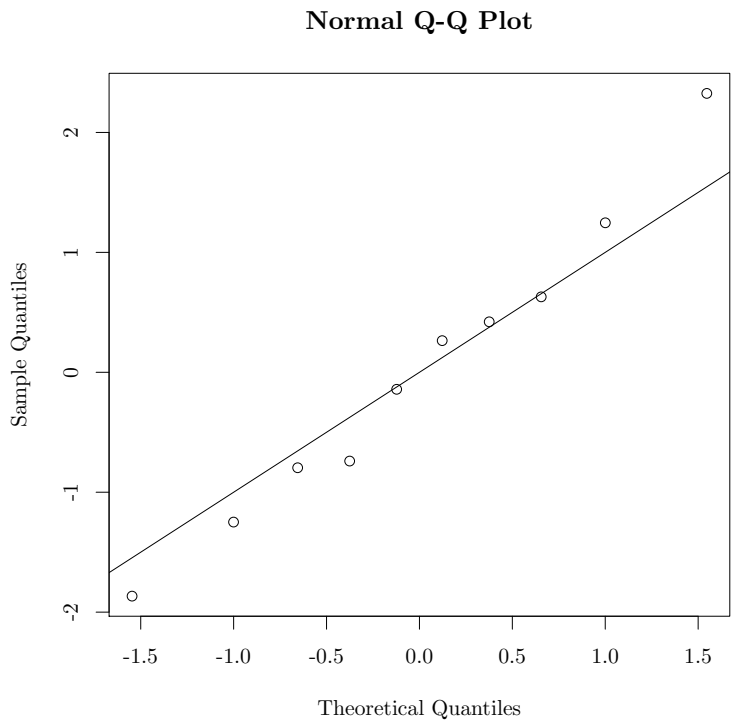


We finally obtain:

$$MSE = 0.24 \text{ and } Q_2 = 0.34.$$

We can also look at the residual distribution. For leave-one-out, there is no joint distribution for the residuals so they have to be standardised independently.





Kernel Design

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

Kernels can be:

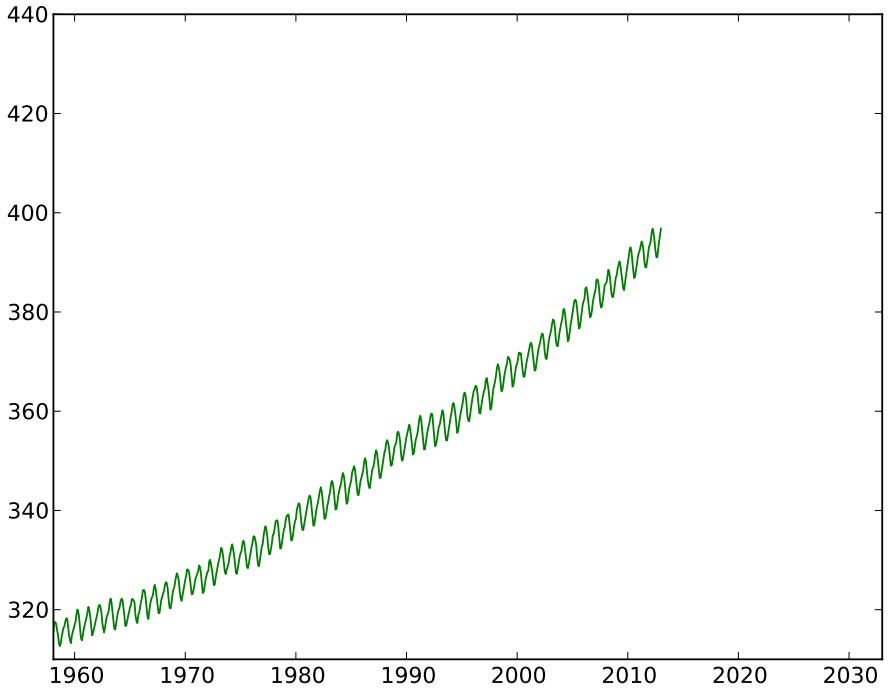
- Summed together
  - On the same space $k(x, y) = k_1(x, y) + k_2(x, y)$
  - On the tensor space $k(x, y) = k_1(x_1, y_1) + k_2(x_2, y_2)$
- Multiplied together
  - On the same space $k(x, y) = k_1(x, y) \times k_2(x, y)$
  - On the tensor space $k(x, y) = k_1(x_1, y_1) \times k_2(x_2, y_2)$
- Composed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

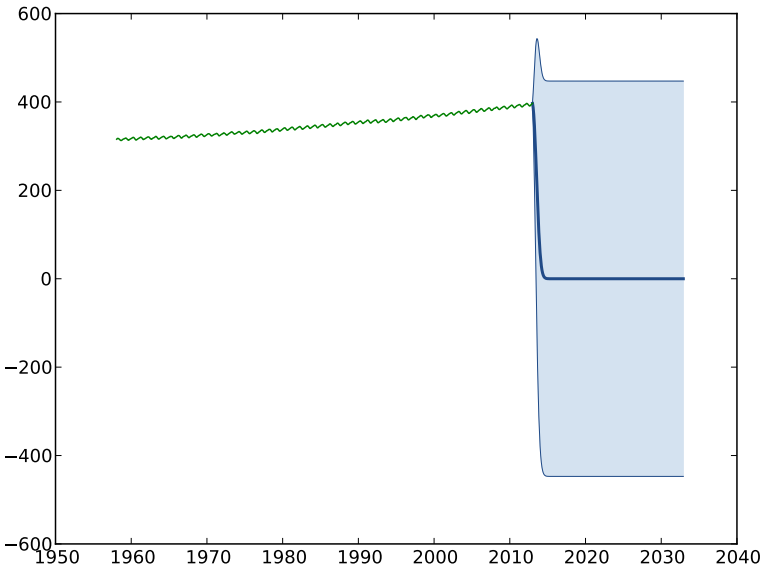


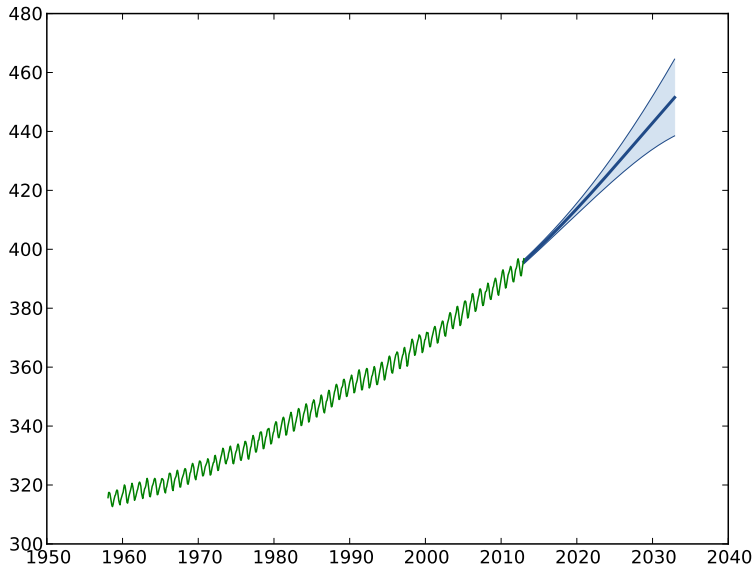
Let's try to predict the concentration for the next 20 years.

Sum of kernels over the same space

We first consider a squared-exponential kernel:

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





The results are terrible!

Sum of kernels over the same space

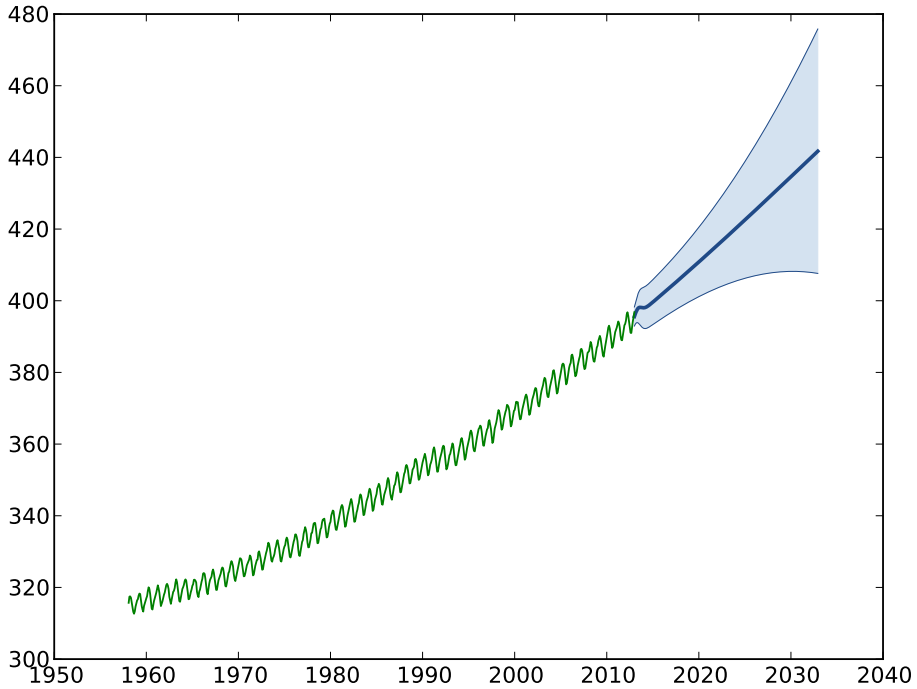
What happen if we sum both kernels?

$$k(x, y) = k_{rbf1}(x, y) + k_{rbf2}(x, y)$$

Sum of kernels over the same space

What happen if we sum both kernels?

$$k(x, y) = k_{rbf1}(x, y) + k_{rbf2}(x, y)$$



The model is drastically improved!

Sum of kernels over the same space

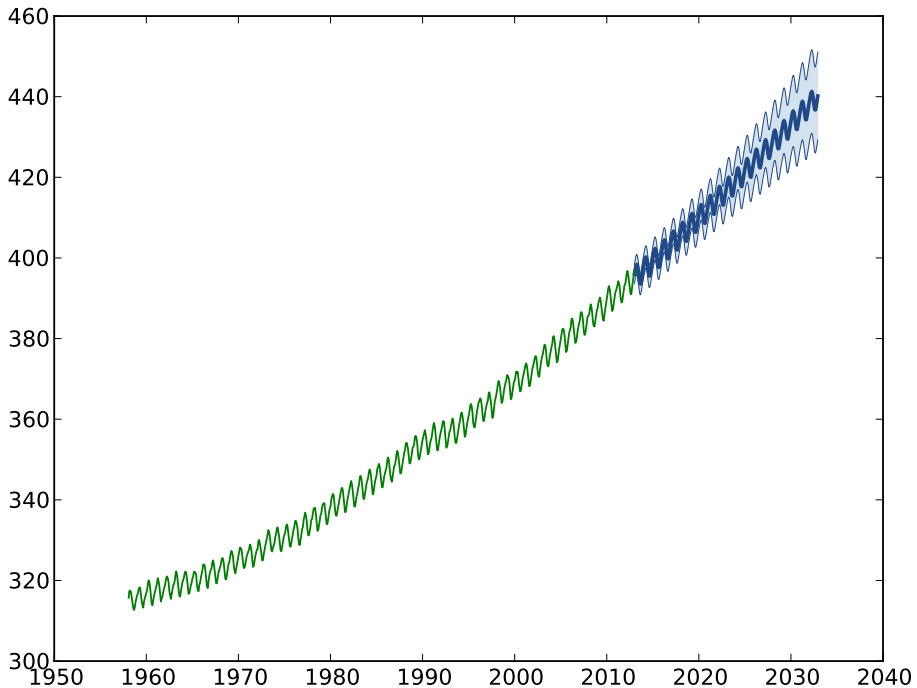
We can try the following kernel:

$$k(x, y) = \sigma_0^2 x^2 y^2 + k_{rbf1}(x, y) + k_{rbf2}(x, y) + k_{per}(x, y)$$

Sum of kernels over the same space

We can try the following kernel:

$$k(x, y) = \sigma_0^2 x^2 y^2 + k_{rbf1}(x, y) + k_{rbf2}(x, y) + k_{per}(x, y)$$



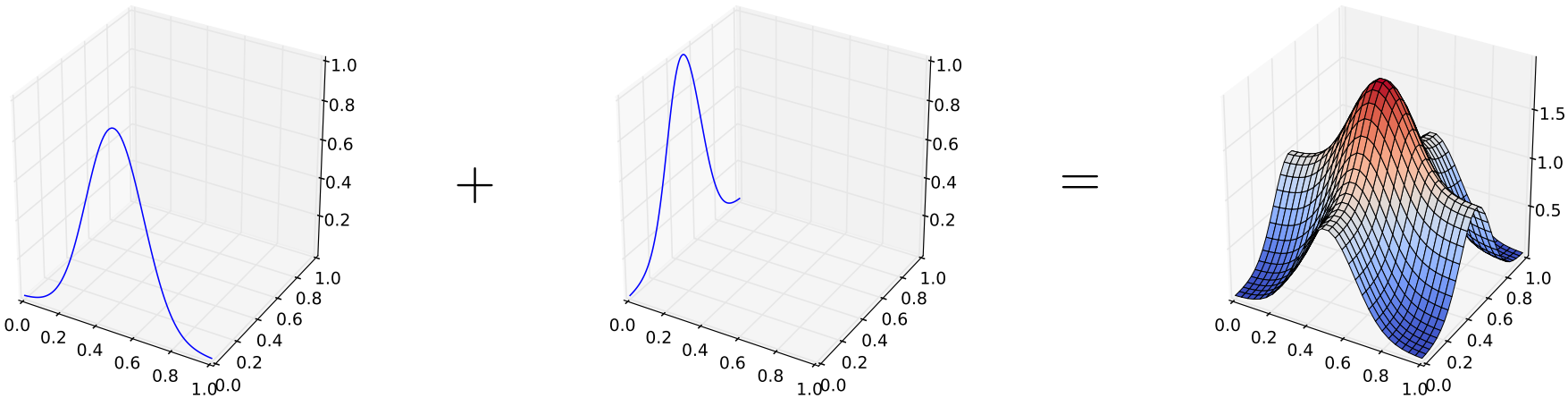
Once again, the model is significantly improved.

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.

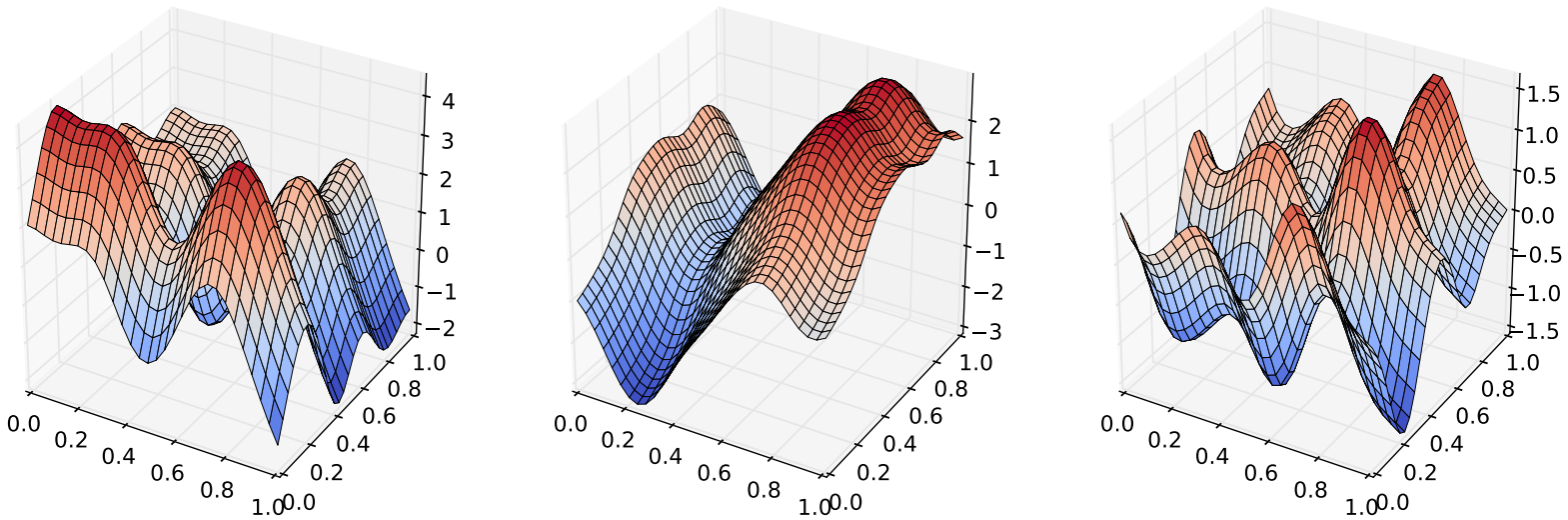


Remark: From a GP point of view, k is the kernel of

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

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

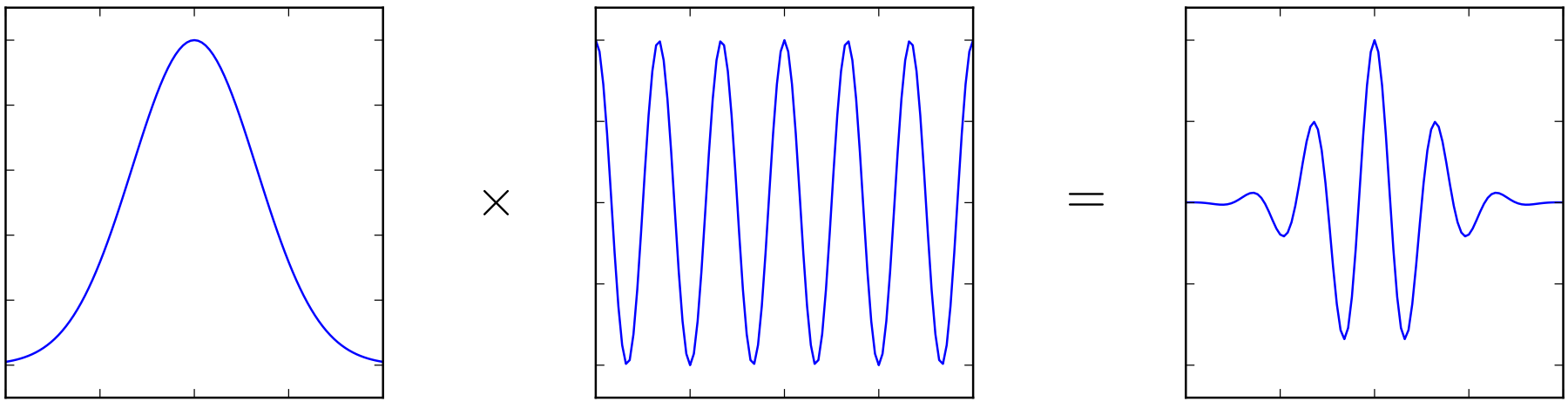
Property

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

is valid covariance structure.

Example

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



Product over the tensor space

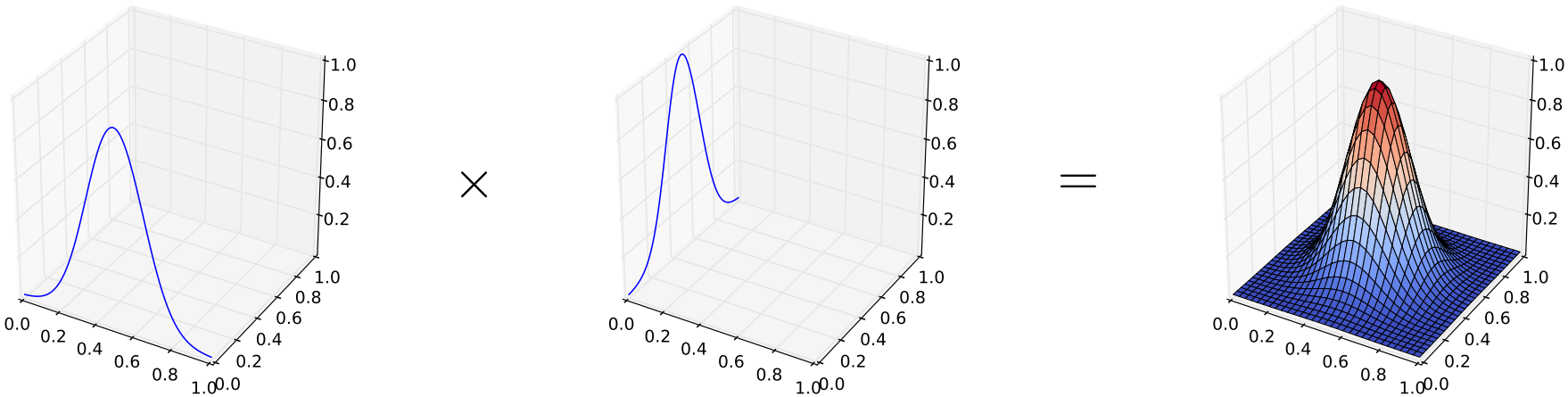
Property

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

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:

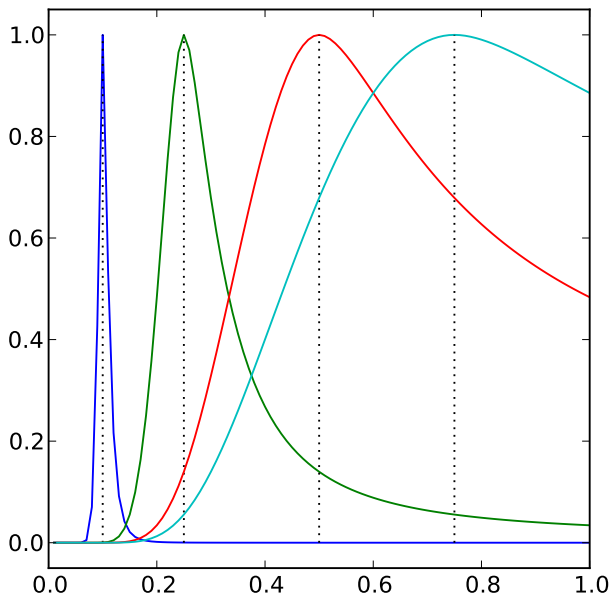
- k corresponds to the covariance of $Z(x) = Z_1(f(x))$
- This can be seen as a (non-linear) rescaling of the input space

Example

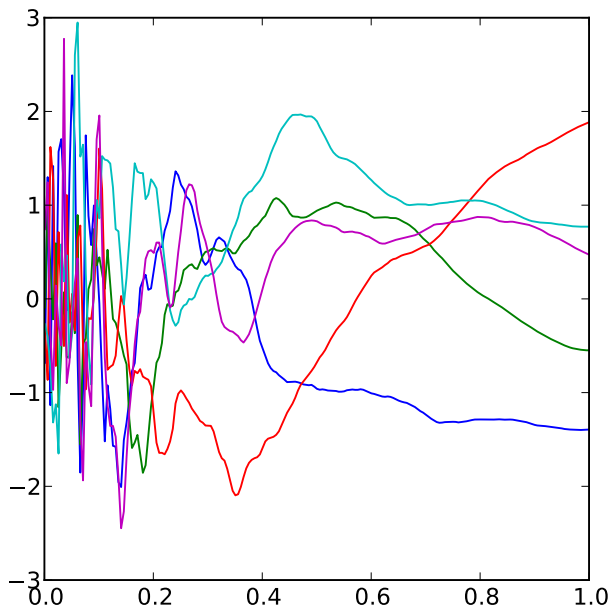
We consider $f(x) = \frac{1}{x}$ and a Matérn 3/2 kernel $k_1(x, y) = (1 + |x - y|)e^{-|x-y|}$.

We obtain:

Kernel



Sample paths



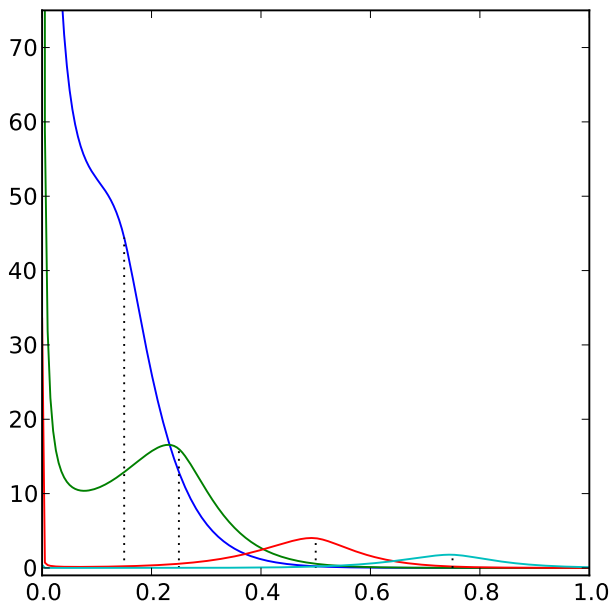
All these transformations can be combined!

Example

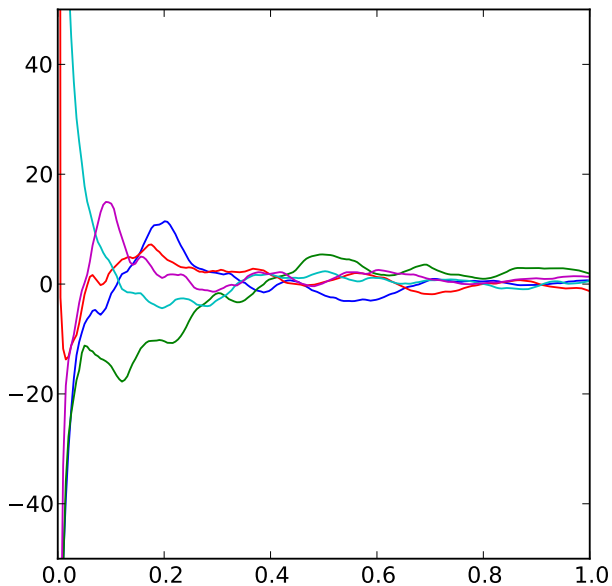
$k(x, y) = f(x)f(y)k_1(x, y)$ is a valid kernel.

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

Kernel



Sample paths



$\Rightarrow$ R demo

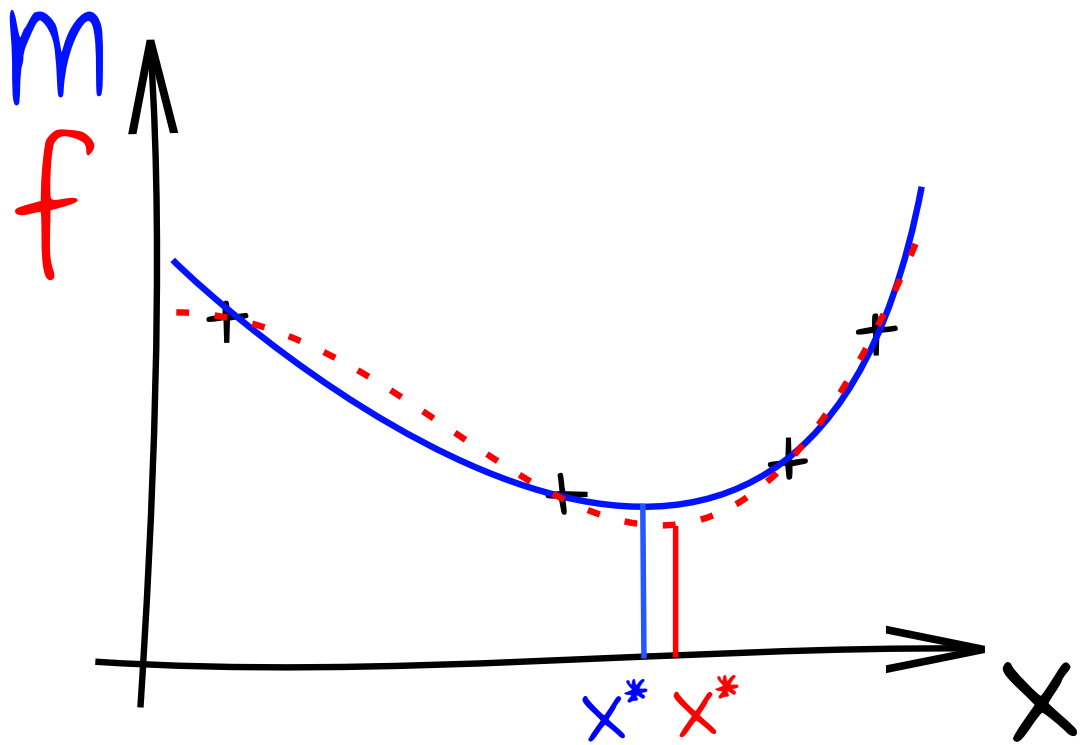
Other kernel design methods

There are two other popular methods for kernel design:

- Bochner Theorem
  There is an equivalence between positive measures and stationnary positive definite functions.
- Linear operators
  If the function to approximate has particular properties that can be obtained via a linear transform, it is possible to build a GP with the wanted properties. For example, one can build symmetric GPs or GPs with integral equal to zero.

Model based optimization methods

If the number of function evaluations are limited, we can run the optimization on the model instead of running it directly on the function

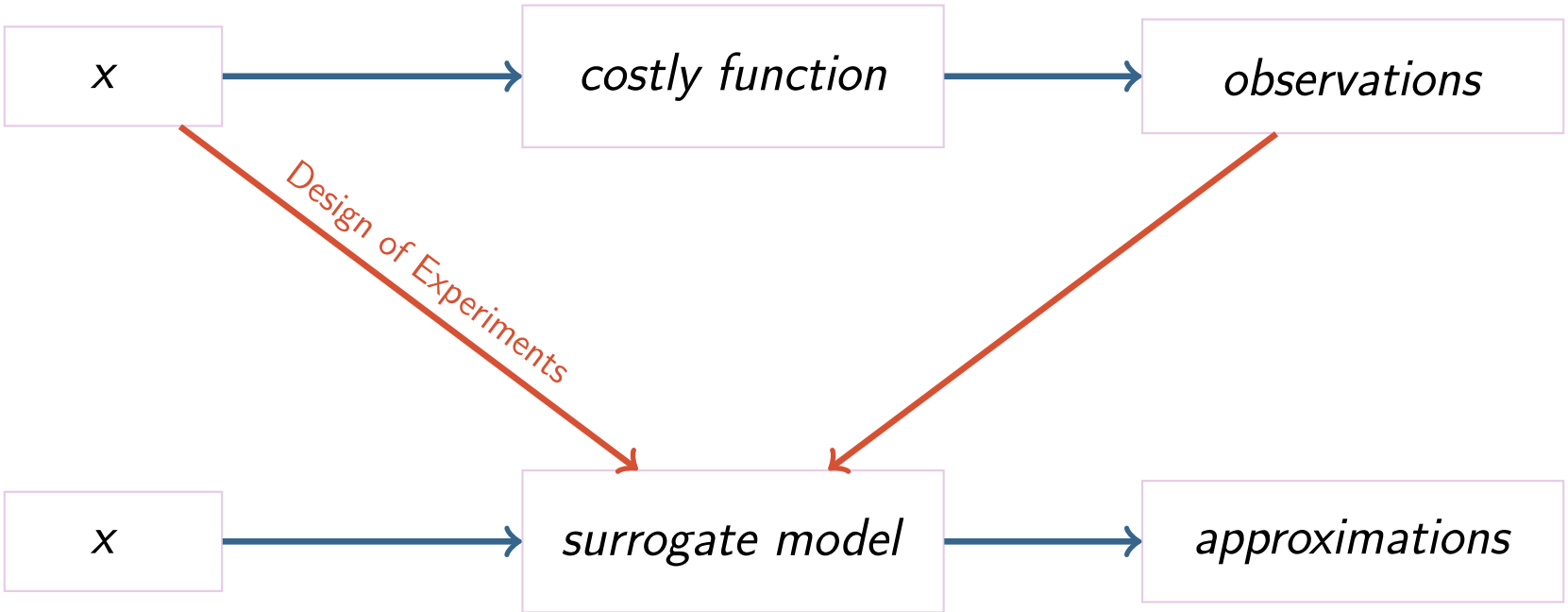


In the end, we hope that:

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

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

Overall framework



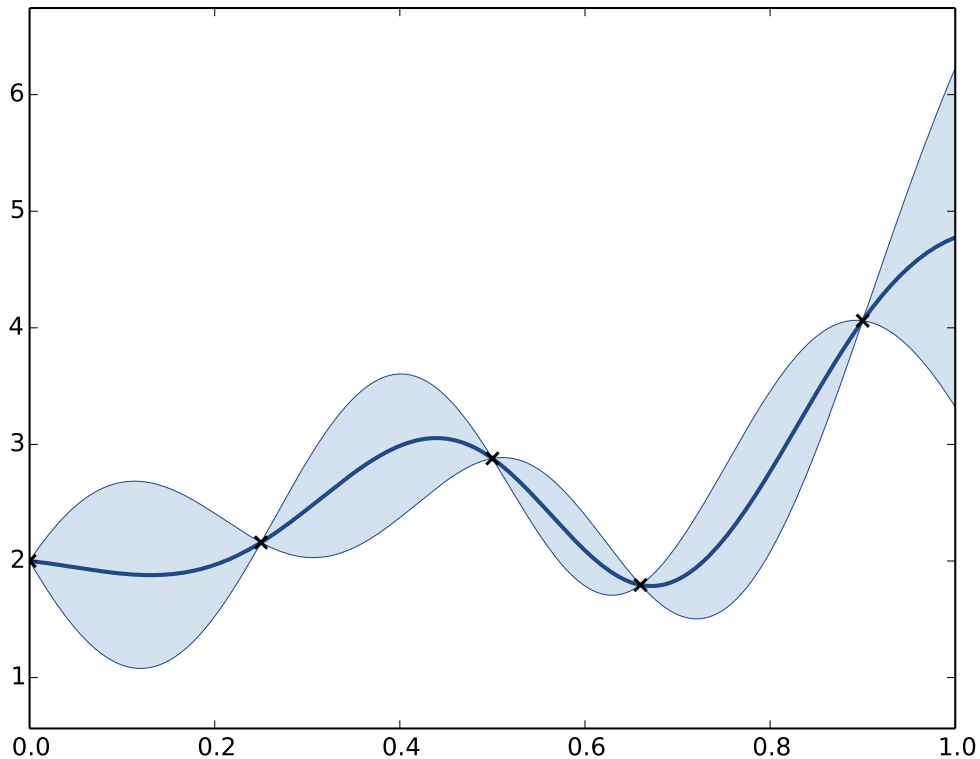
In practice, it is risky to take decisions based only on the model...

On the other hand, the model can be used to guide us in the search for the optimum.

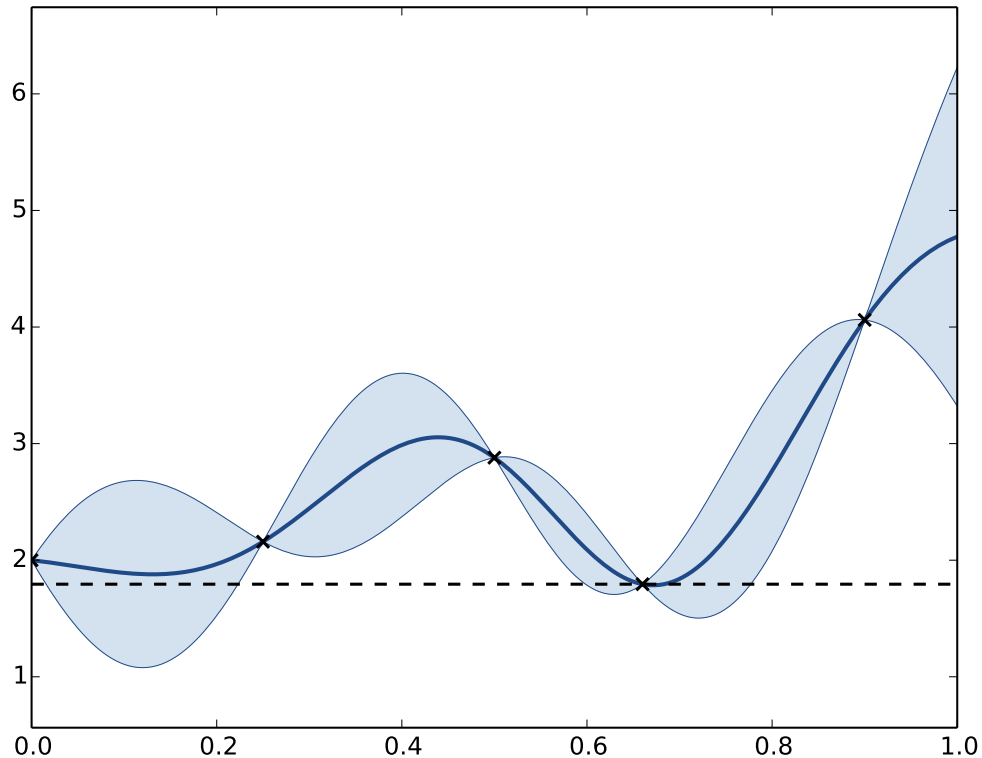
Global optimization methods are a trade-off between

- Exploitations of good results
- Exploration of the space

How can GPR models be helpful?



In our example, the best observed value is 1.79

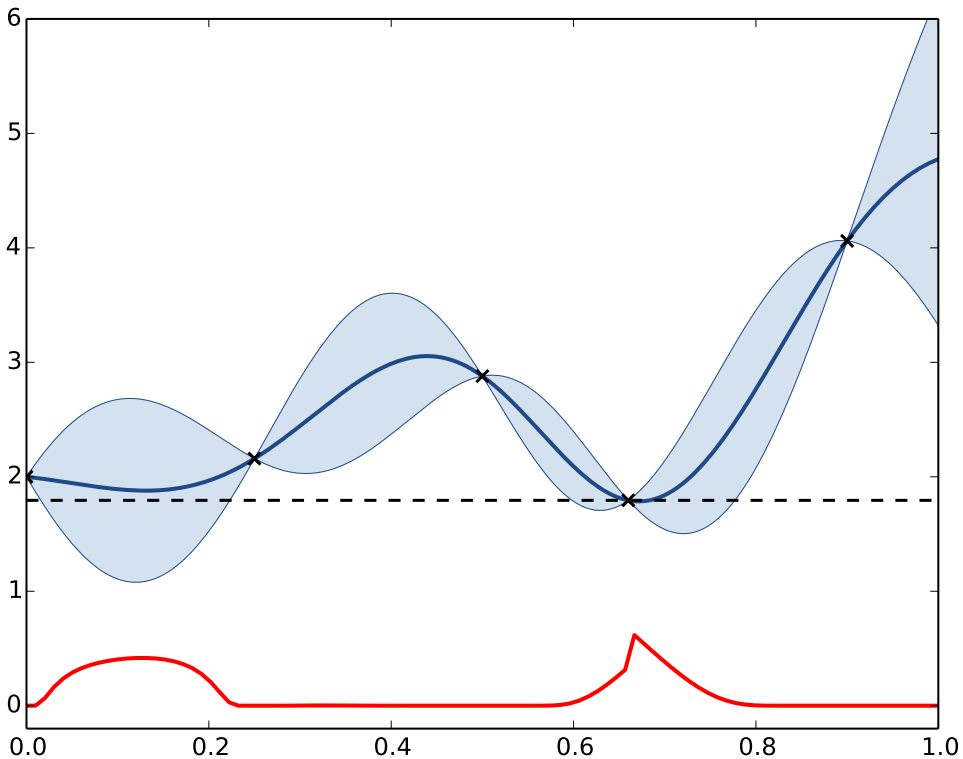


Various criteria can be studied

- probability of improvement
- Expected improvement

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 $PI(x) \geq 0.5$.

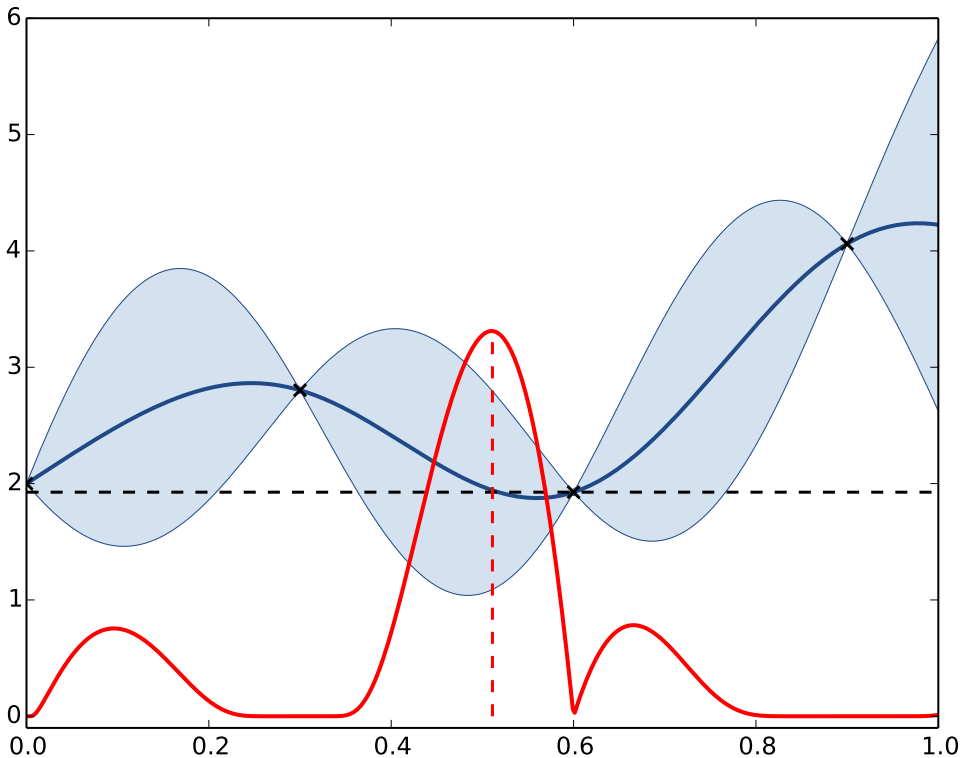
For such points, the improvement cannot be large...

Can we find another criterion?

Expected Improvement:

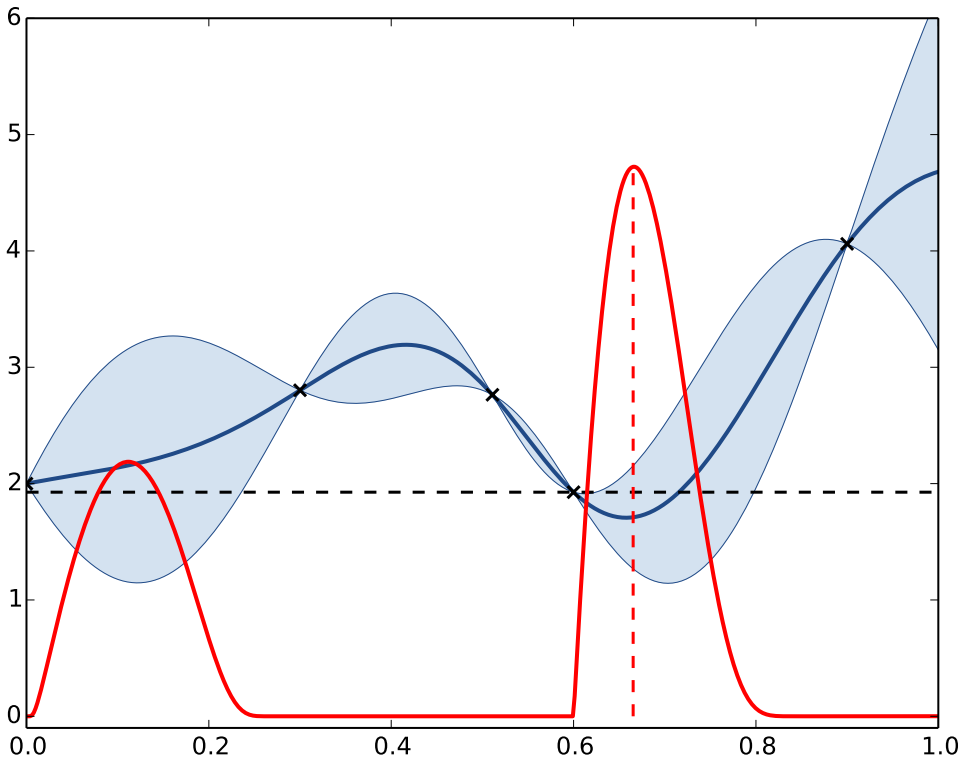
$$EI(x) = \sqrt{c(x,x)}(u(x)cdf(u(x)) + pdf(u(x)))$$

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



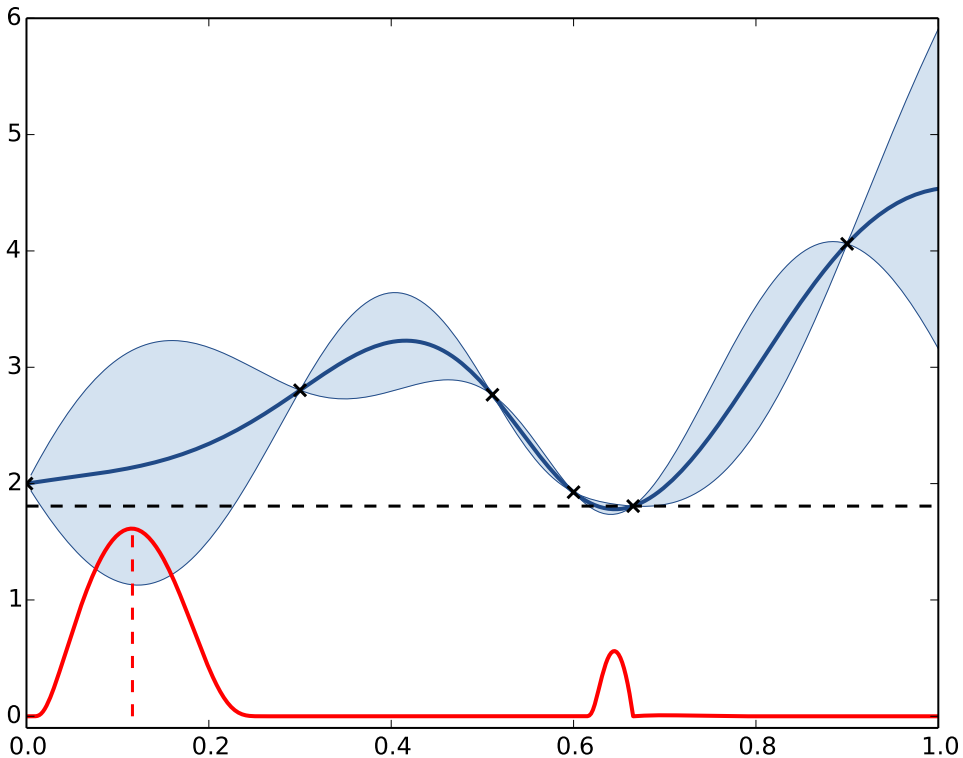
Expected Improvement

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



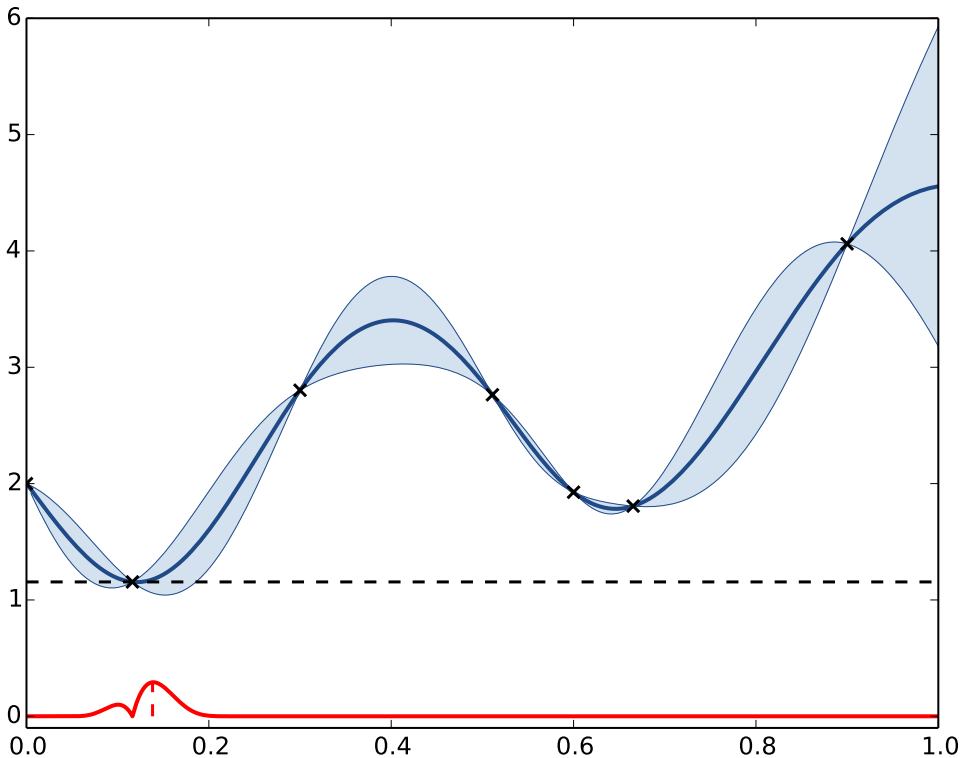
Expected Improvement

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



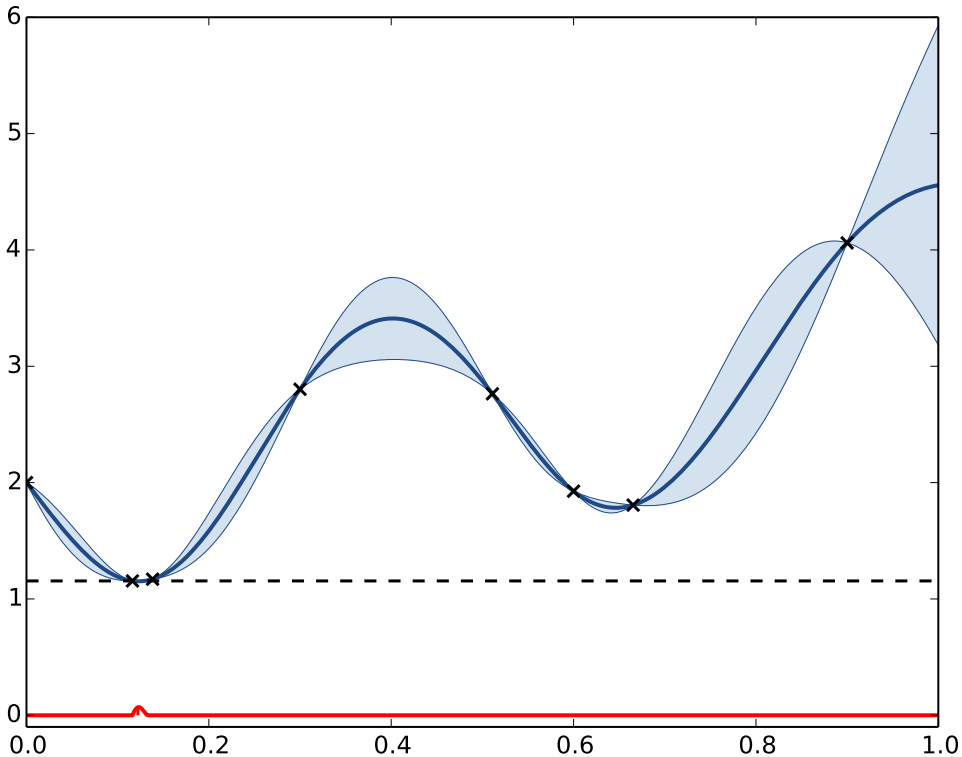
Expected Improvement

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

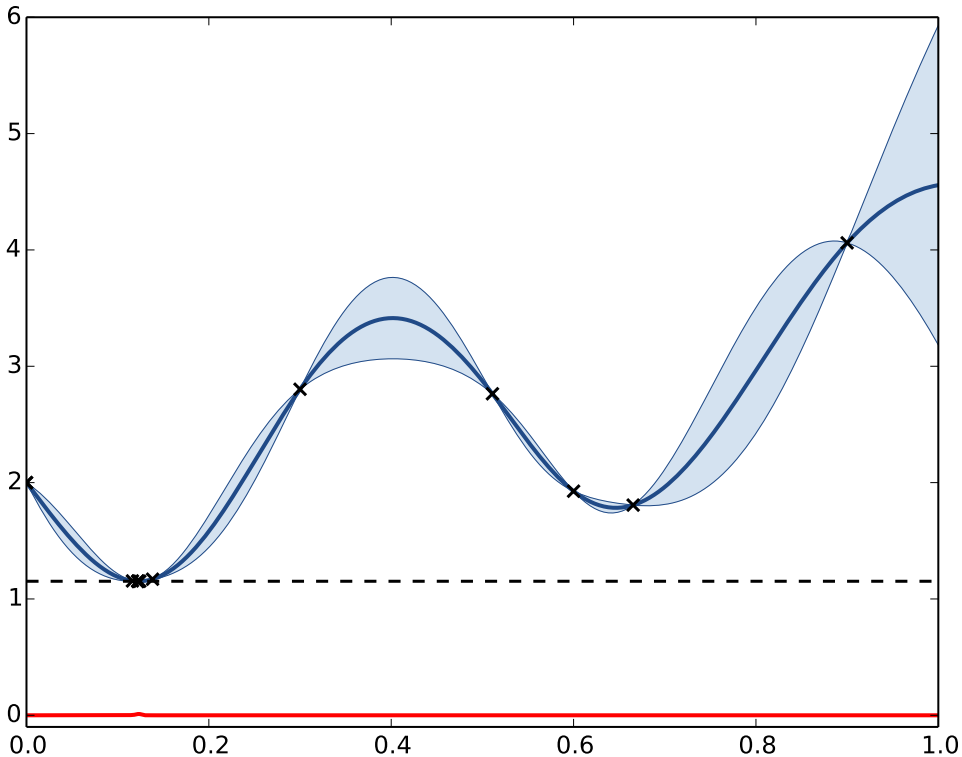
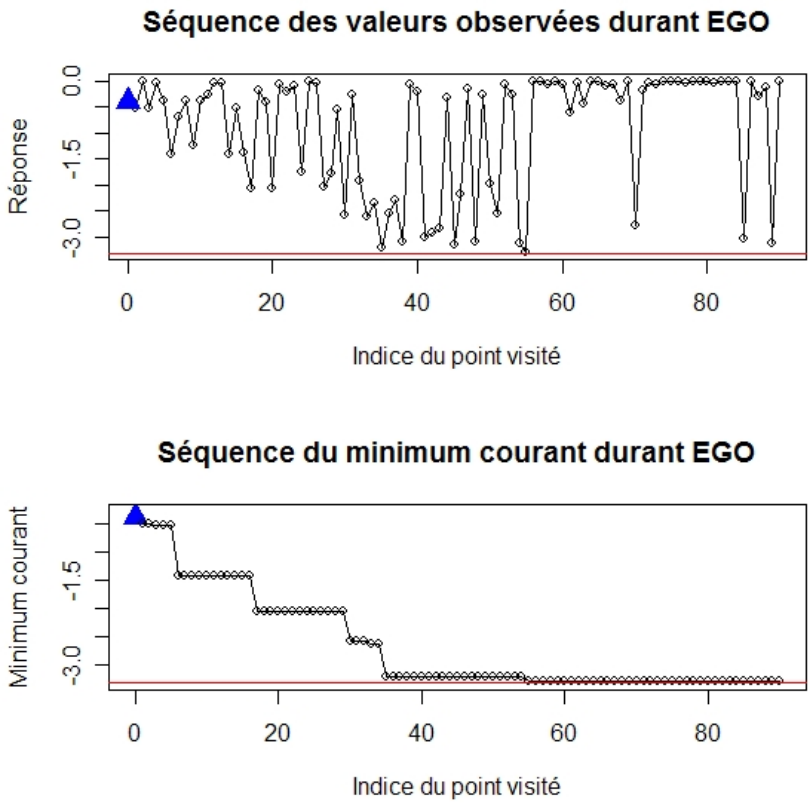
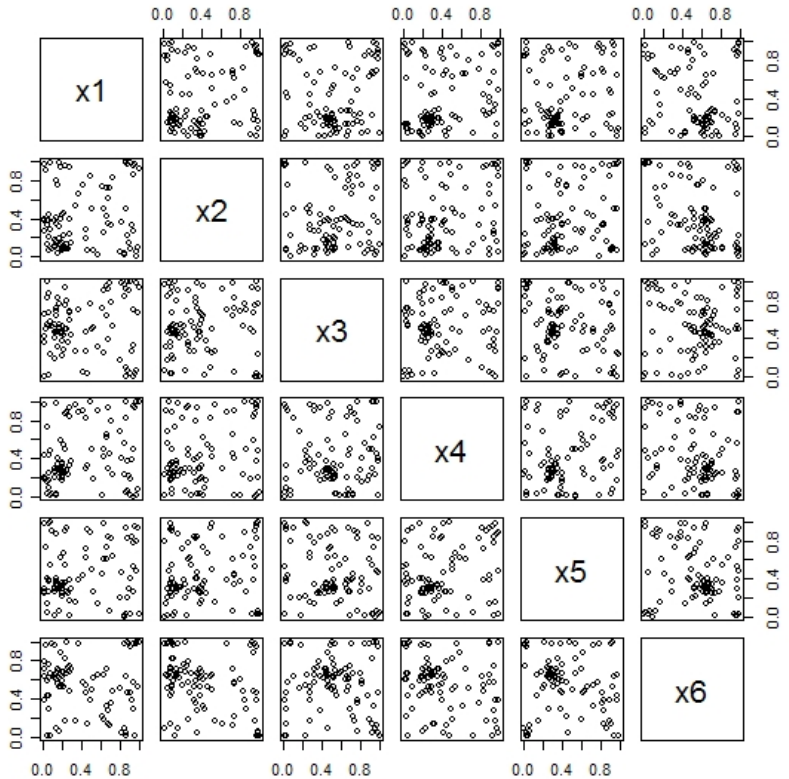


Illustration for $d = 6$ (Hartman)

Illustration in higher dimension





Source: DiceOptim, D. Ginsbourger, 2009.

Expected Improvement

This algorithm is called **Efficient Global Optimization** (EGO). It is famous since a paper of Jones et Al in 1998.

+ EGO provides a good trade-off between exploitation and exploration.
+ It only requires a few function observations (10 in the example)

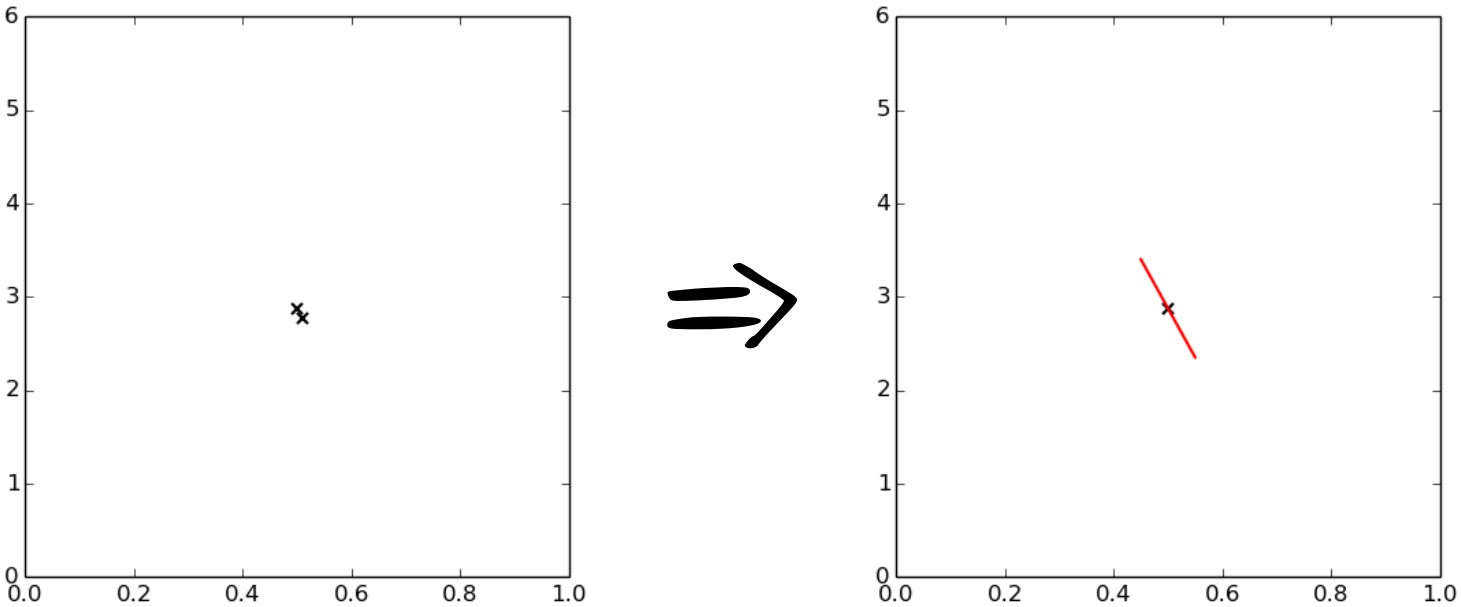
One issue is that we may have a model with observations very close one from each other

Example

From the previous 5 iterations, we obtain $1.44e39$ for the conditioning of the covariance matrix. Eigenvalues are

$$(67.70,\ 24.86,\ 5.13,\ 1.68,\ 0.45,\ 0.16,\ 0.01,\ 0.00,\ 0.00,\ 0.00)$$

One way to improve the conditioning of the covariance matrix is to replace two values that are close-by by one function value and one derivative:



Cond. $= 3842$ Cond. $= 10$

This can be generalised to higher orders $\rightarrow$ Taylor expansion

see articles from M. Osborn

If we know the computational budget in advance, adding new points at the **best one step ahead location** is not optimal.

Some improvements have been made toward this

- Batch EGO
- Parallelization of the algorithm

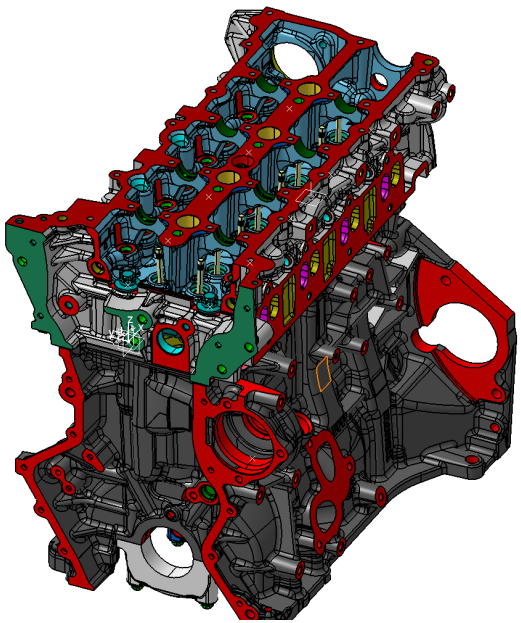
see works from D. Ginsbourger

Robust optimization

Robust optimization may mean various things:

- There is observation noise on the output
- Some input variables are uncertain
- Model is uncertain

Example

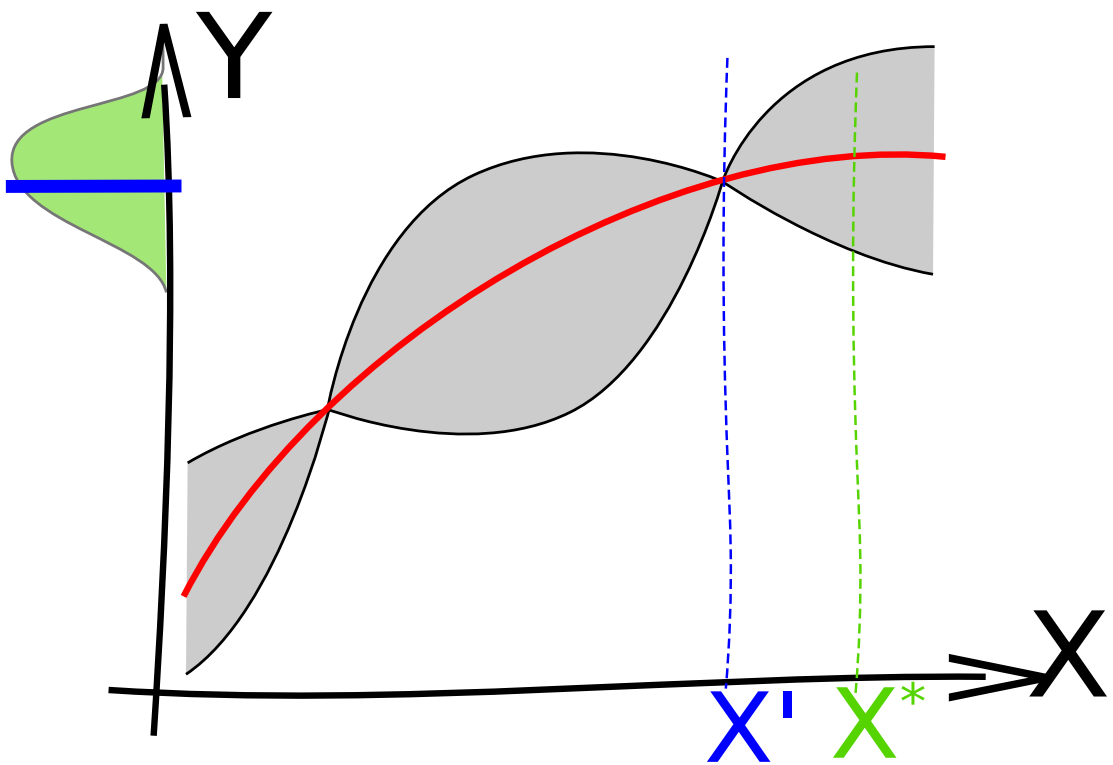


a +/- 1mm dispersion in the manufacturing of a car cylinder head can degrade its performance (g CO2/km) by -20% (worst case)

Source: Talk from R. Le Riche at the Porquerolles Summer School, 2014

Example

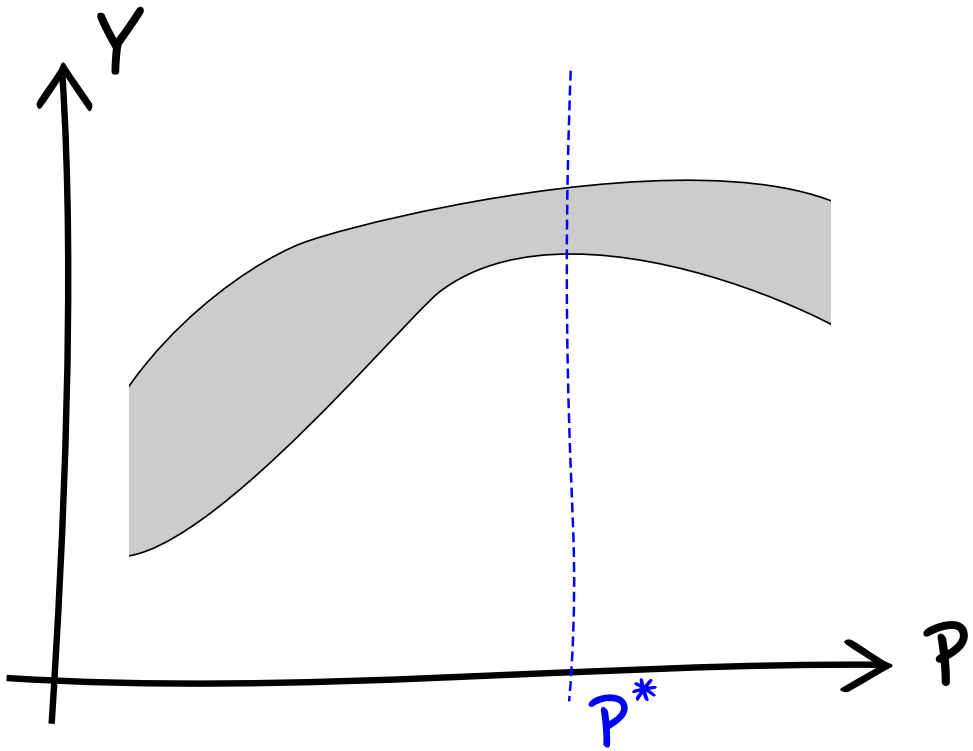
Here is a basic example:



Which input is the best?

Example

A non Gaussian example



In some cases, we may want to optimize the worst case scenario.

Can EGO be adapted when observations are noisy?

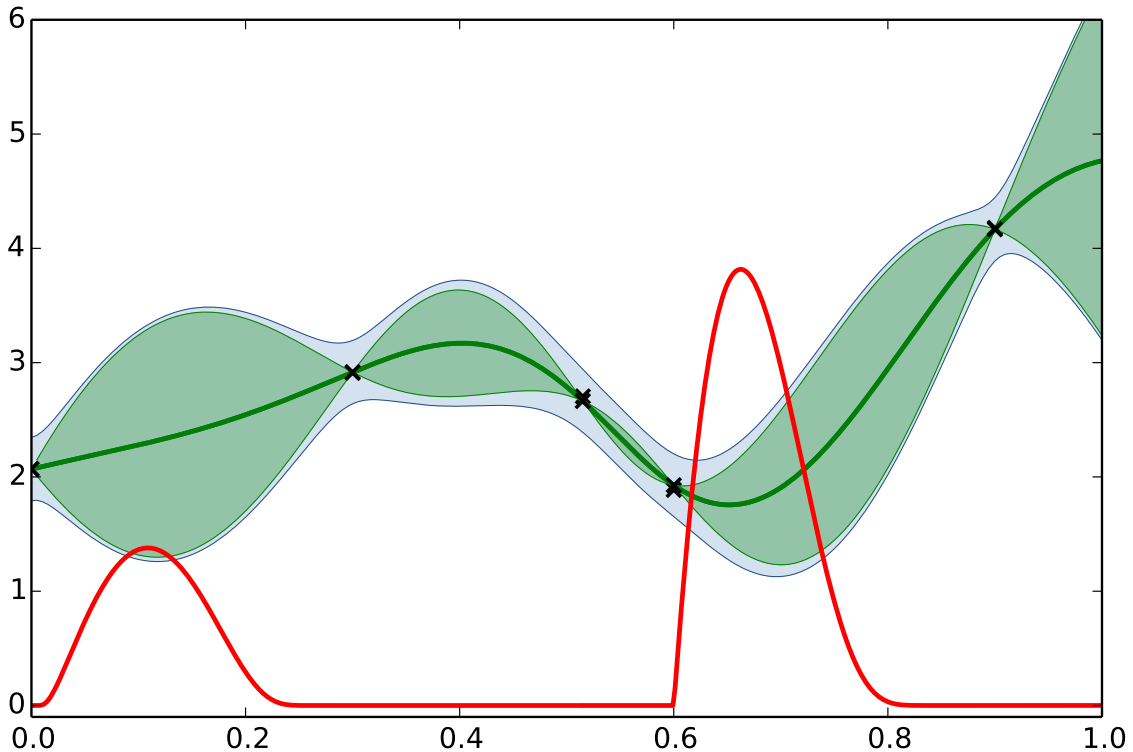
First of all, using the current best observation as a minimum does not make much sense...

Some solutions are

- S1 Build a new model that interpolates $m(X)$ at X.
- S2 Include observation noise and replace $\min(F)$ by $\min(m(X))$ in the EI expression
- S3 Similar to 2 but consider an Expected Mean Improvement.

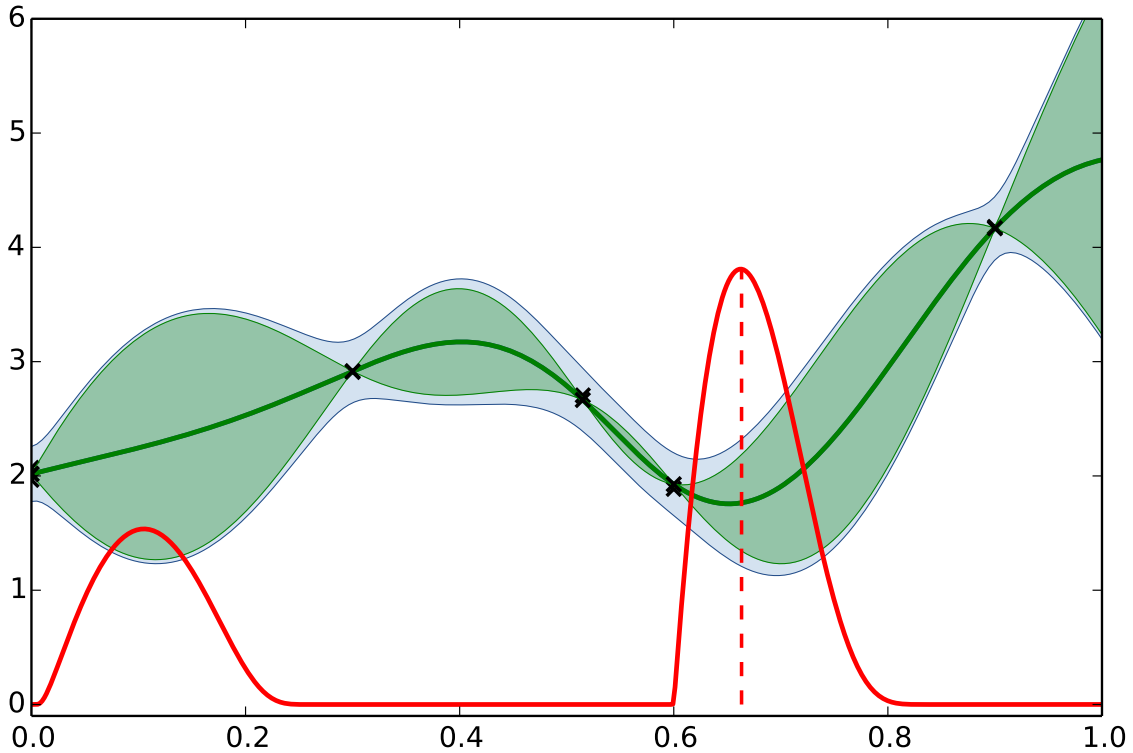
Solution 1

iteration 0



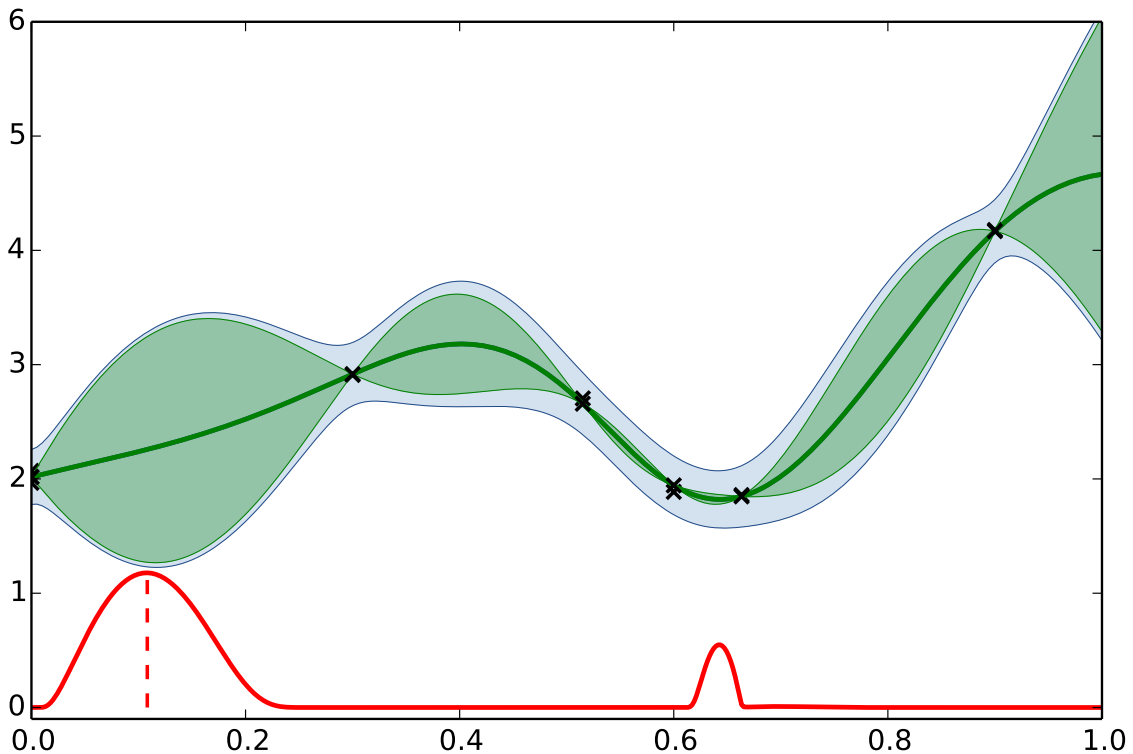
Solution 1

iteration 1



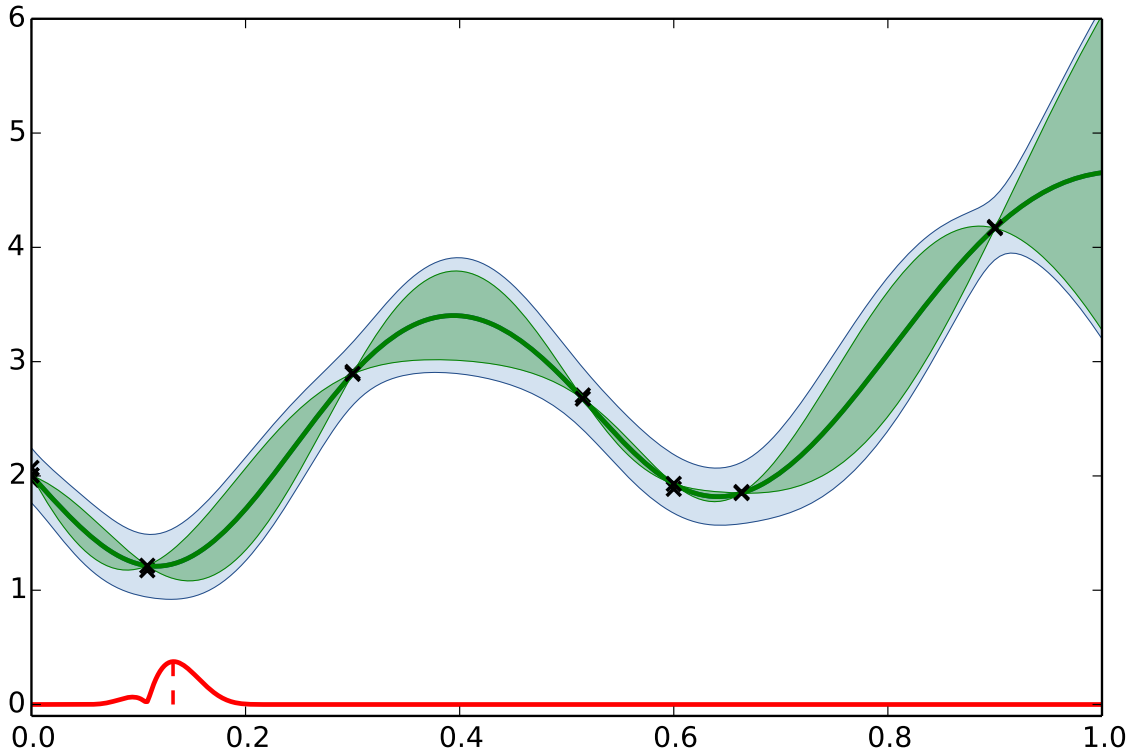
Solution 1

iteration 2



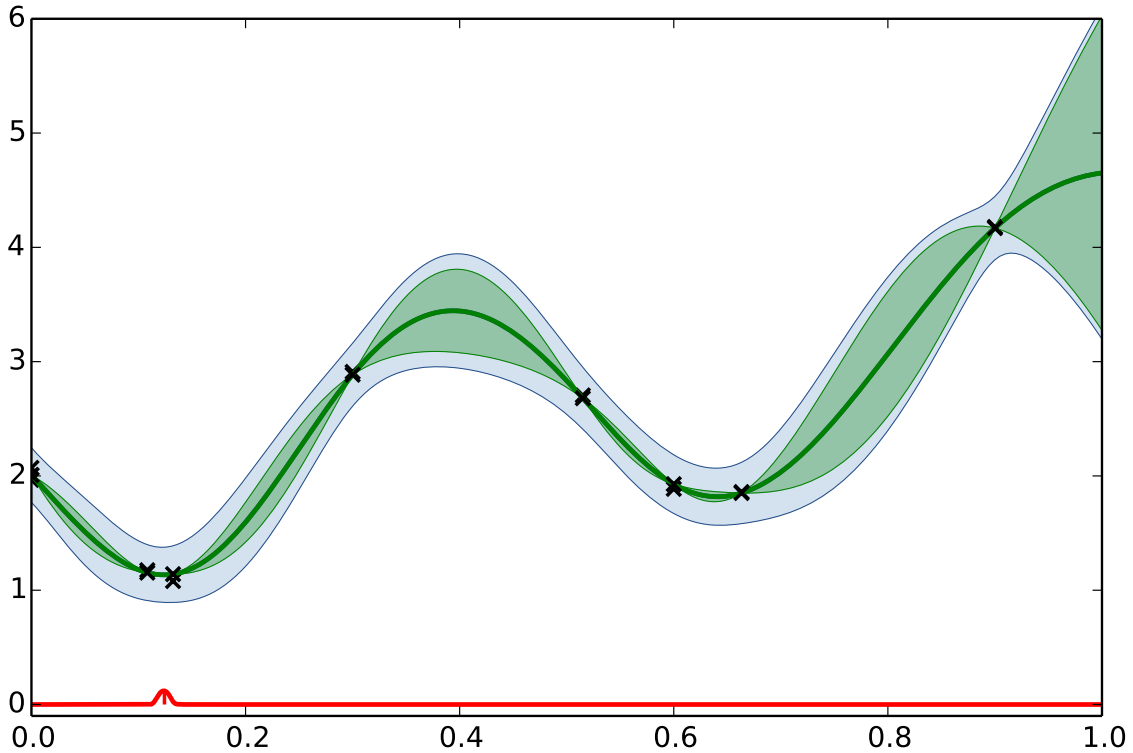
Solution 1

iteration 3



Solution 1

iteration 4



Related problems

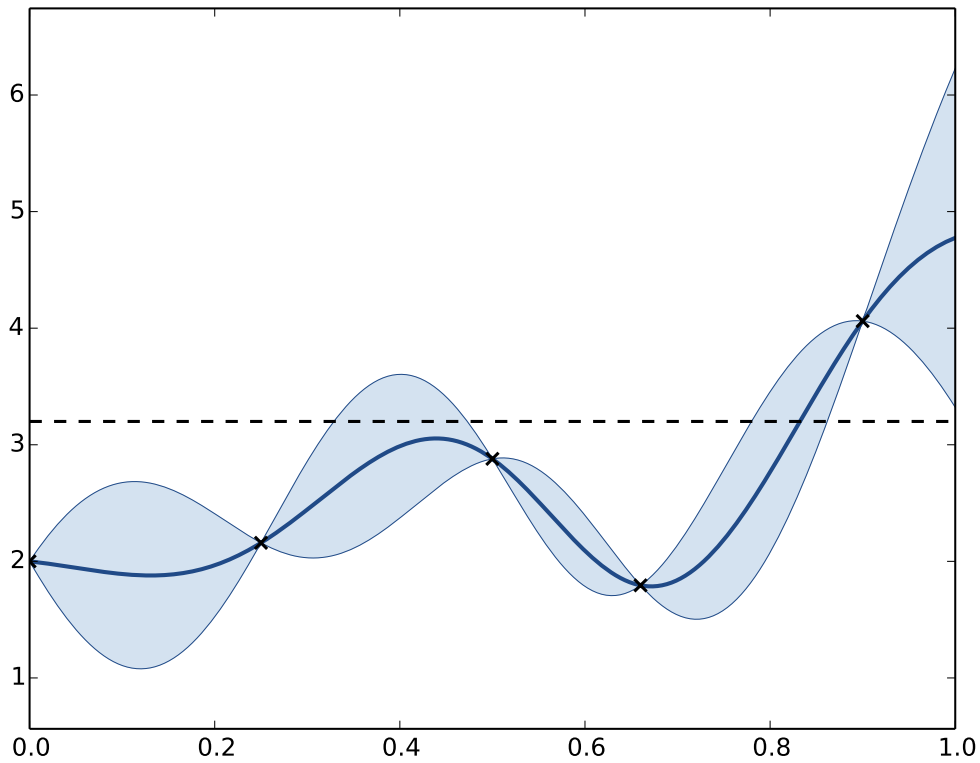
Some related optimization problems are:

- calibration problems
- probability computations

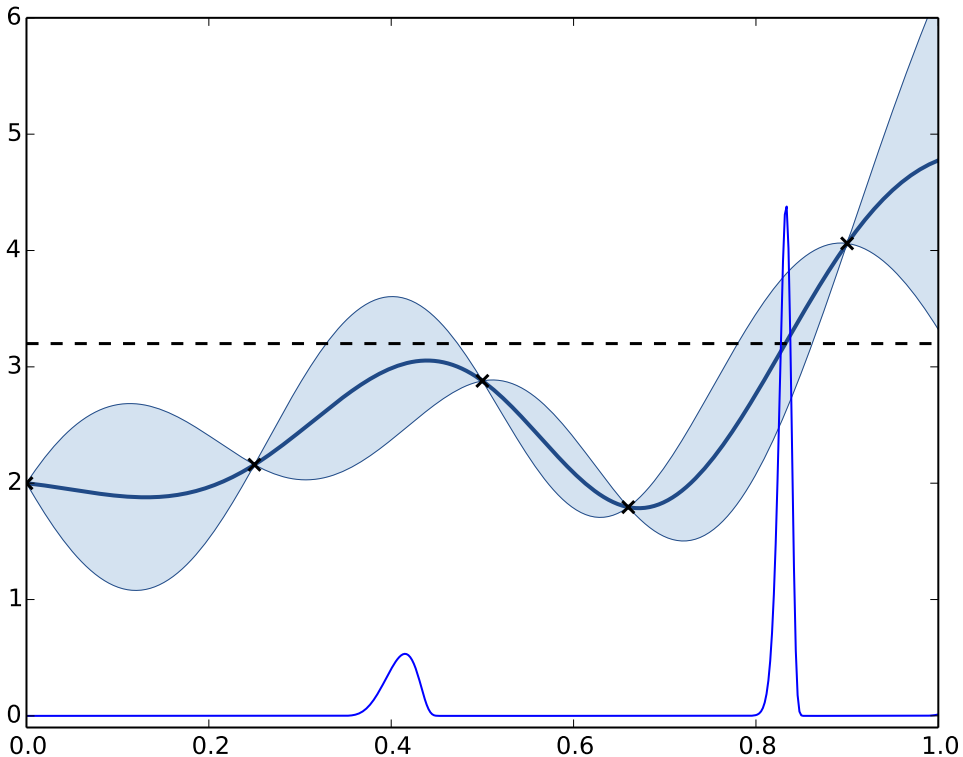
Some algorithms with an EGO spirit can be applied:

- SUR methods

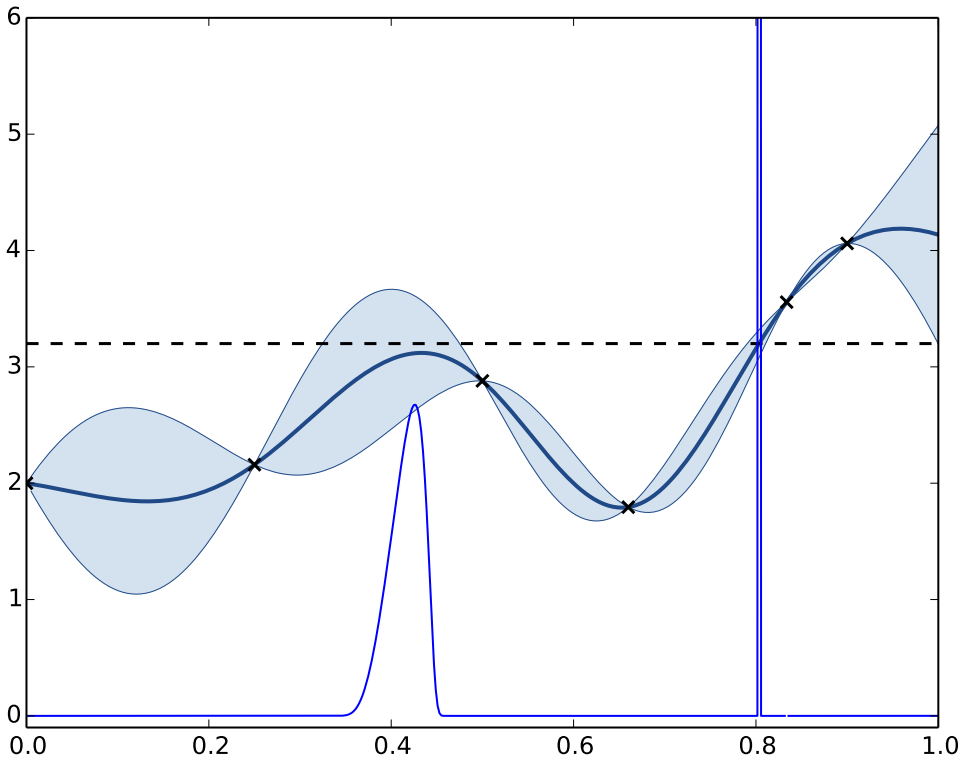
We want to find the input(s) such that f(x) = 3.2



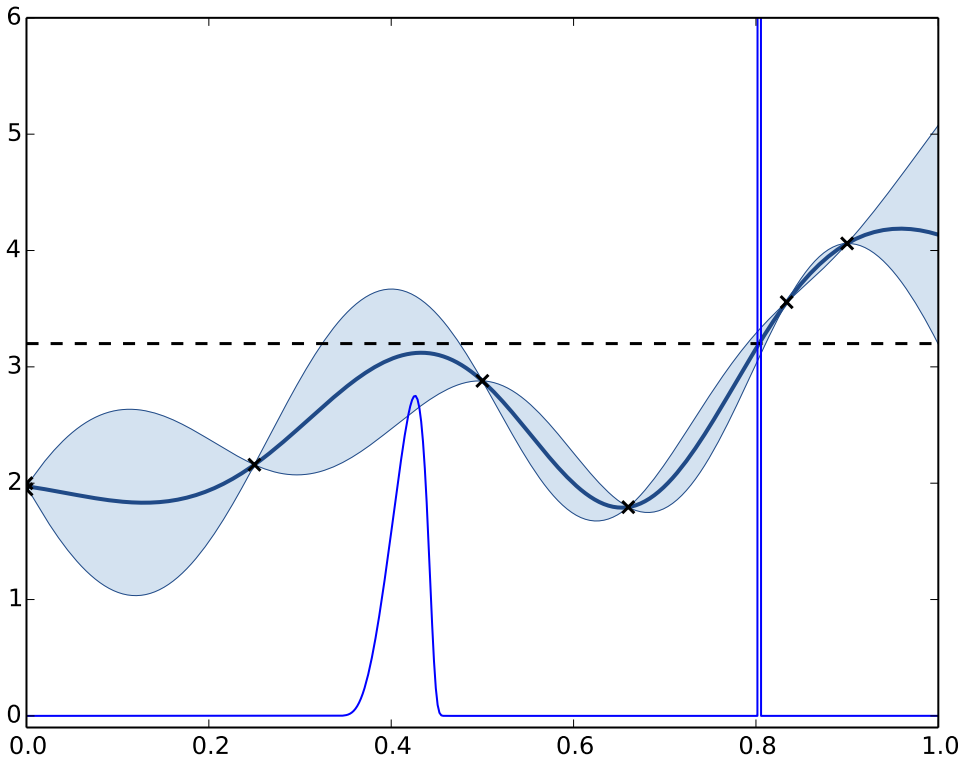
iteration 0:



iteration 1:



iteration 2:



iteration 3: