Advanced numerical engineering

& Gaussian Process

Y. Richet (yann.richet@irsn.fr)

credits: N. Durrande, R. Le Riche, V. Picheny, N. Garland

2018

...

- General metamodel engineering approach
- Optimization basics & engineering context
- Inversion basics & engineering context
- Improved numerical engineering algorithms & models

Overview

What *was/is* engineering ?

- Before 70's
  Analytical approach to *roughly* design nuclear plants, rockets, cars, …

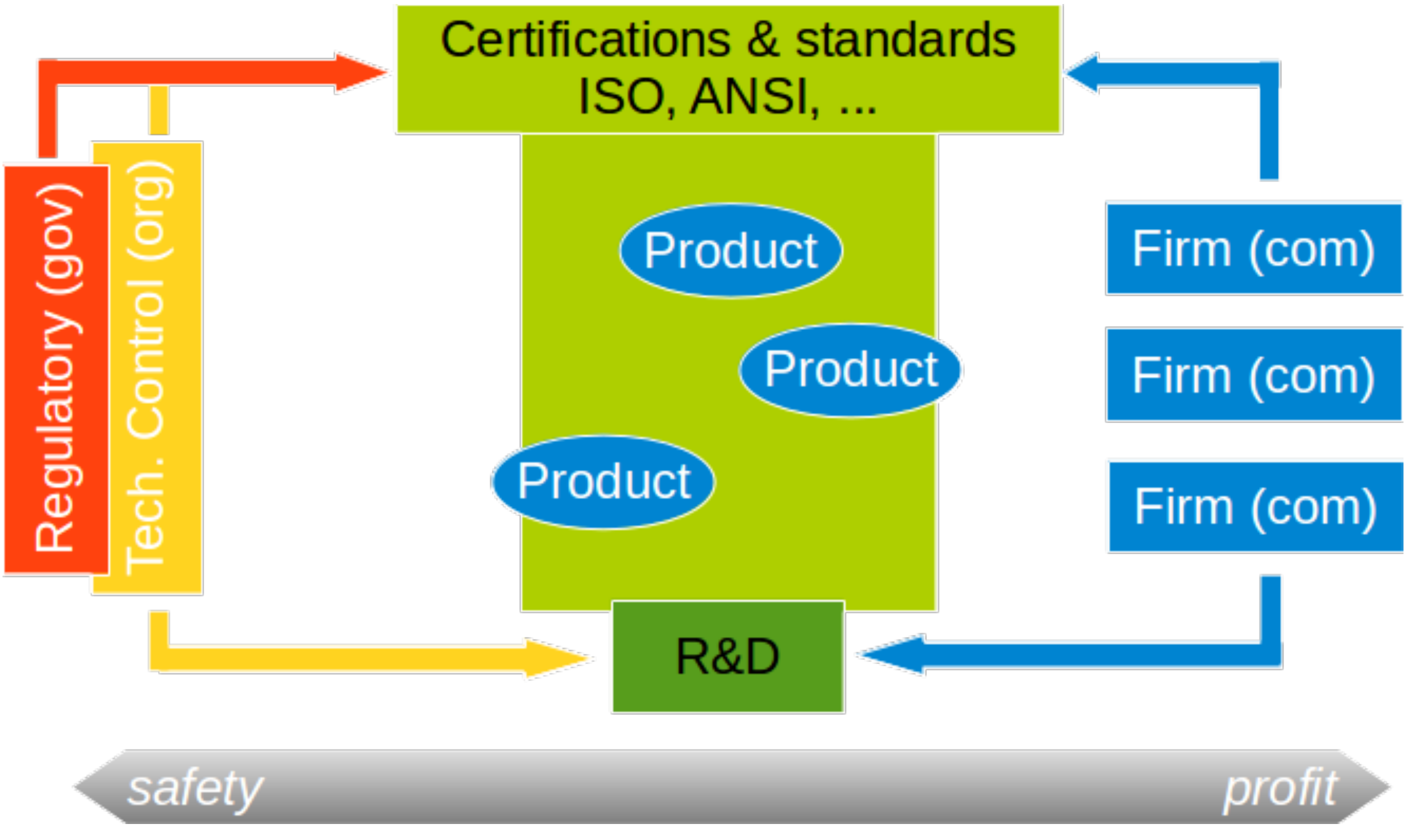
The Unreasonable Effectiveness of Mathematics, E. Wigner

- Since 70-80's Experimental/numerical engineering to simulate (vs. solve) fine multi-physics, unstable systems, …

"I am an old man now, and when I die and go to heaven there are two matters on which I hope for enlightenment.
One is quantum electrodynamics, and the other is the turbulent motion of fluids.

And about the former I am rather optimistic.", H. Lamb

Engineering: industry & standards



Engineering: industry *vs.* standards

Industrial products:

- wide product target identification
- optimization of products profitability
- optimization of investments

Regulatory control:

- define industrial-field standards
  - end-user safety
  - ecological & resources parcimony
  - (fair market)
- check products conformity
- evolve standards to best knowledge

Optimization (reminder)

$$f^* = min_{x \in S \subset \mathbb{R}^d} \; f(x)$$

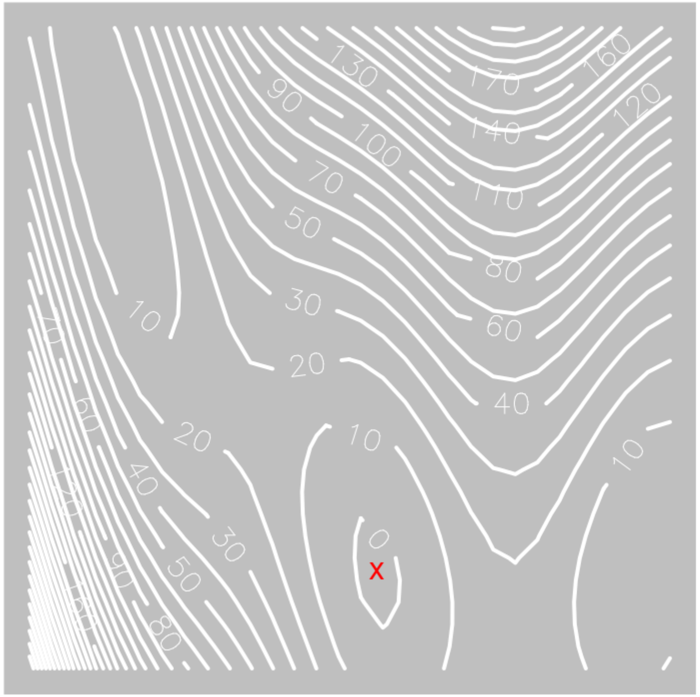
$$x^* = argmin_{x \in S \subset \mathbb{R}^d} \; f(x)$$

- S is the input domain for x
- d is the dimension of x
- f is the cost function (ex. stress, risk, power, temperature, …)

---

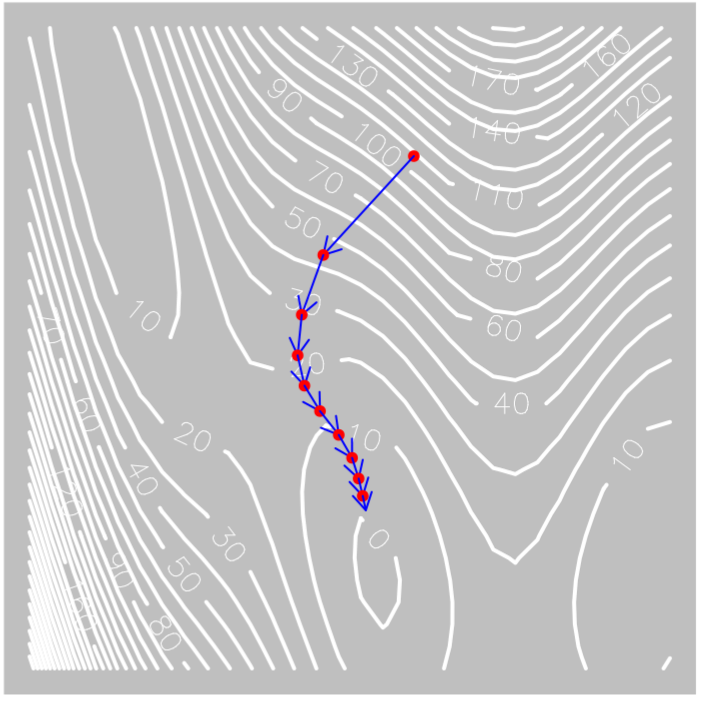
for sake of simplicity and by convention, we consider **min***imization*

Optimization



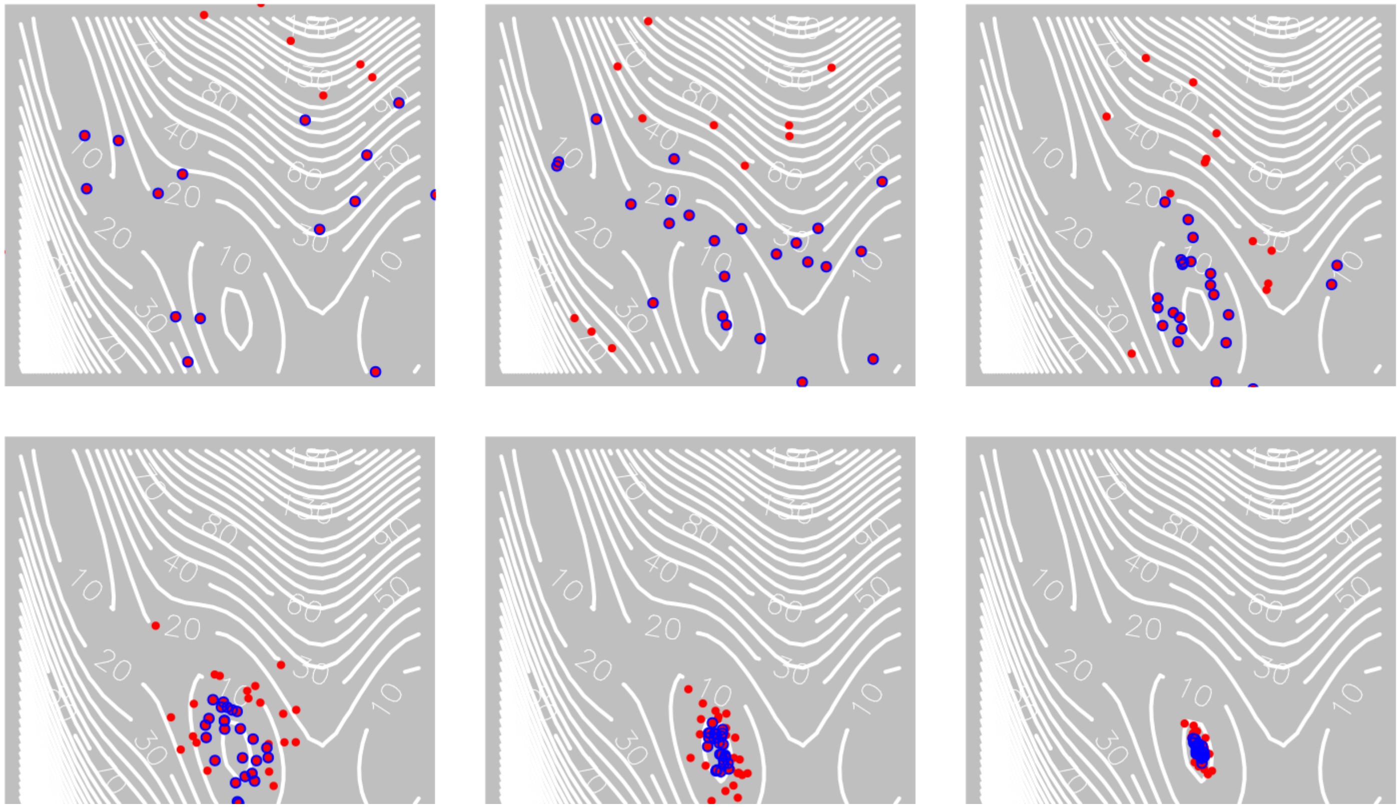
Optimization - basics

Gradient / Newton methods



Optimization - basics

Evolutionnary algorithms (CMA-ES, PSO, ...)



Optimization - basics vs. engineering

Optimization in **engineering** context raises many issues:

- one f evaluation is expensive (CPU/time/user)
- dimension of problem input d may be large
- time to result is constrained
- added-value from optimization is initially *unknown*

But **basic** methods are not suitable:

- Gradient / Newton methods
  - gradients have to be computed (finite differences ?)
  - *local* optimization, while we need *global*
  - handle noise of f evaluation ?
- Evolutionnary algorithms

Optimization - basics vs. engineering

Optimization in **engineering** context raises many issues:

- one f evaluation is expensive (CPU/time/user)
- dimension of problem input d may be large
- time to result is constrained
- added-value from optimization is initially *unknown*

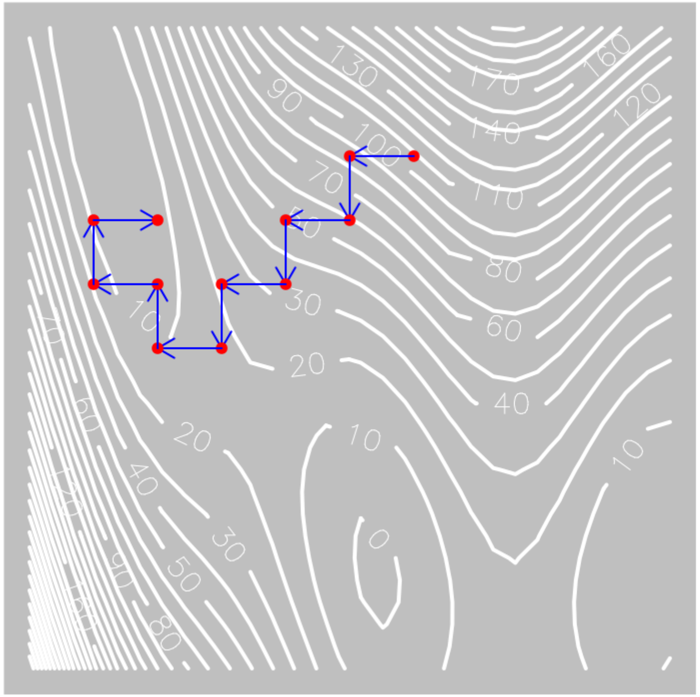
But **basic** methods are not suitable

So, **true** practice is often very rough:

- change one parameter at a time (to avoid mistakes)
- f evaluations performed should be enough different
- startup with some assumptions on the solution
    - intrinsinc physics

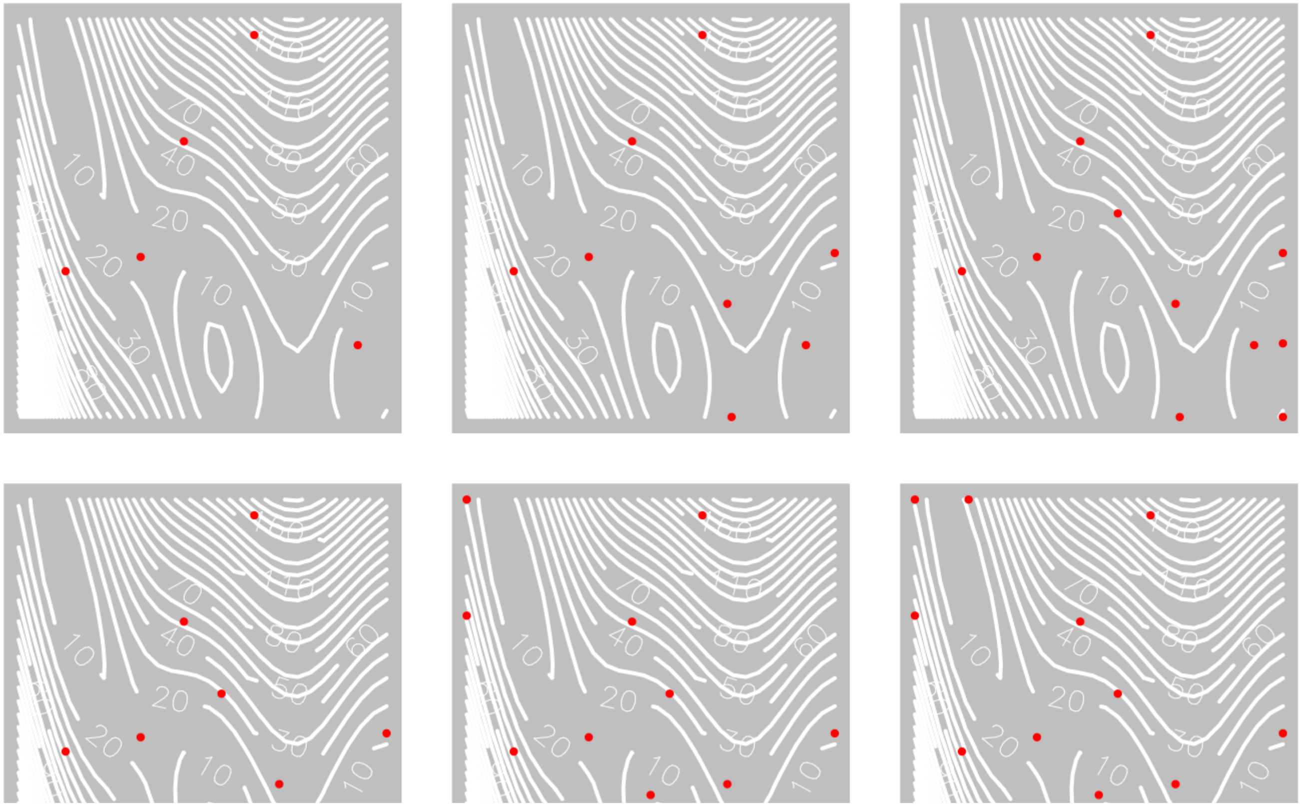
Optimization - engineering

One-at-a-time optimization



Optimization - engineering++

Bayesian optimization (EGO,...)



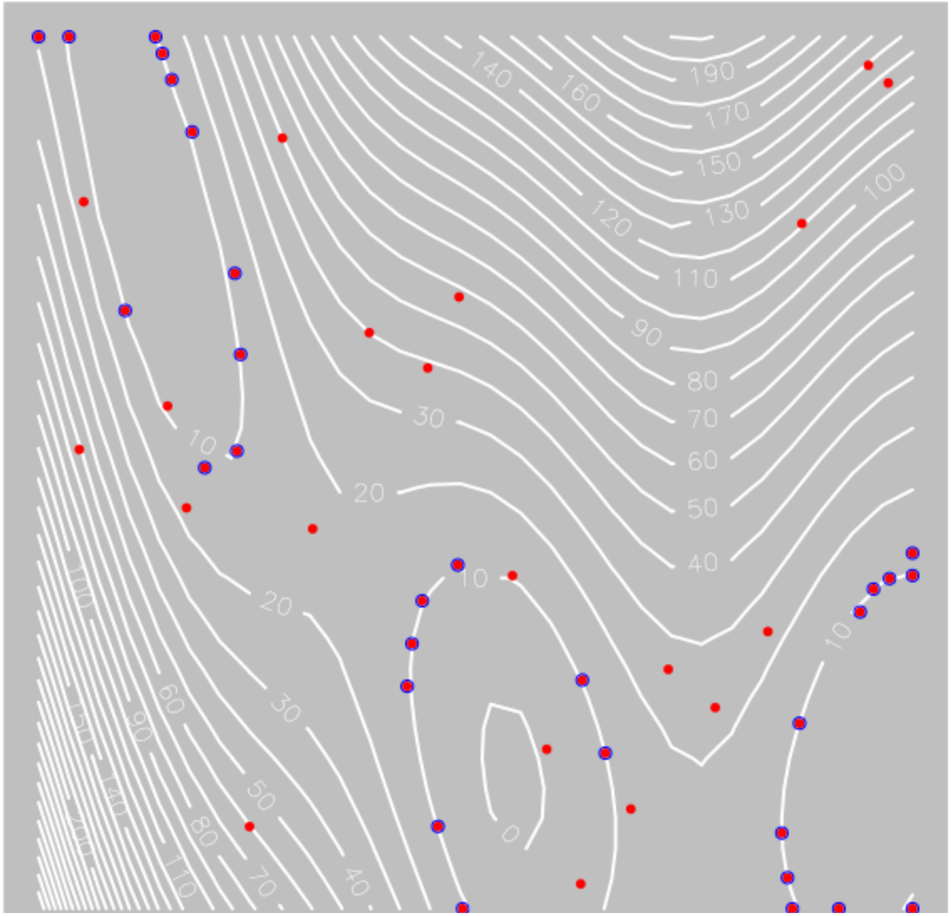
Inversion

$$\{x^*\} = arg_{x \in S \subset \mathbb{R}^d} \ \{f(x) = T\}$$

$$\{x^*\} = arg_{x \in S \subset \mathbb{R}^d} \ \{f(x) < T\}$$

- S is the input domain for x
- d is the dimension of x
- f is the cost function (ex. stress, risk, power, temperature, …)

Inversion



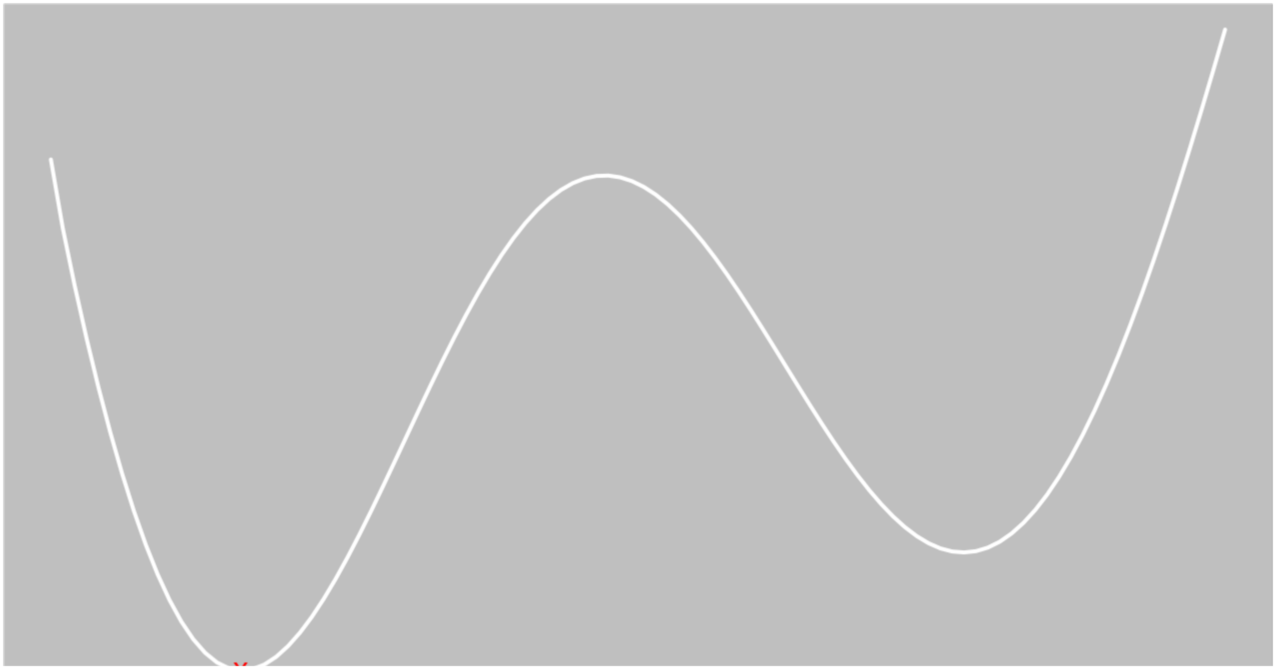
Numerical engineering algorithms…

- uncertainties propagation:
  - *"What spreading of output when inputs follow given random laws ?"*
  - random sampling (~ Monte Carlo), dedicated models (polynomial chaos)
- sensitivity analysis
  - *"What share of output spreading is due to random inputs ?"*
  - relative indices: $V[S[Y|X]]/\ldots$ (Sobol, HSIC, …), random sampling, dedicated models (Fourier, polynomial chaos)
- parameters screening
  - *"What output behaviour for each input ?"*
  - screening designs (sparse), OaT designs (Morris)
- optimization
  - *"What worst/best value possible for output ?"*
  - bayesian optimization (EGO)
- inversion

Metamodel based engineering

Basic idea

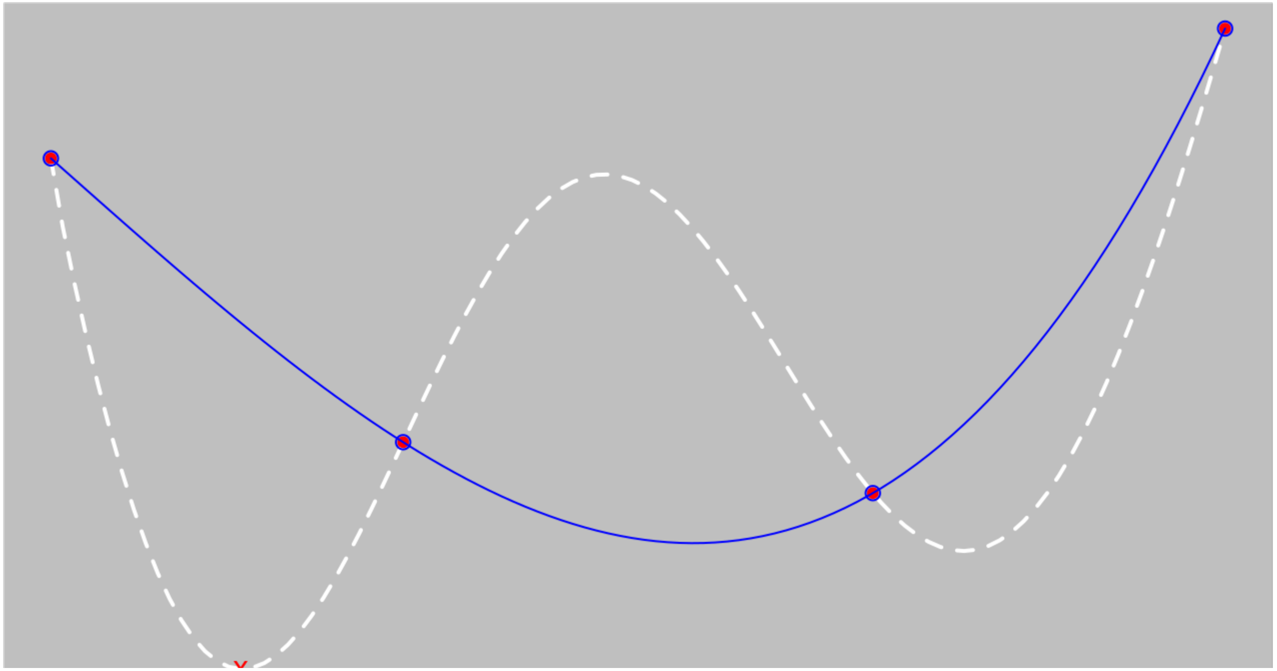
f



Basic idea

To limit number of f evaluations:

- create a *model of f* based on few evaluations $x = X$



Basic idea

To limit number of f evaluations:

- create a *model of* f based on few evaluations $x = X$

But it is risky to take decision only based on a single model …

… However, we would expect the minimum to be not so far from model's one.

Basic idea

To limit number of f evaluations:

- create a $\textit{model of } f$ based on few evaluations $x = X$

But it is risky to take decision only based on a model …

=> Diversifiy the model

… However, we would expect the minimum to be not so far from model's one.

=> Take model minimum just as a clue

Basic idea

To limit number of f evaluations:

- create a $model\ of\ f$ based on few evaluations $x = X$

But it is risky to take decision only based on a model …

=> Diversifiy the model: [EXPLORE]

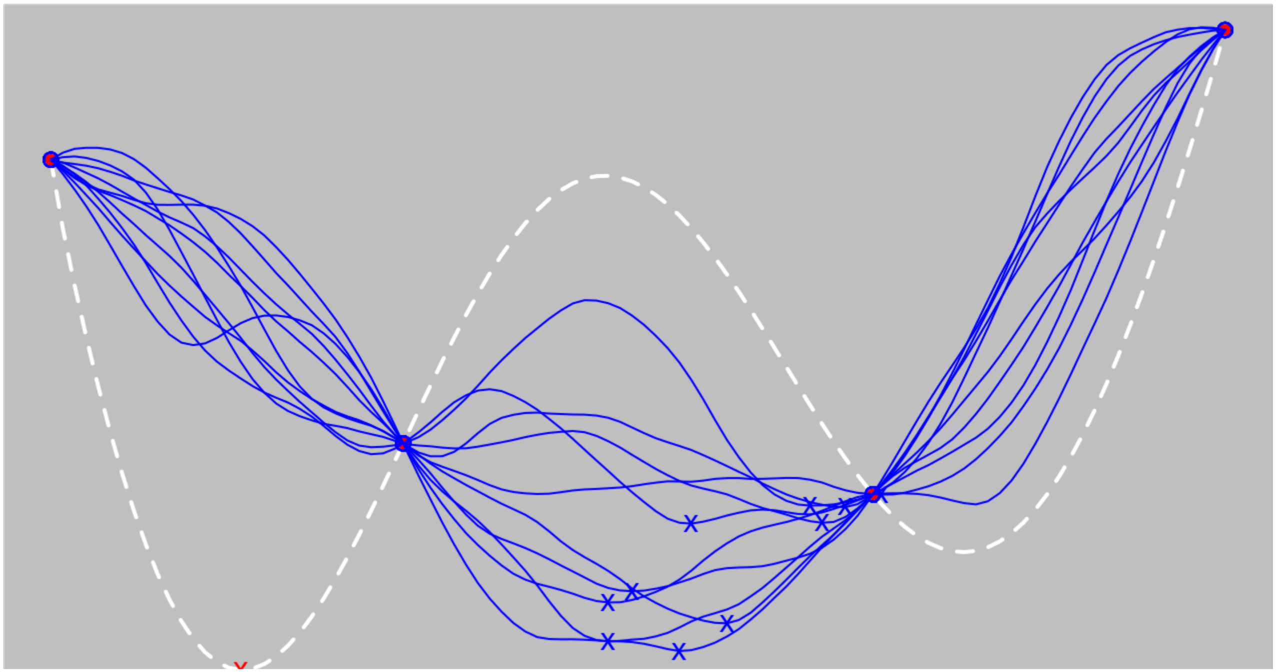
… However, we would expect the minimum to be not so far from model's one.

=> Take model minimum just as a clue: [EXPLOIT]

Basic idea

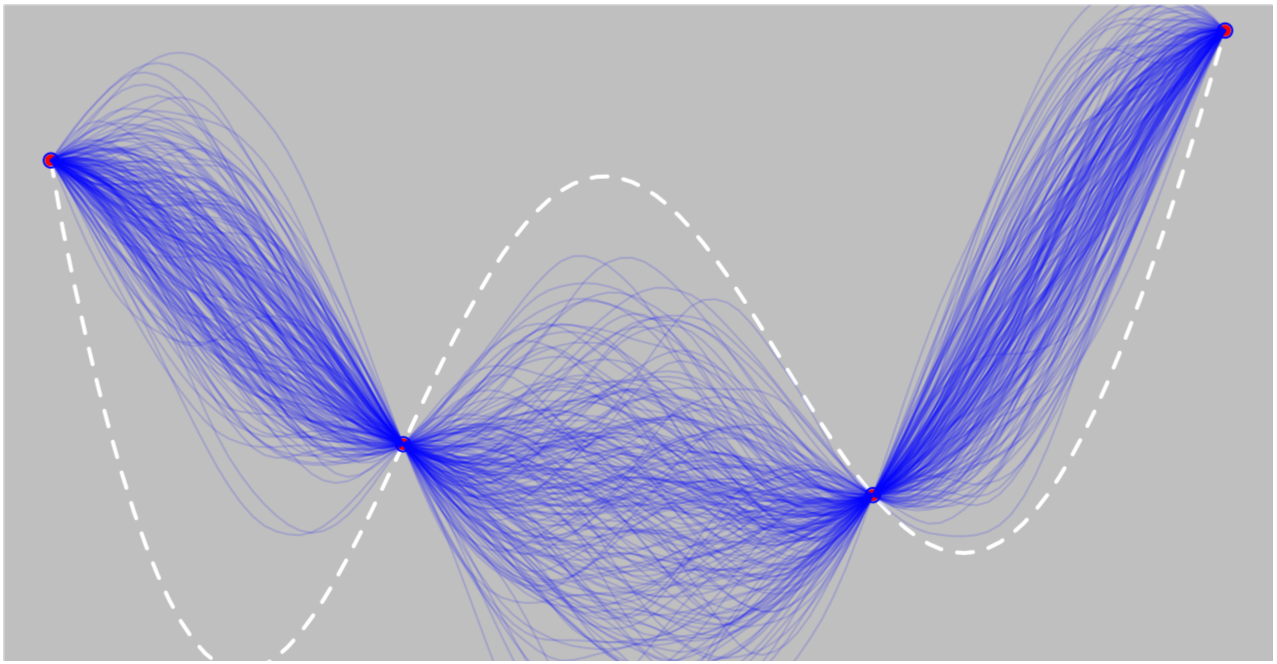
To limit number of f evaluations:

- create **many** *models of f* based on few evaluations $x = X$



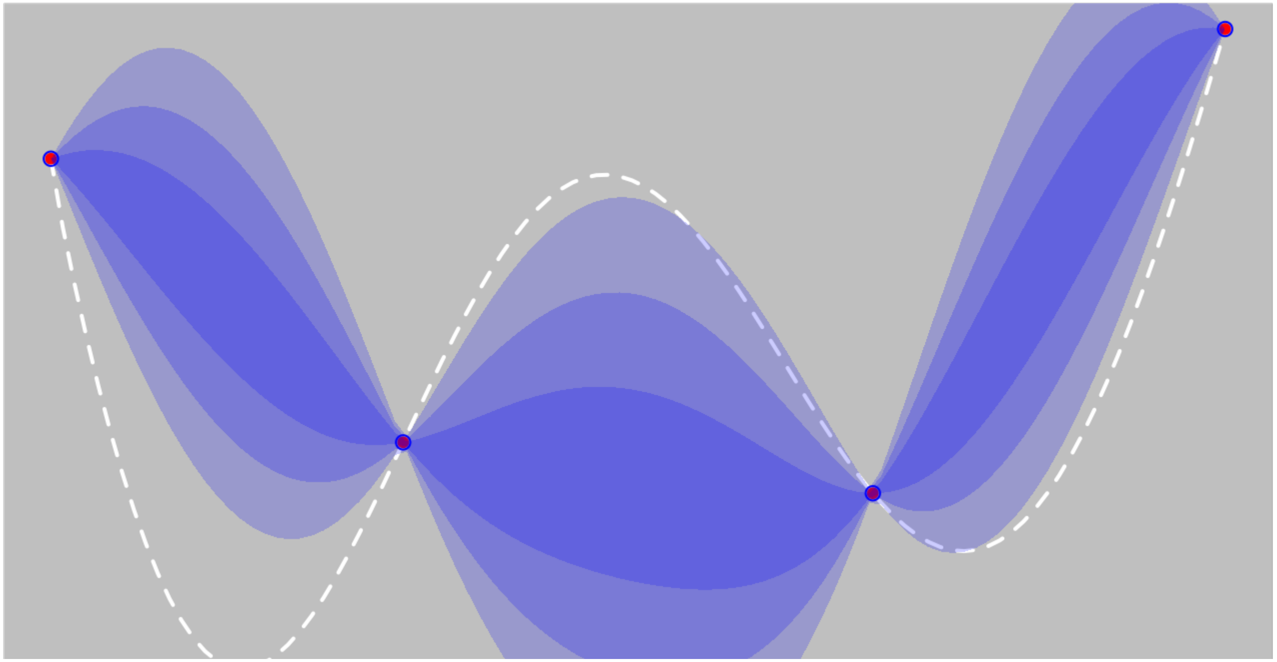
Conditional Gaussian Process

- Suitable to "interpolate" few evaluations
- Suitable to sample in a model family
-



Conditional Gaussian Process

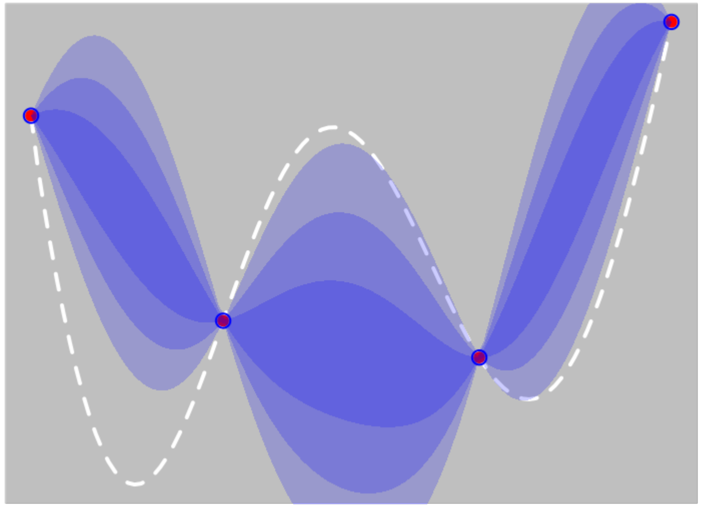
- Suitable to "interpolate" few evaluations
- Suitable to sample in a model family
- Give an **explicit** (Gaussian) density marginal on x



Conditional Gaussian Process

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



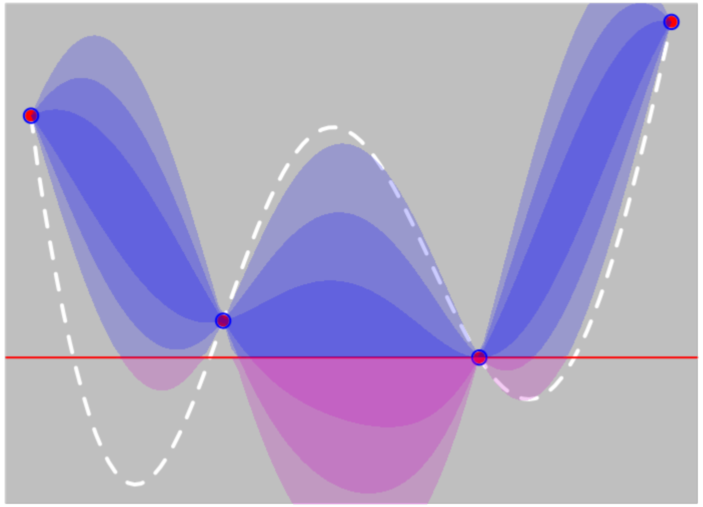
- define a suitable criterion for engineering target
- find its maximum over X
- sample f at this new point

Model *pointwise* mining

Optimization

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



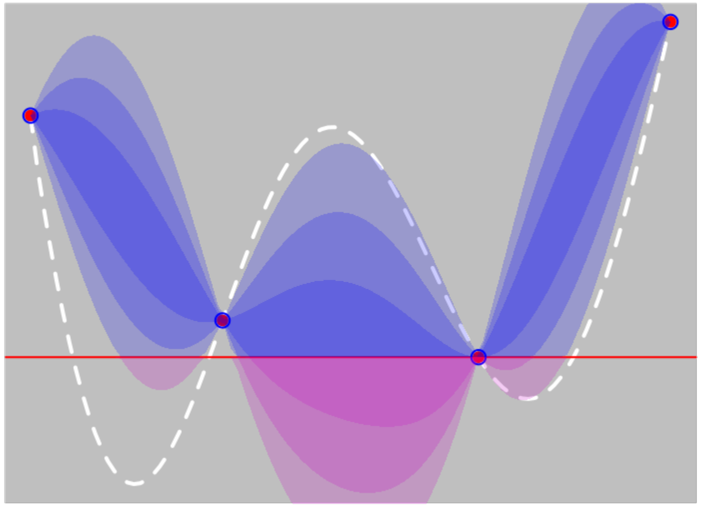
To trade-off between exploration/exploitation we will consider:

- distribution of $M(x)$: [EXPLORE]
- set of x where $M(x) < min\{f(X)\}$: [EXPLOIT]

Efficient Global Optimization

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



Let's define the *Probability of Improvement*:

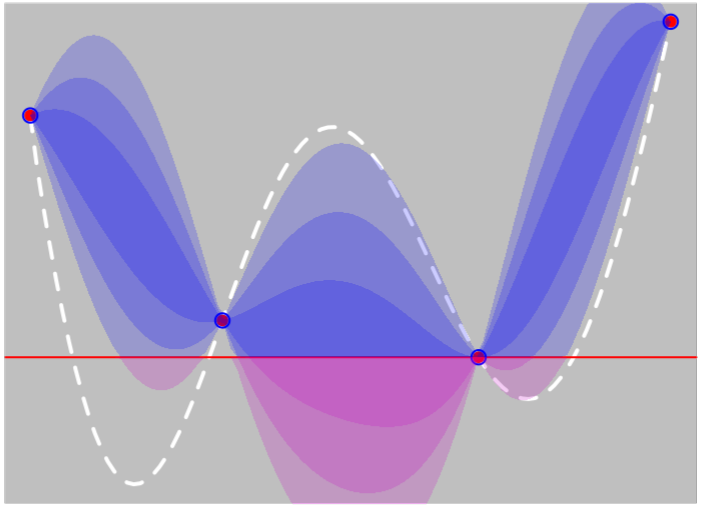
$$PI(x) = P[M(x) < min\{f(X)\}]$$

which is analytical thanks to M properties…

EGO (reminder)

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



$$u(x) = \frac{min\{f(X)\} - m(x)}{s(x)}$$

$$PI(x) = p_{\mathcal{N}}\left(u(x)\right)$$

EGO (reminder)

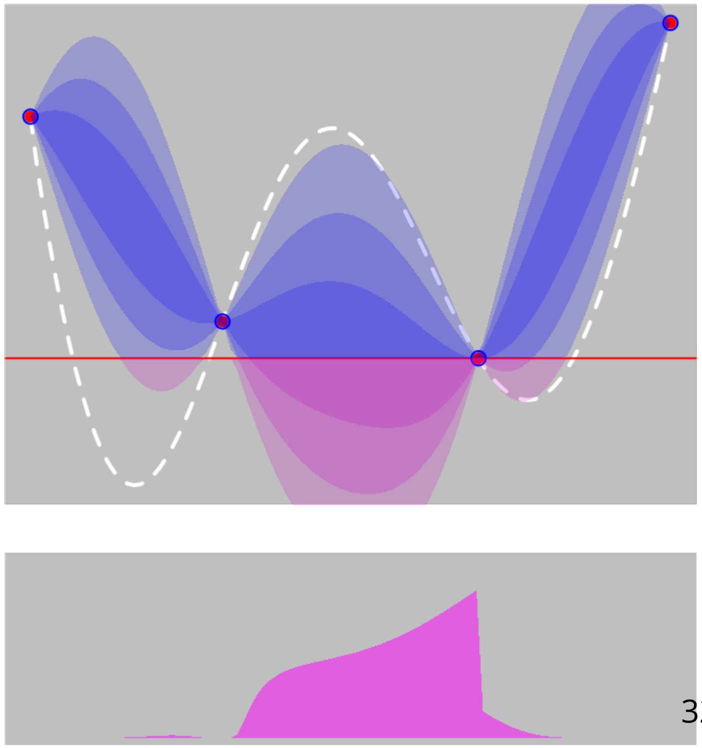
$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .), c(.) = C(x, .)$

$$PI(x) = P[M(x) < min\{f(X)\}]$$

But, x for highest PI is often

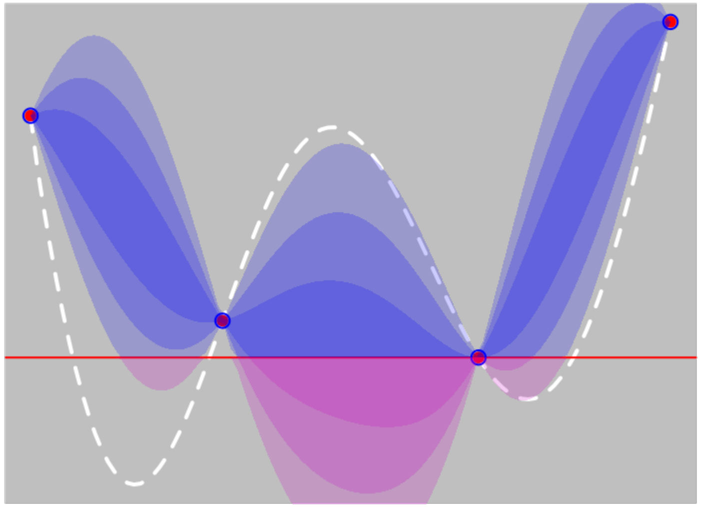
close to best X ...



EGO (reminder)

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



Let's define the *Expected Improvement*:

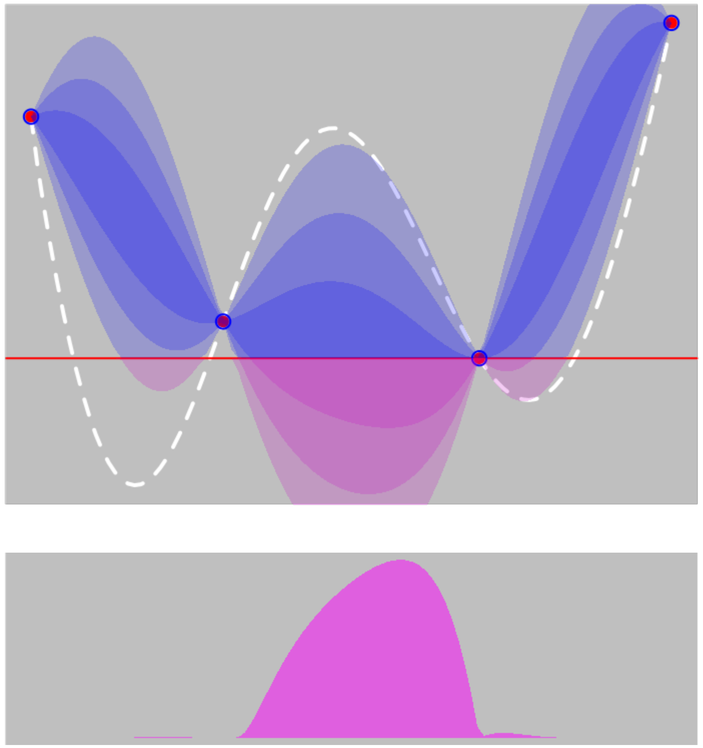
$$EI(x) = E[(min\{f(X)\} - M(x))^+]$$

which is (also) analytical thanks to M properties…

EGO (reminder)

$$\color{blue}{M}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X,.)$, $c(.) = C(x,.)$



$$u(x) = \frac{min\{f(X)\} - m(x)}{s(x)}$$

$$EI(x) = s(x) \left(u(x) p_{\mathcal{N}}(u(x)) + d_{\mathcal{N}}(u(x)) \right)$$

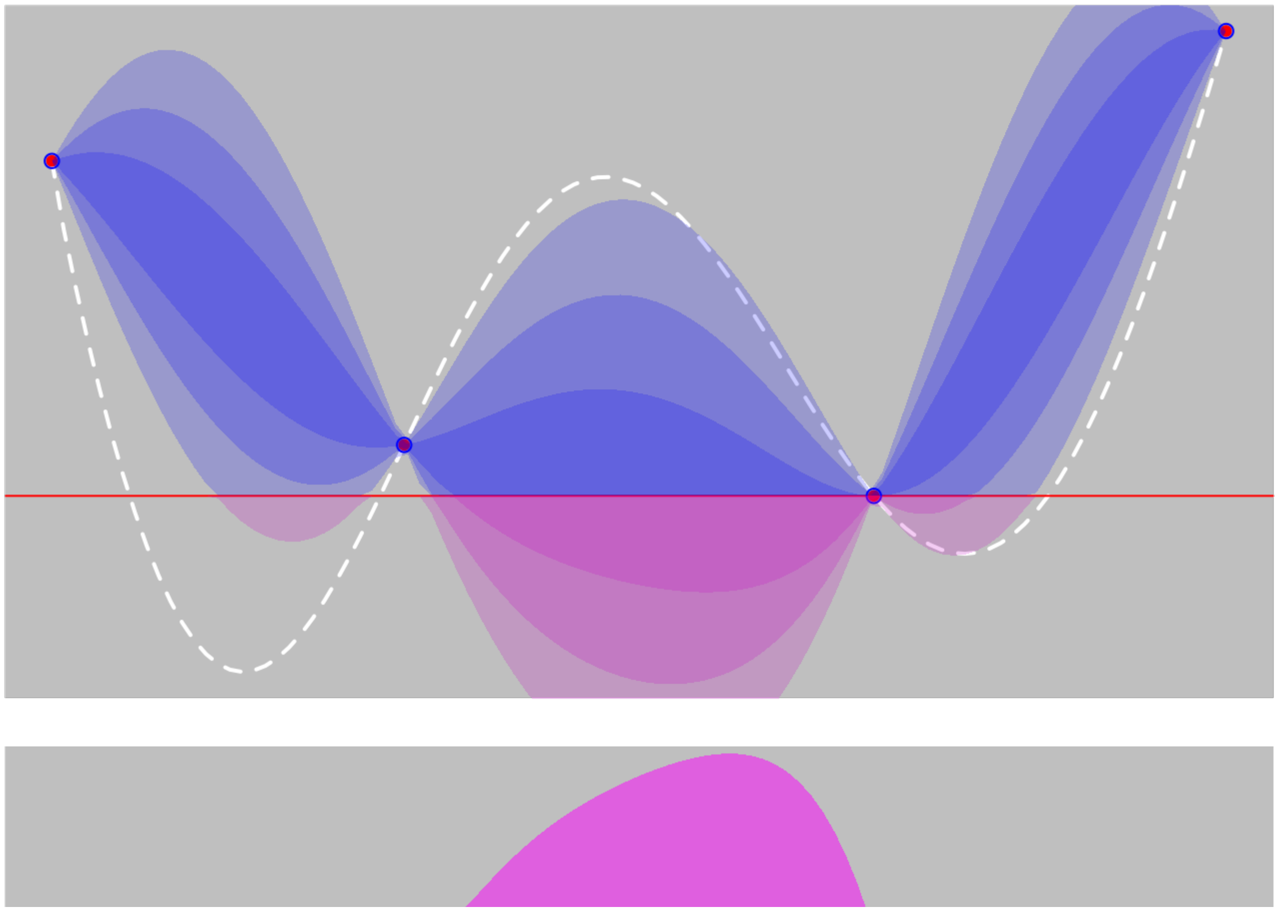
EGO (reminder)

"Efficient Global Optimization of Expensive Black-Box Functions"- Jones, Schonlau, Welch,
(Journal of Global Optimization, December 1998)

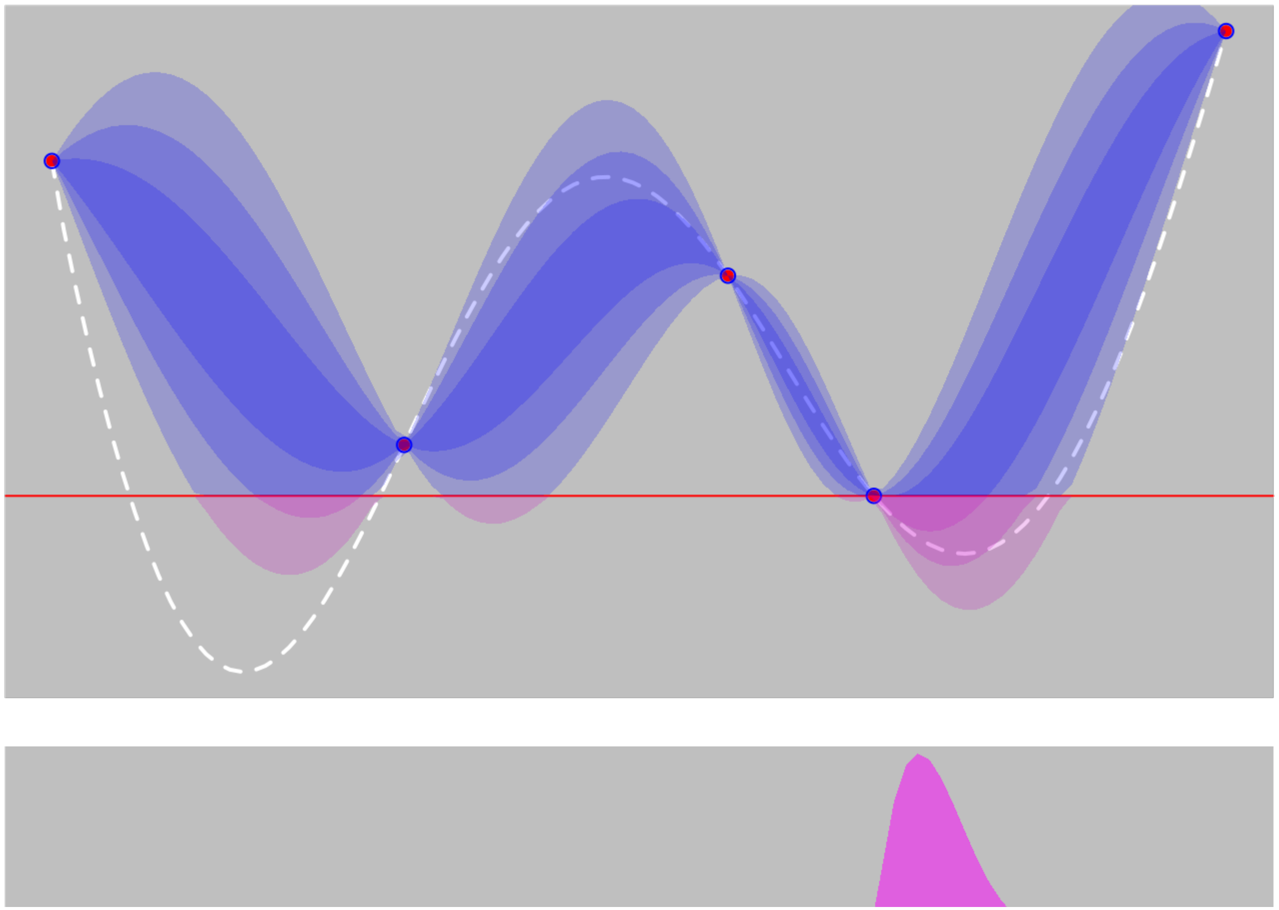
EGO:
Maximize $EI(x)$ (*), compute f there, add to X.
Repeat until …

- **+** good trade-off between exploration and exploitation
- **+** requires few evaluations of f
- **-** often lead to add close points to each others …
  Which is not very comfortable for kriging numerical stability
- **-** "one step lookahead" (myopic) strategy
- **-** rely on model suitability to f

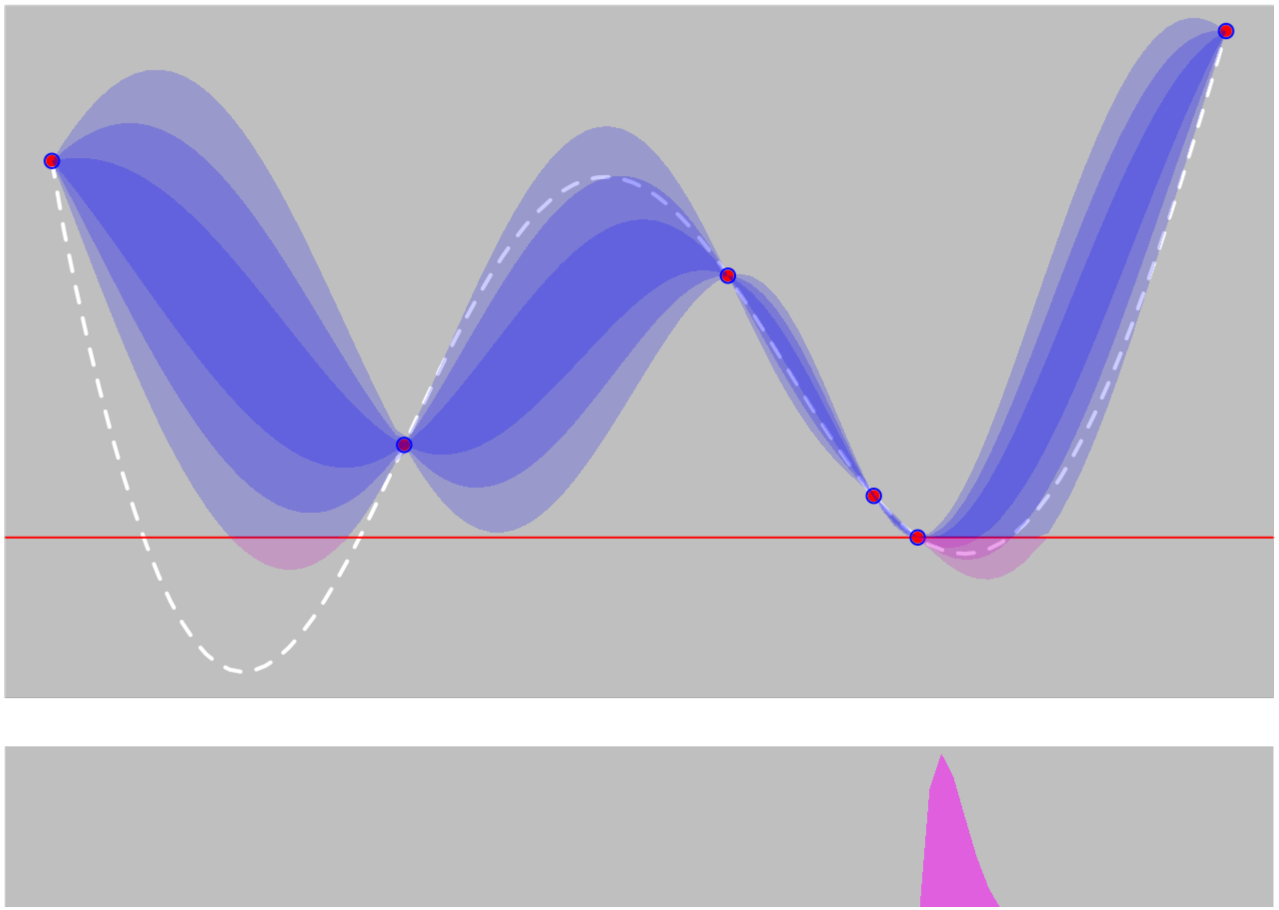
EGO (reminder)



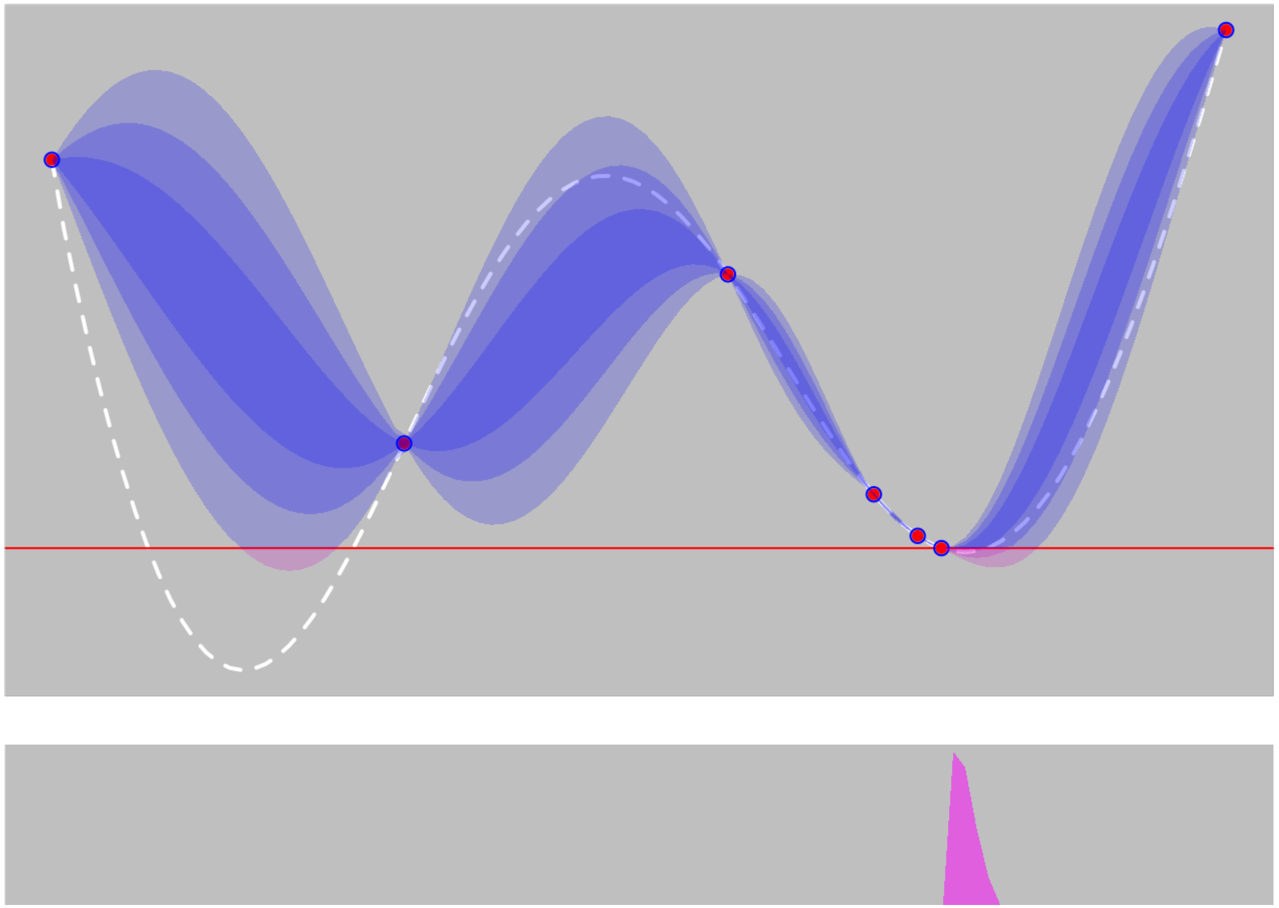
EGO (reminder)



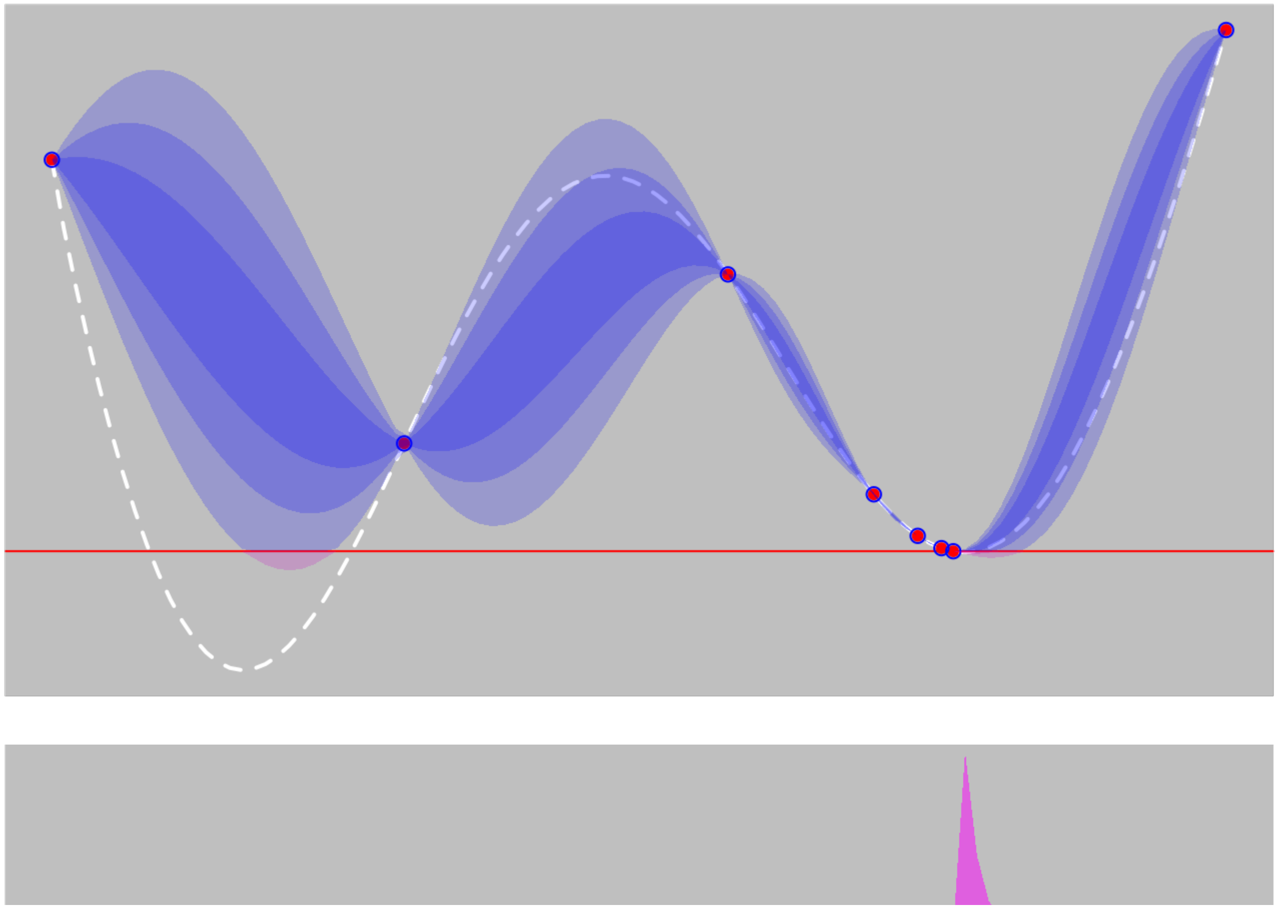
EGO (reminder)



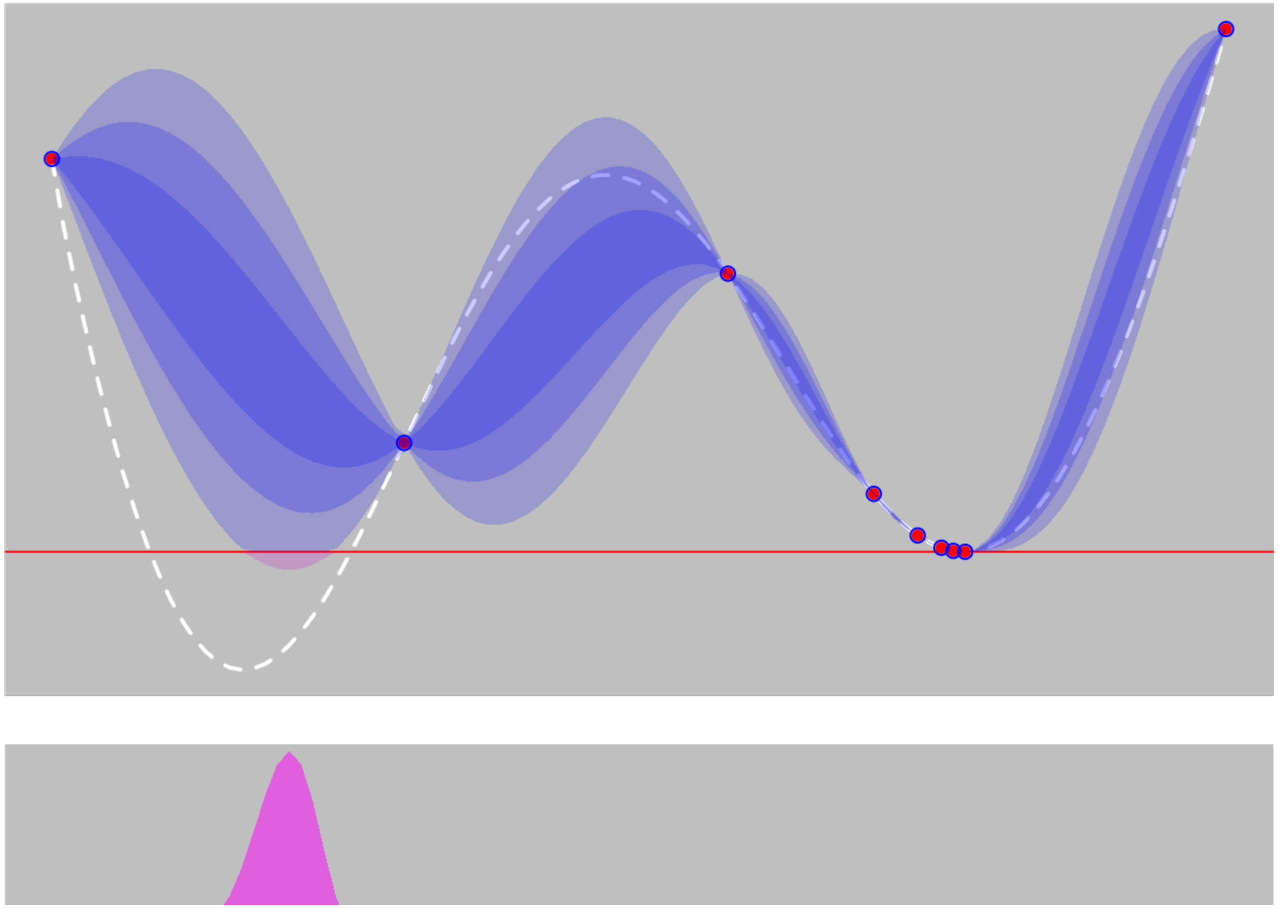
EGO (reminder)



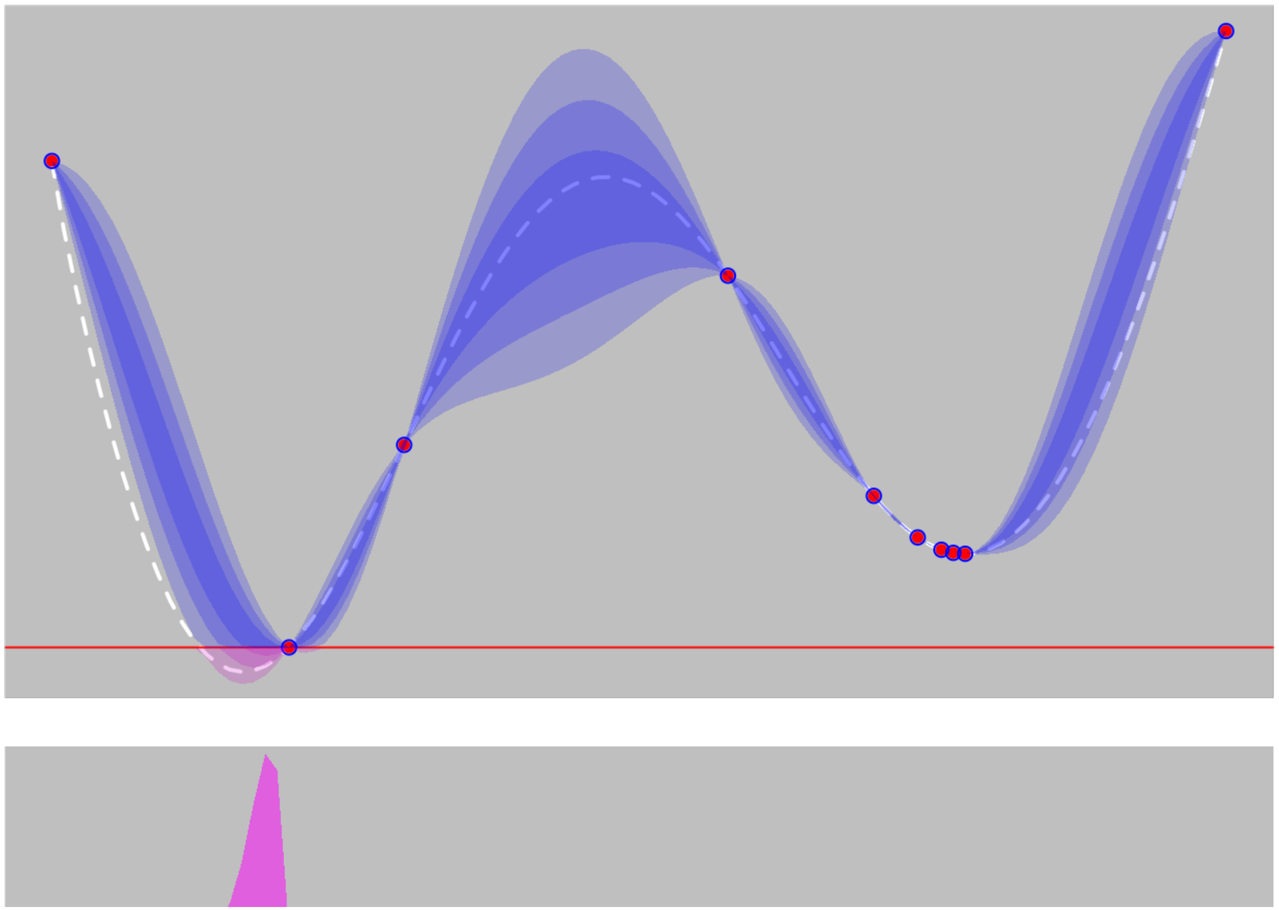
EGO (reminder)



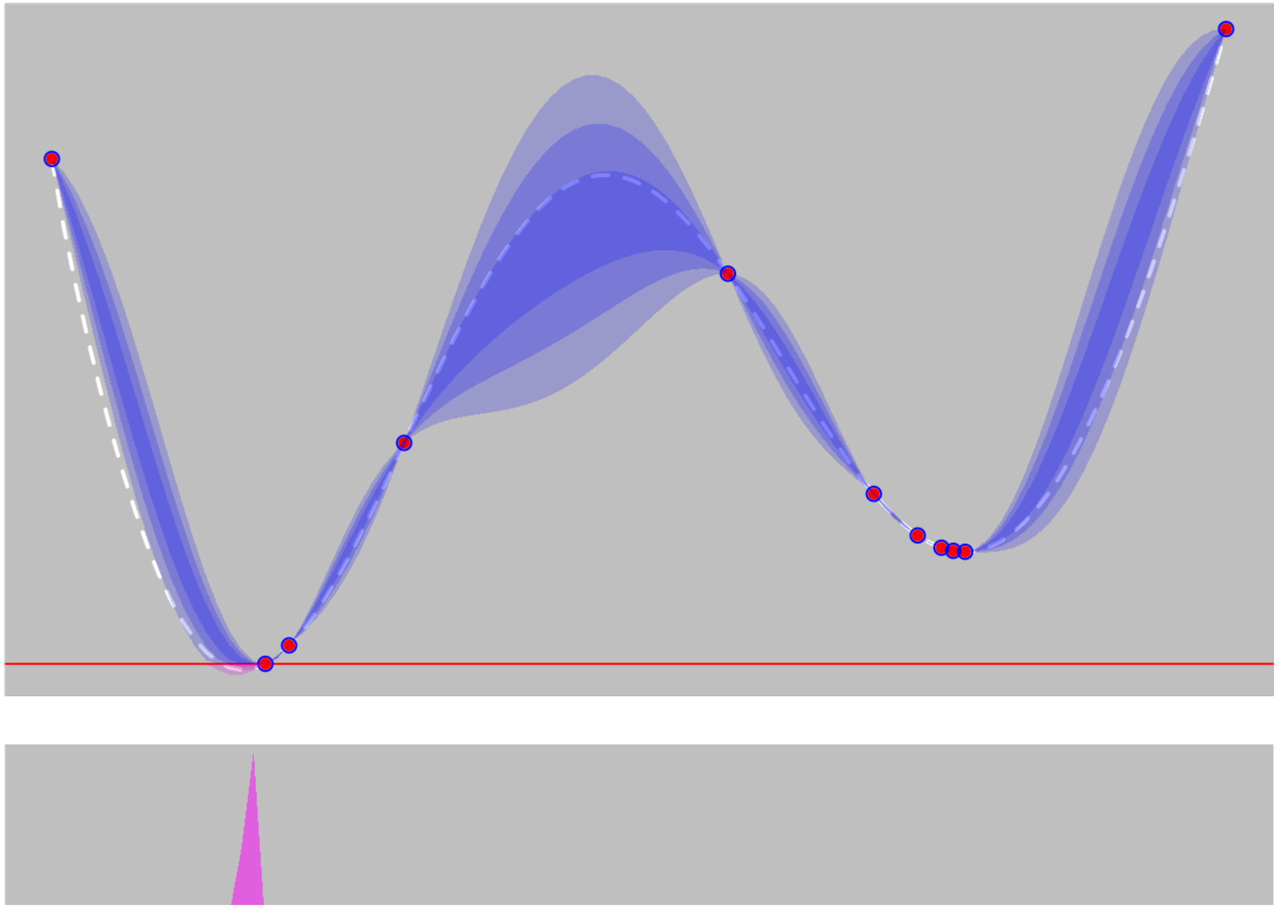
EGO (reminder)



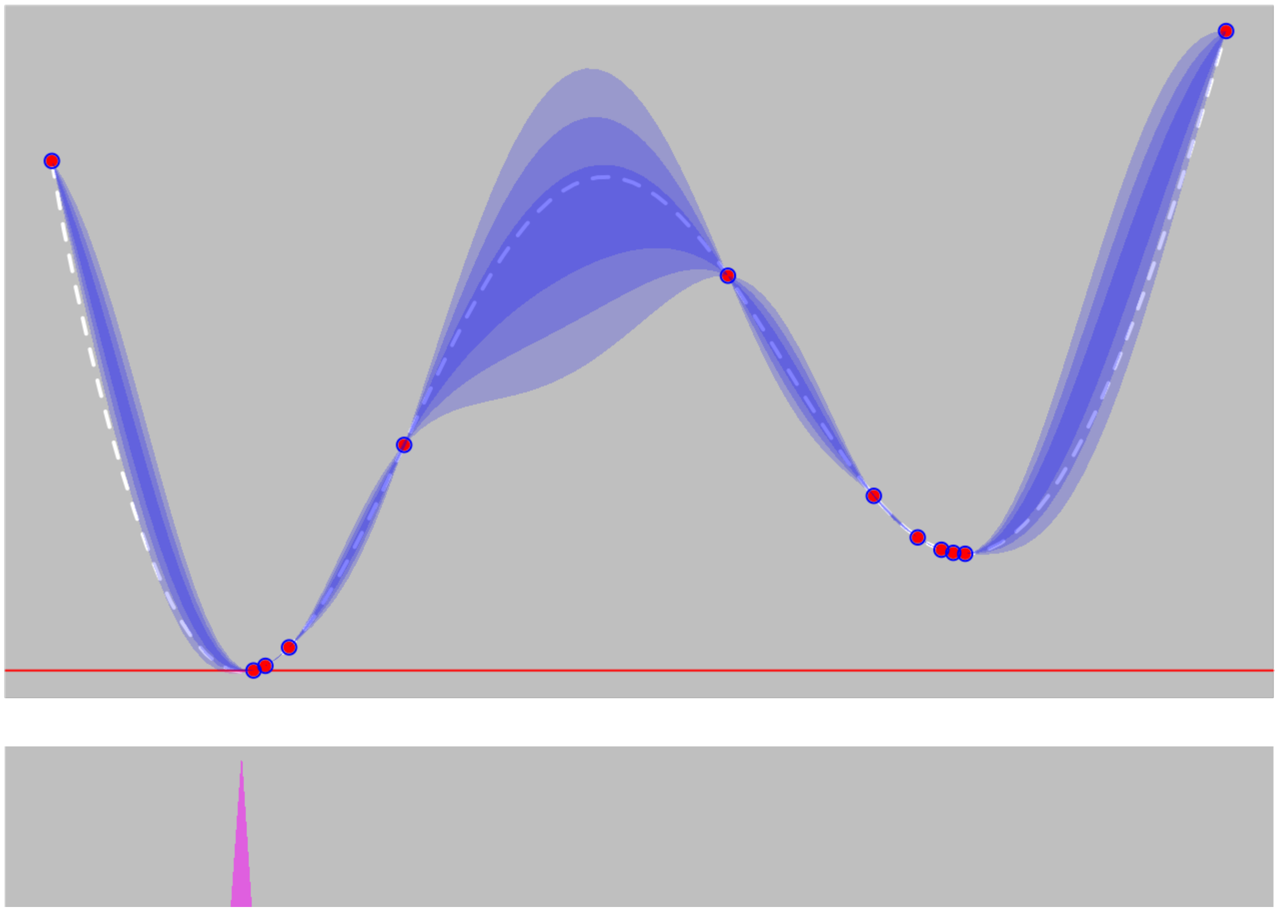
EGO (reminder)



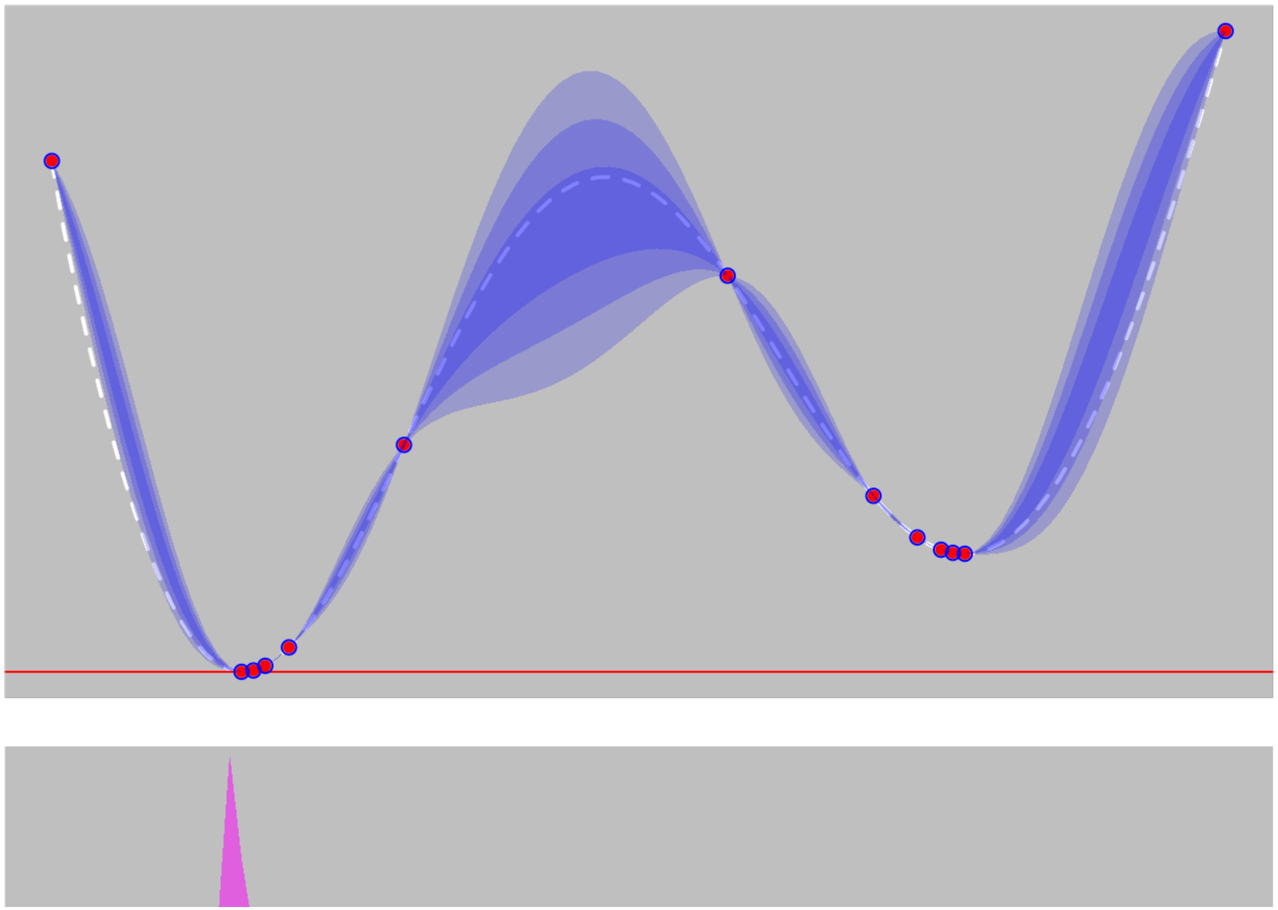
EGO (reminder)



EGO (reminder)



EGO (reminder)



EGO (reminder)

Support for parallel evaluations of f

- Obviously major feature of optimization:
  - make profit of HPC investment
  - even multi-CPU standalone computers
  - non-parallel numerical solvers become relevant (again)

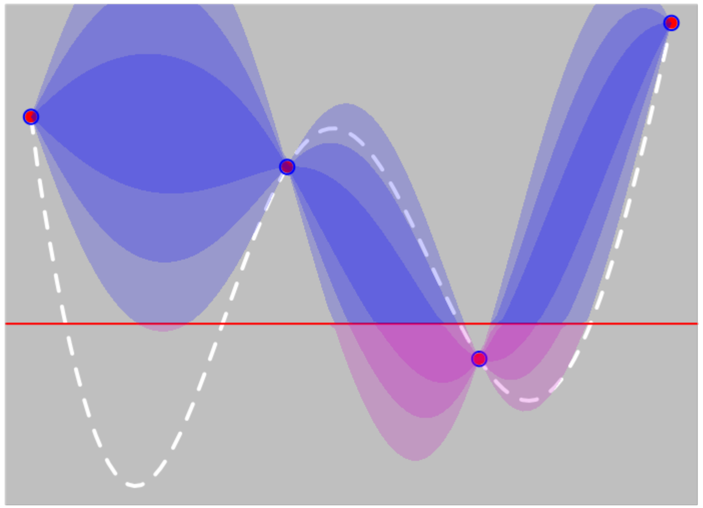
EGO++:

- GP-consistency $q - EI$ criterion (quite hard)
- Batch-sequential heuristic "Constant Liar":
  1. instead of evaluating $f(x^*)$, assume $f(x^*) = min\{f(X)\}$
  2. update (including $(x^*, min\{f(X)\})$ in X) and optimize EI
  3. …

Inversion

$$\color{blue}{M}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .), c(.) = C(x, .)$

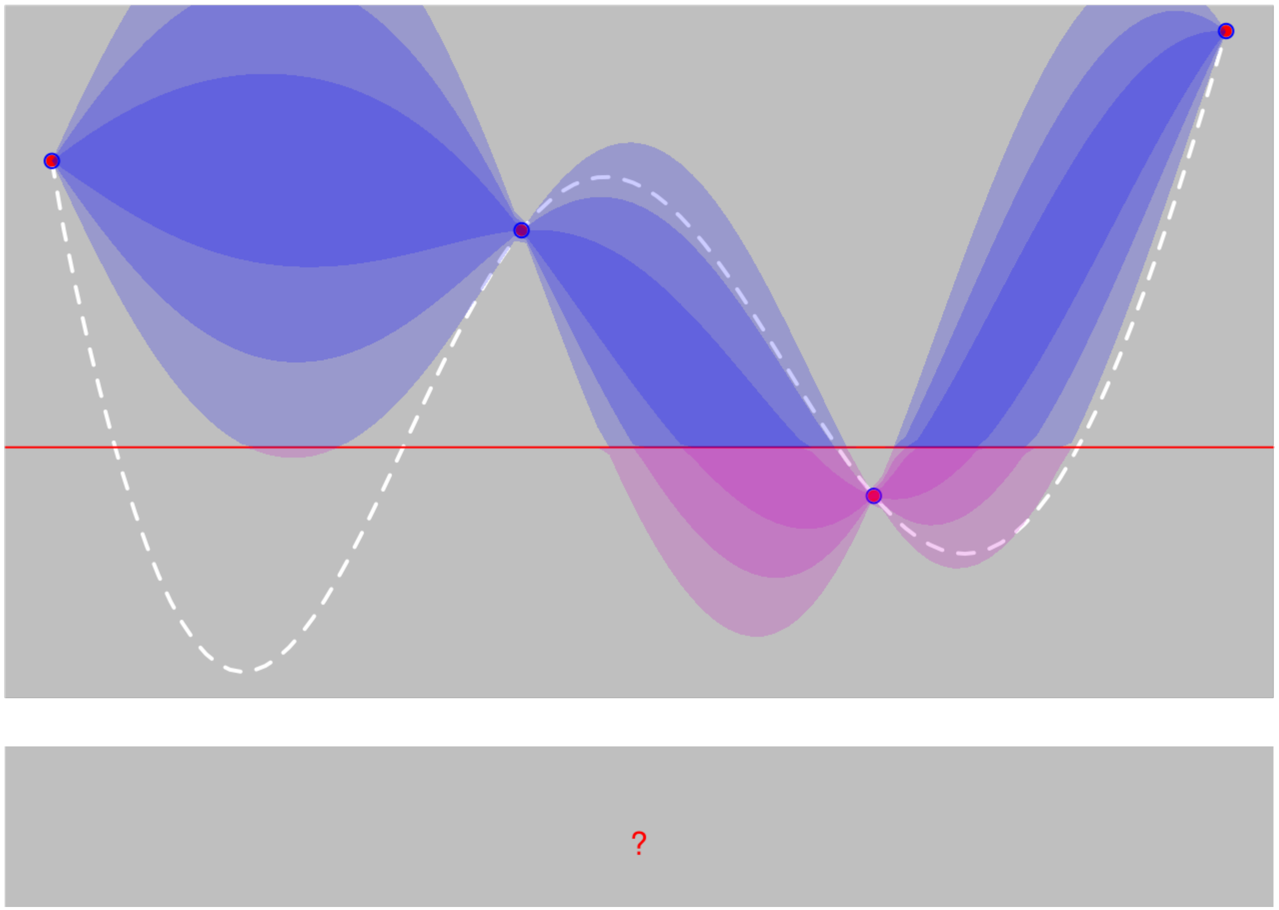


$$\{x^*\} = arg_{x \in S \subset \mathbb{R}^d} \{f(x) < T\}$$

- what [approx] trend(x) will be useful ?
- draw $s(x)$, $m(x)$, $T - m(x)$, …

Build your own inversion criterion …

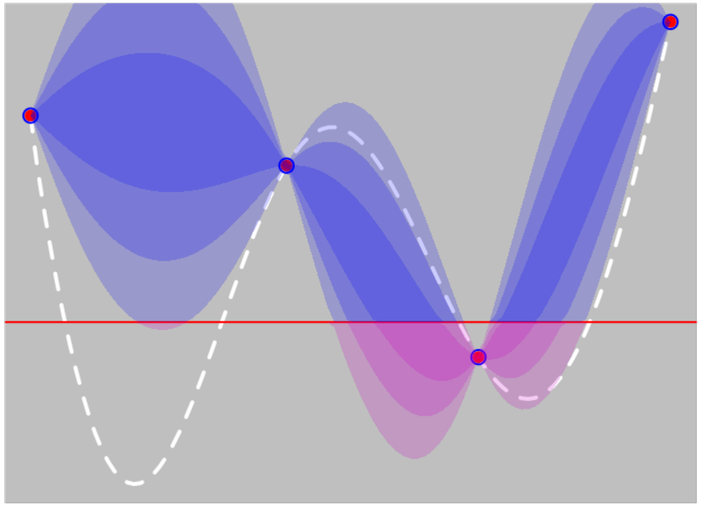
Build your own inversion criterion …



Efficient Global Inversion - Bichon's

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



Let's define the *Expected Excursion*:

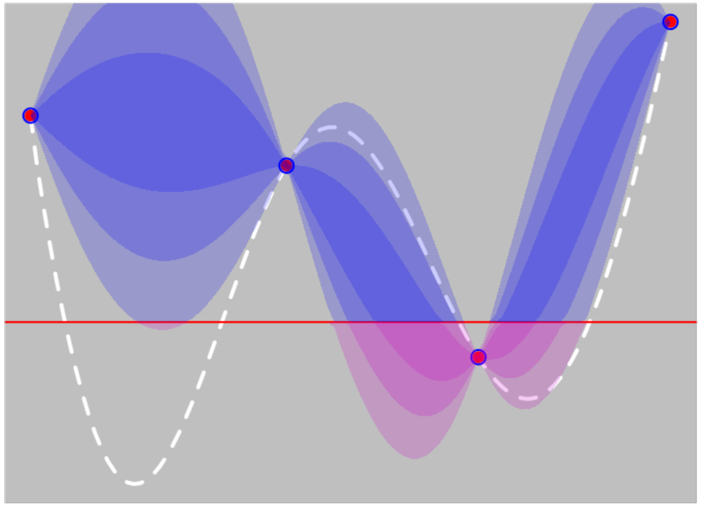
$$EE(x) = E[(s_n(x) - |T - M(x)|)^+]$$

which is (also) analytical thanks to M properties…

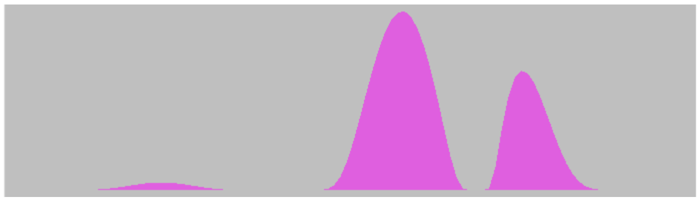
EGI - Bichon's

$$\color{blue}{M}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



$$v(x) = \frac{T - m(x)}{s(x)}$$



$$EE(x) = s(x) \Big(\; p_{\mathcal{N}}(v(x) + 1) - p_{\mathcal{N}}(v(x) - 1)$$

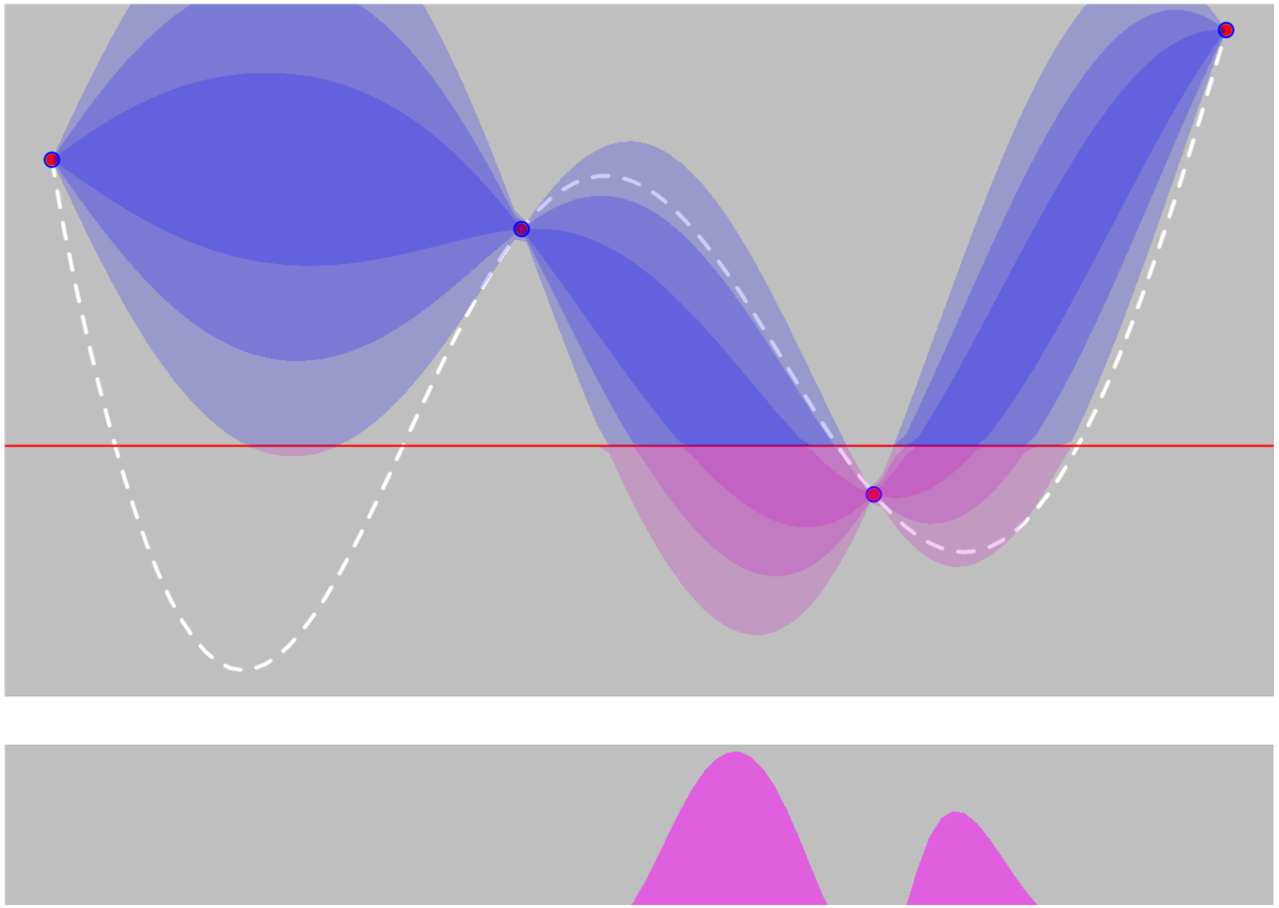
EGI - Bichon's

"Sequential design of computer experiments for the estimation of a probability of failure" - Bect, Ginsbourger, Li, Picheny, Vazquez, (Statistics and Computing 2012)

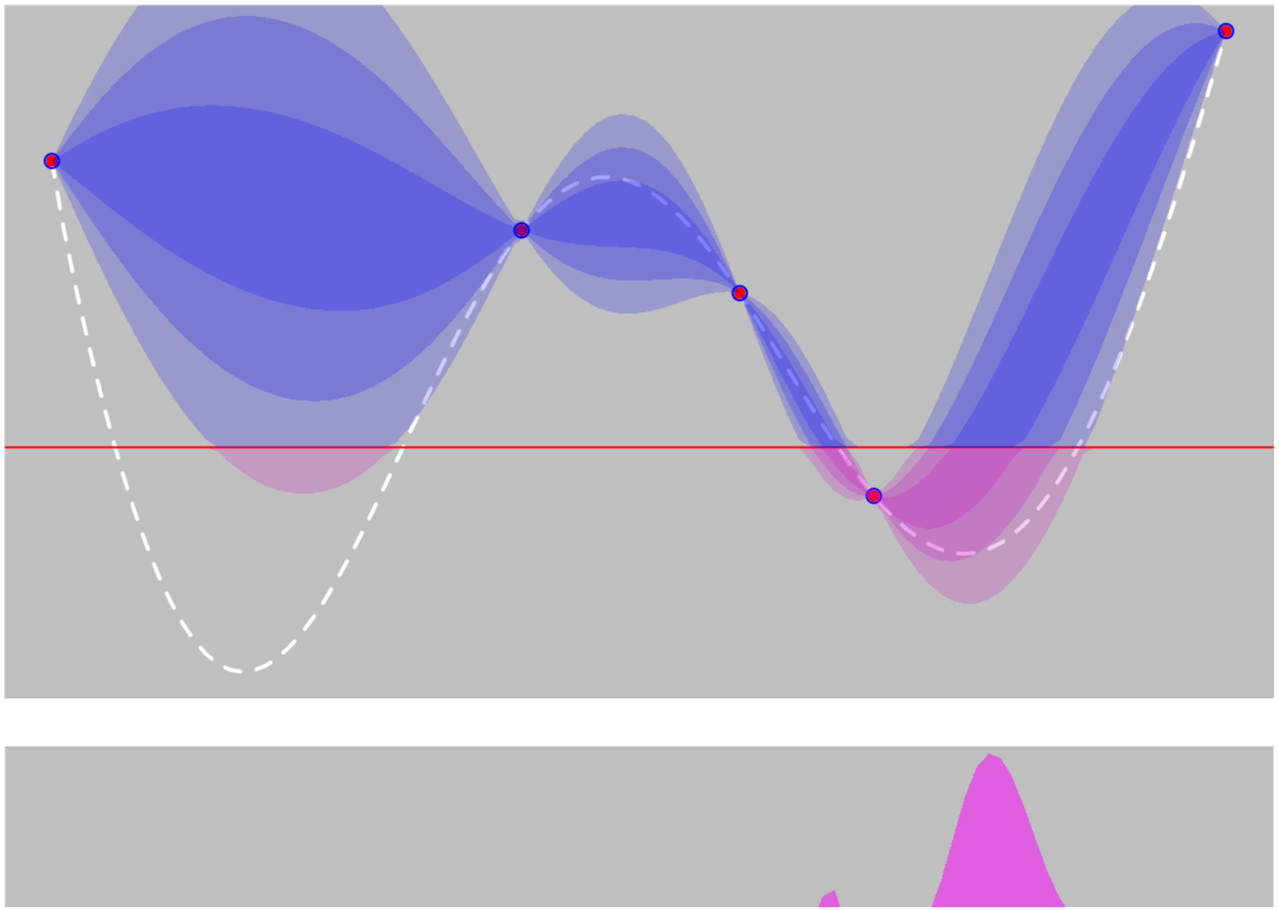
EGI:
Maximize $EE(x)$ (*), compute f there, add to X.
Repeat until …

- **+** trade-off between exploration and exploitation
- **+** requires few evaluations of f
- **-** often lead to add close points to each others …
  Which is not very comfortable for kriging numerical stability
- **-** "one step lookahead" (myopic) strategy
- **-** rely on model suitability to f

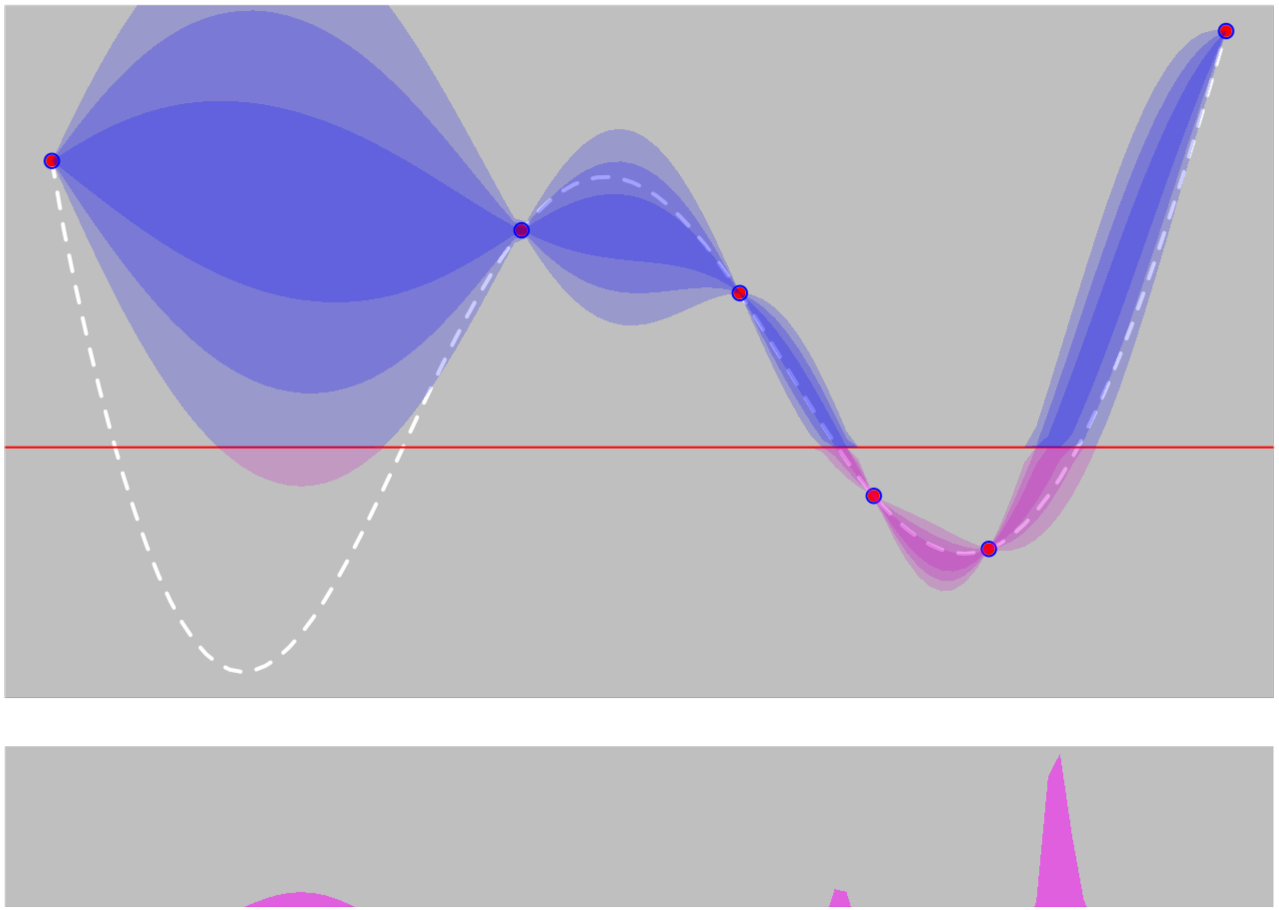
EGI - Bichon's



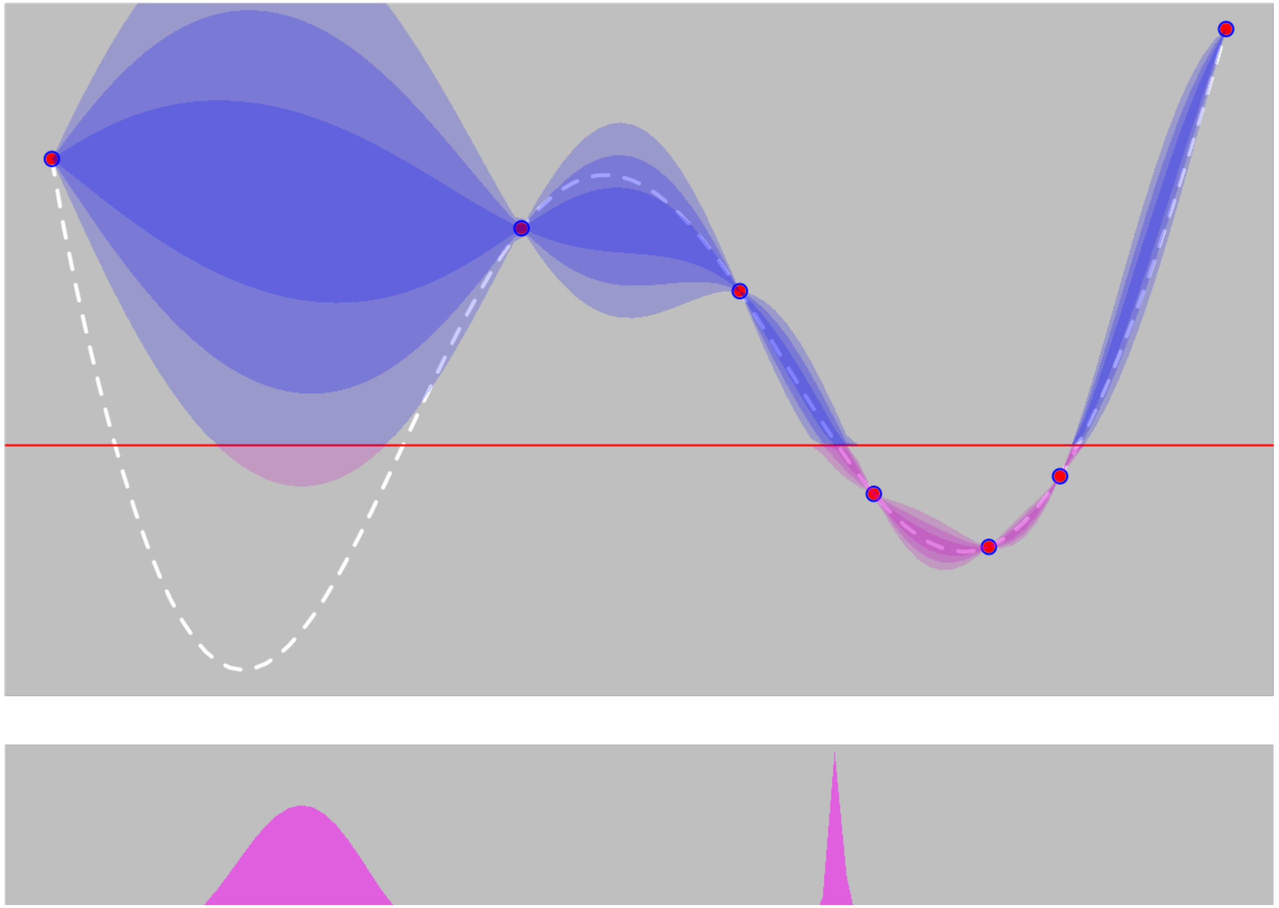
EGI - Bichon's



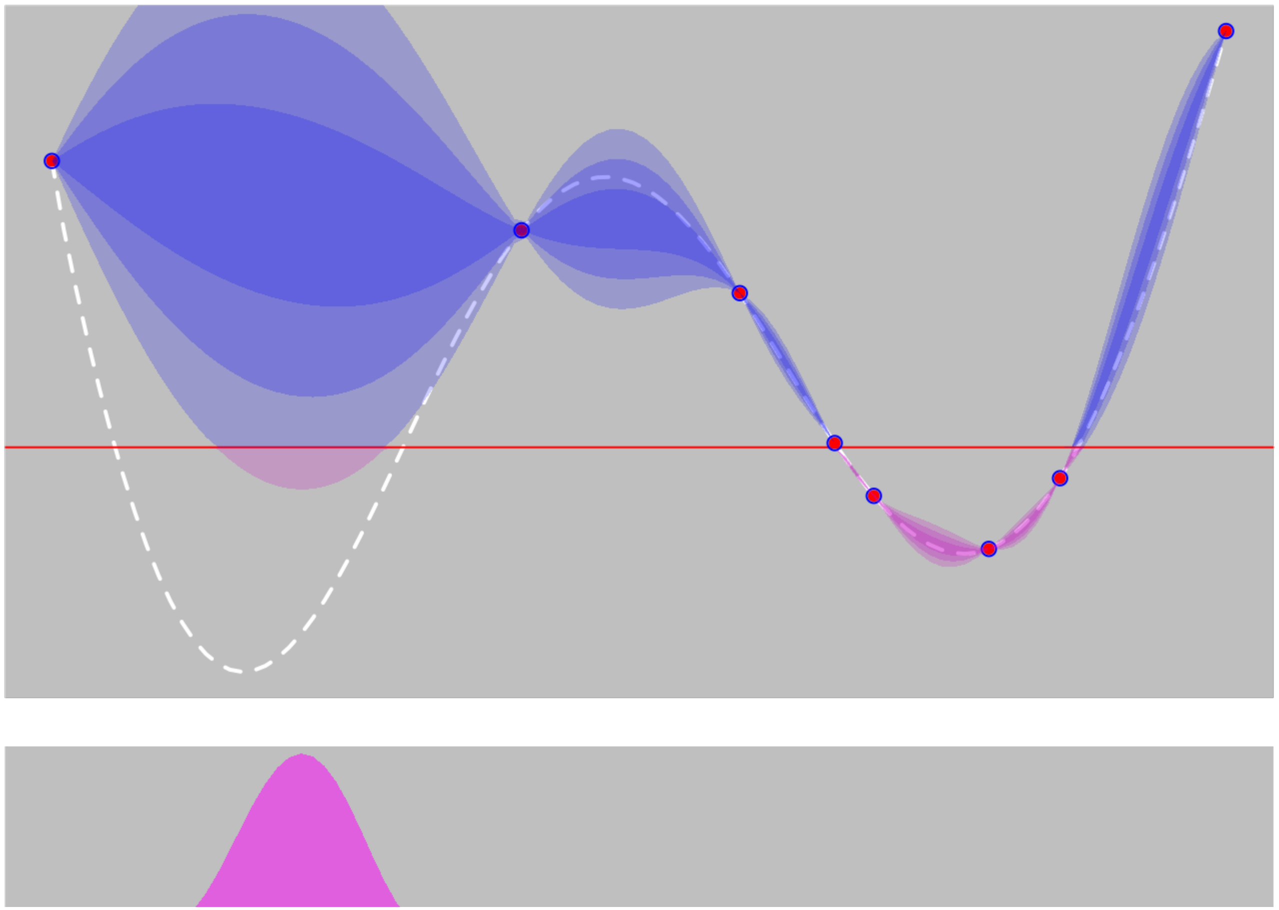
EGI - Bichon's



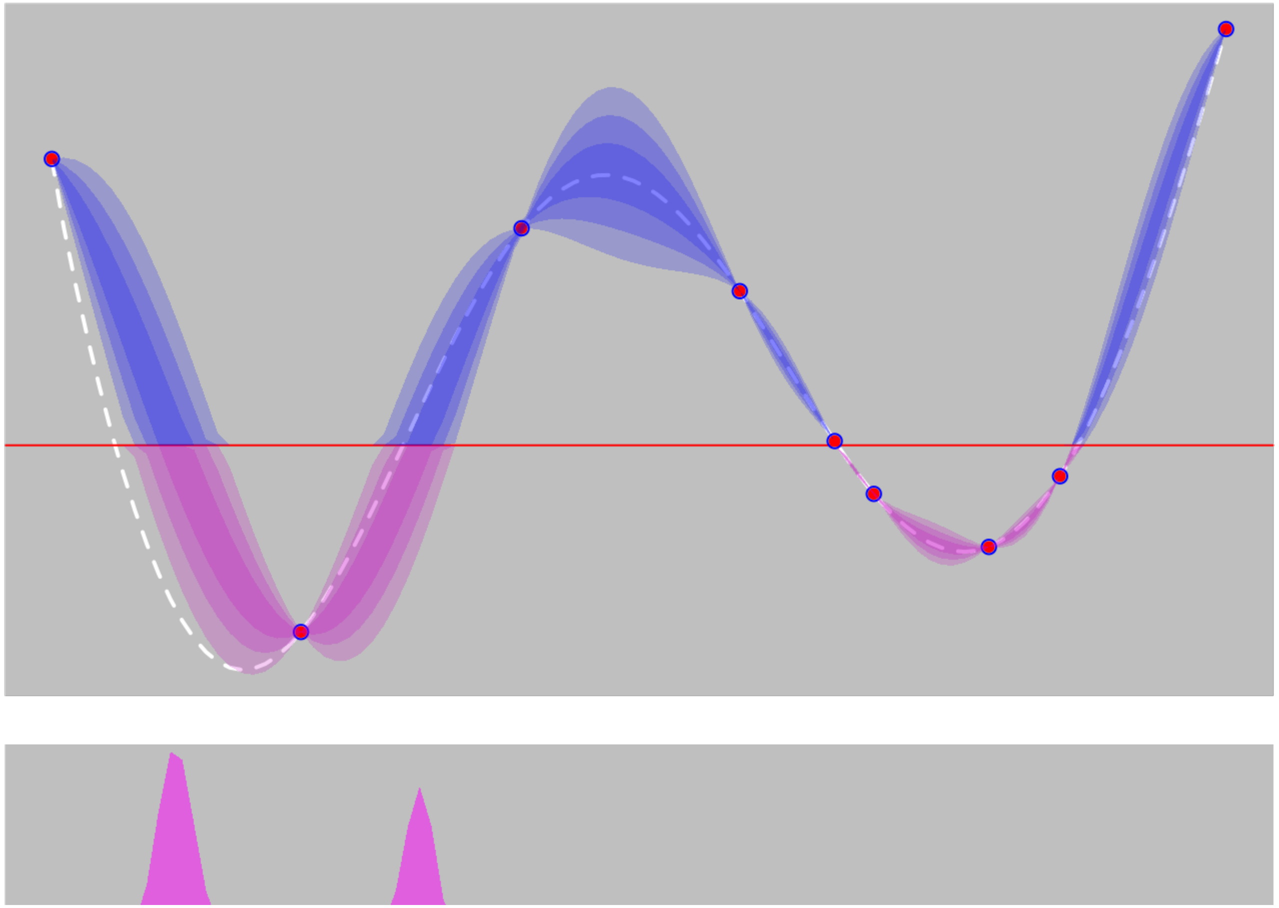
EGI - Bichon's



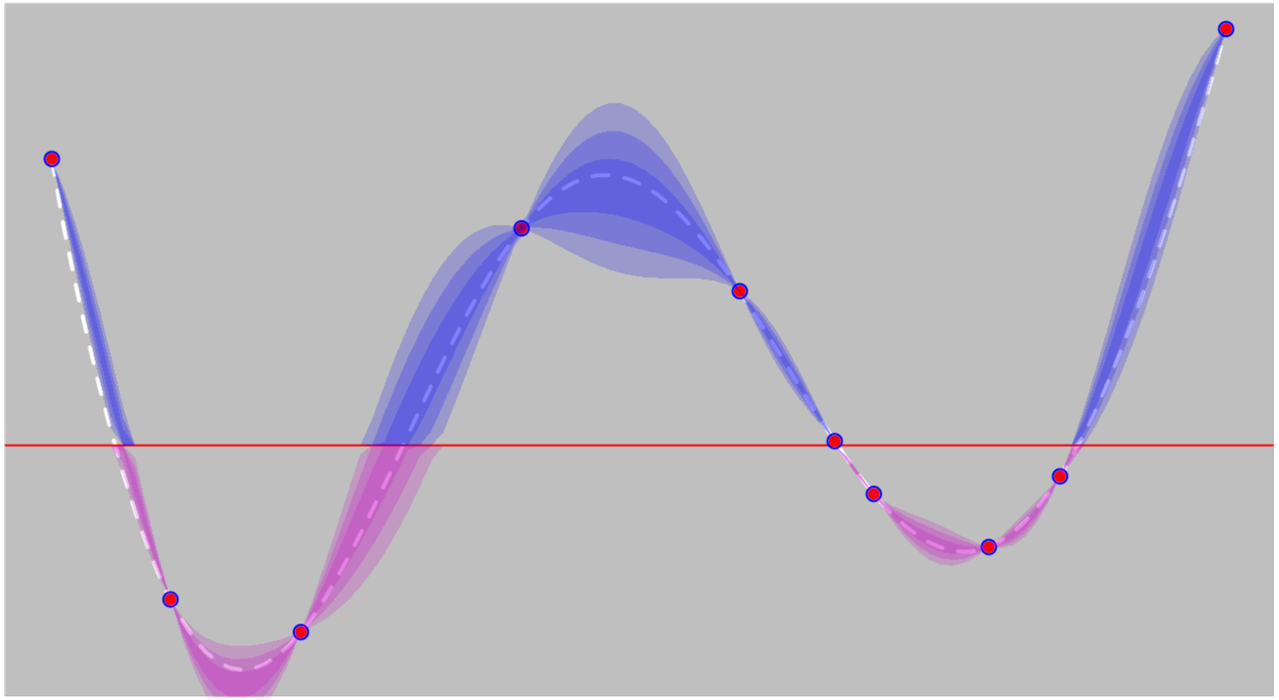
EGI - Bichon's



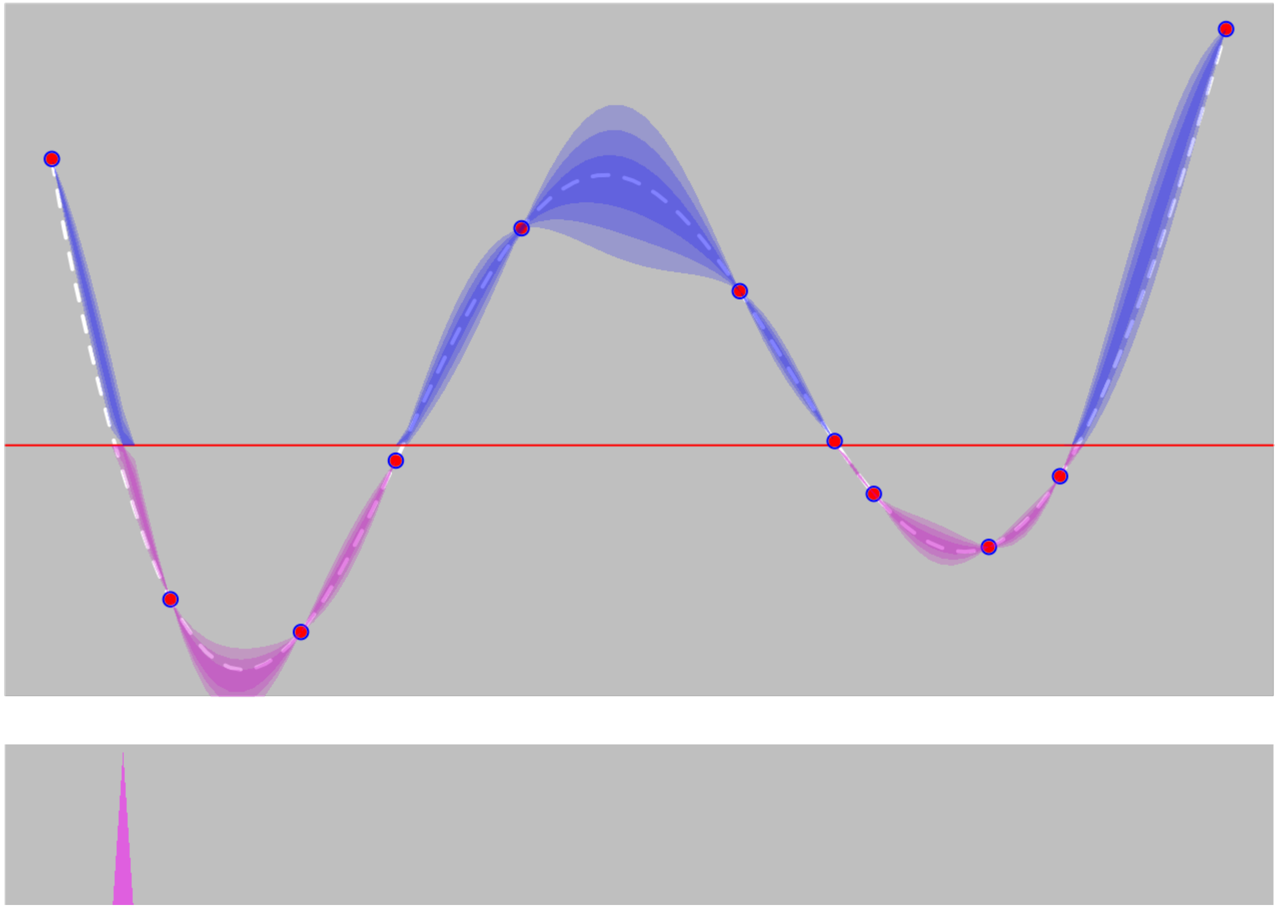
EGI - Bichon's



EGI - Bichon's



EGI - Bichon's



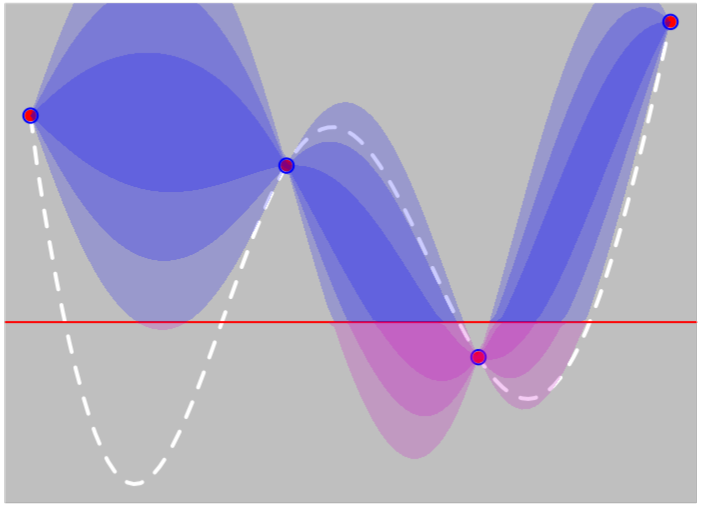
- define a suitable criterion for engineering target
- compute its integral *conditionally* to any x added
- find the minimizer of x
- sample f at this new point

Model *conditional uncertainty* mining

Inversion

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$

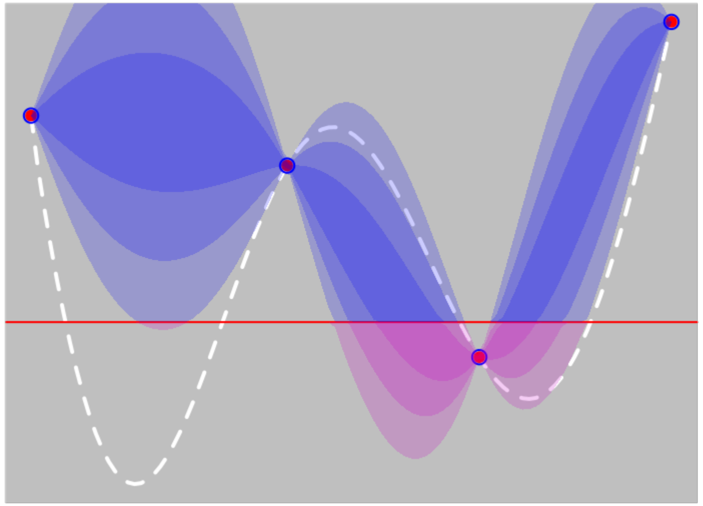


Exploration >> exploitation, which is interesting for identification problems, where discrete solution never apply (like inversion).

Stepwise Uncertainty Reduction

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



Let's define the *Probability of Excursion*:

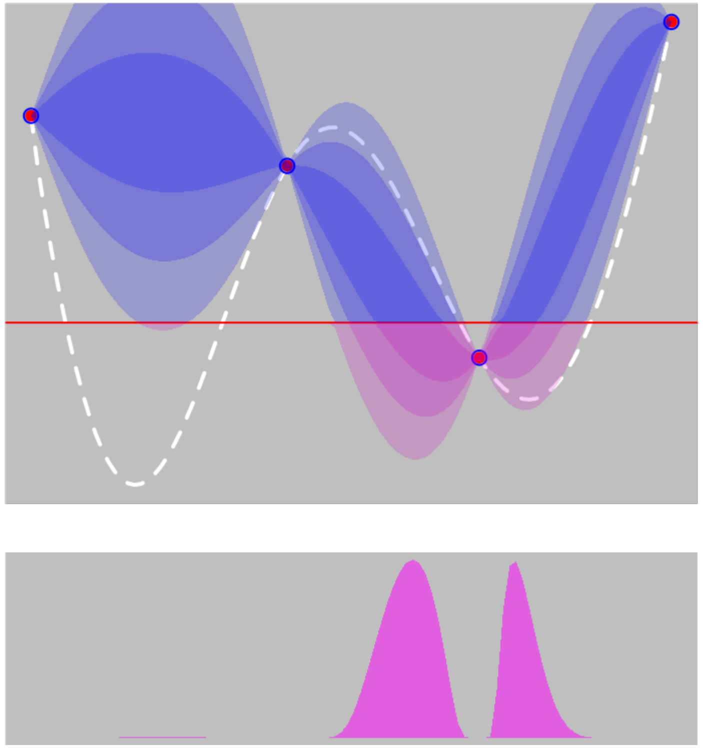
$$PE(x) = P[M(x) < T]$$

… and the *Uncertainty of Excursion*:

SUR

$$\color{blue}{M}\color{black}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .), c(.) = C(x, .)$



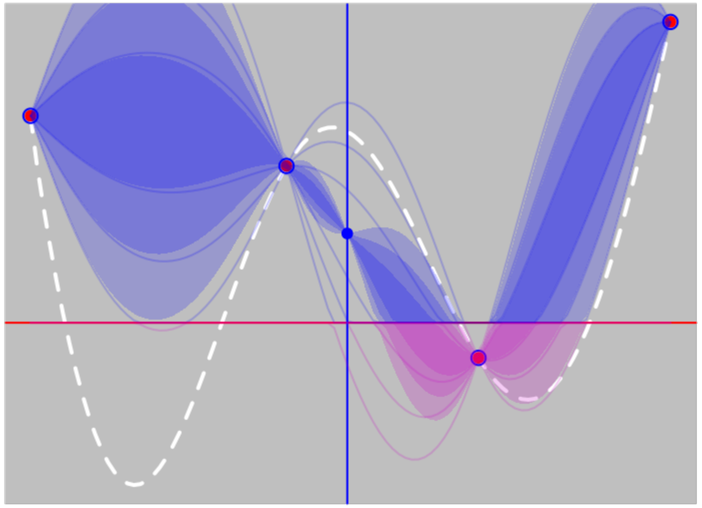
$$v(x) = \frac{T - m(x)}{s(x)}$$

$$SE(x) = p_{\mathcal{N}}\left(v(x)\right) \times (1 - p_{\mathcal{N}}\left(v(x)\right))$$

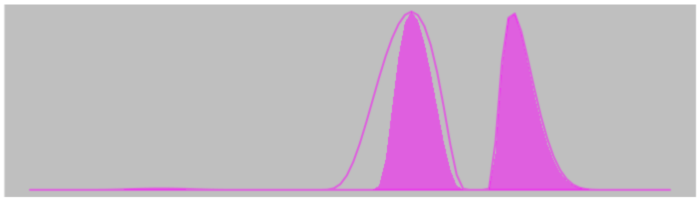
SUR

$$\color{blue}{M}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



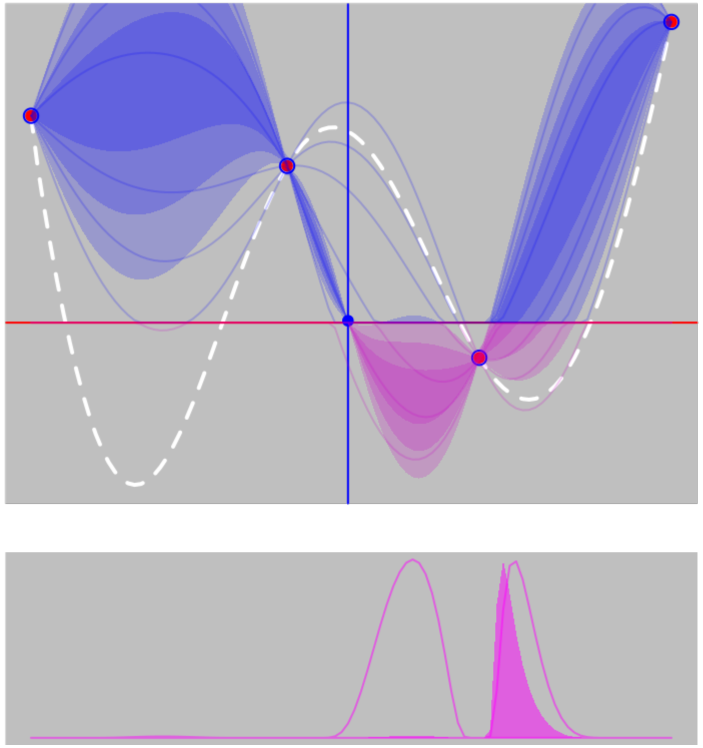
$$y_{new} \sim \mathcal{N}(m(x_{new}), s^2(x_{new}))$$



SUR

$$\color{blue}{M}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .), c(.) = C(x, .)$

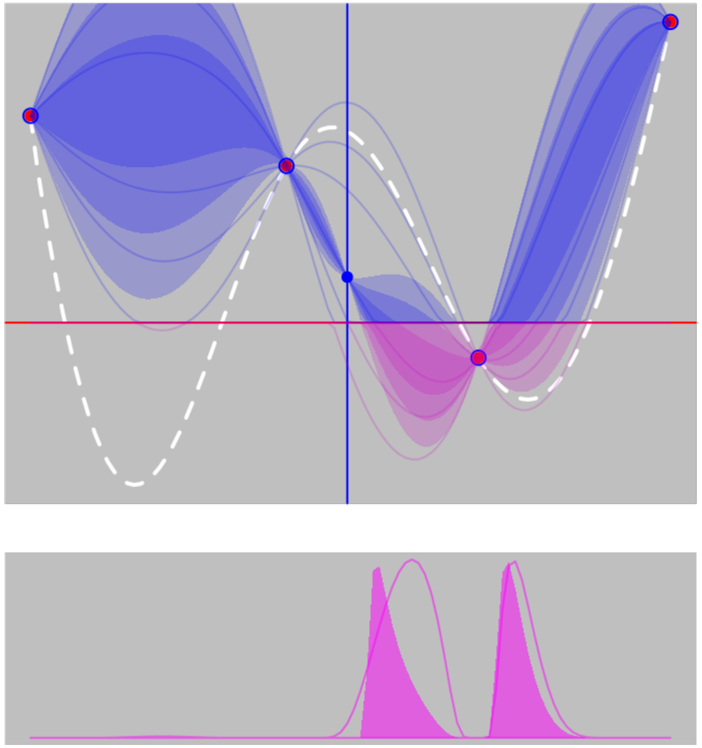


$$y_{new} \sim \mathcal{N}(m(x_{new}), s^2(x_{new}))$$

SUR

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$

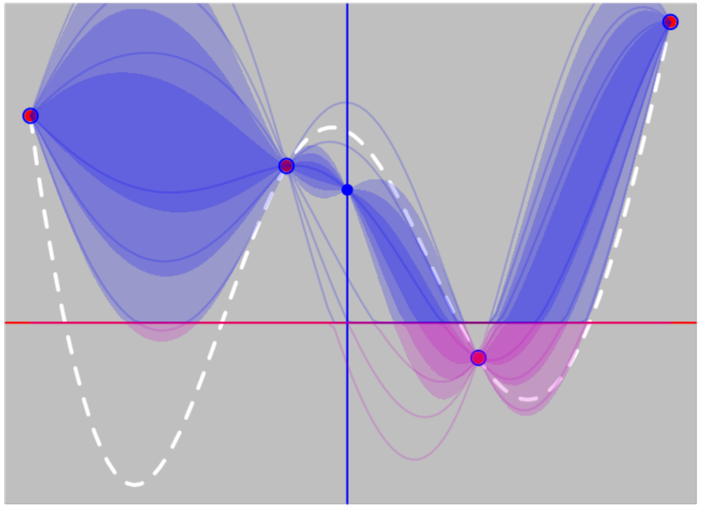


$$y_{new} \sim \mathcal{N}(m(x_{new}), s^2(x_{new}))$$

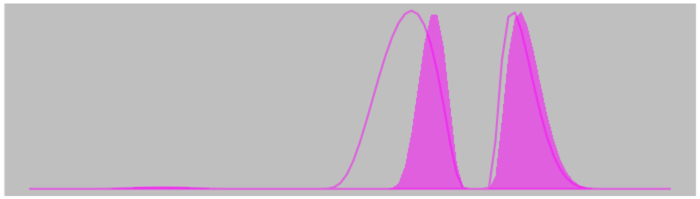
SUR

$$M(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



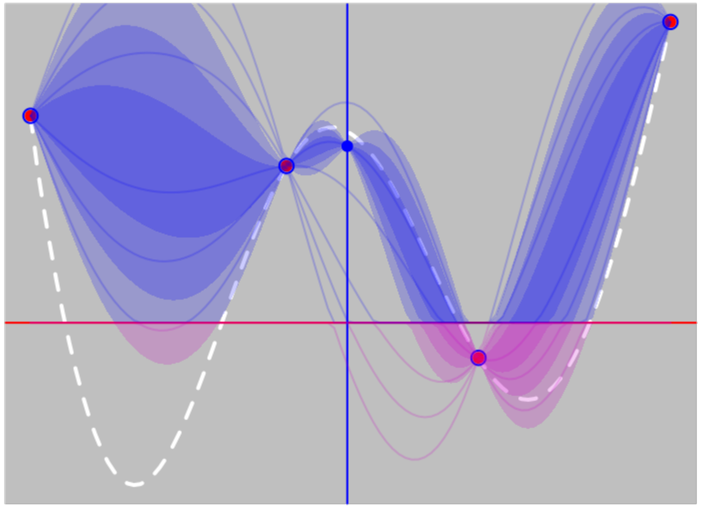
$$y_{new} \sim \mathcal{N}(m(x_{new}), s^2(x_{new}))$$



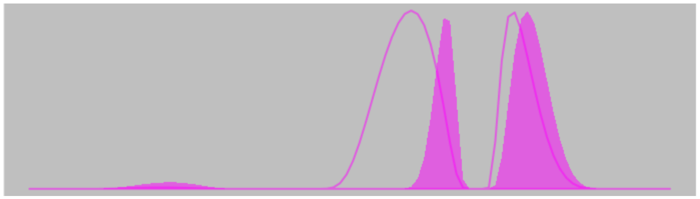
SUR

$$\color{blue}{M}\color{black}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .), c(.) = C(x, .)$



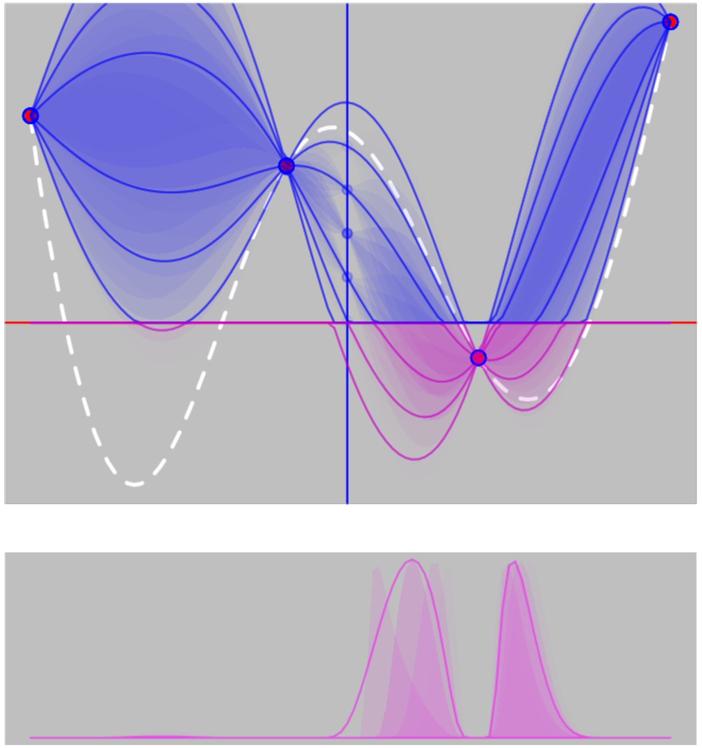
$$y_{new} \sim \mathcal{N}(m(x_{new}), s^2(x_{new}))$$



SUR

$$\color{blue}{M}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$

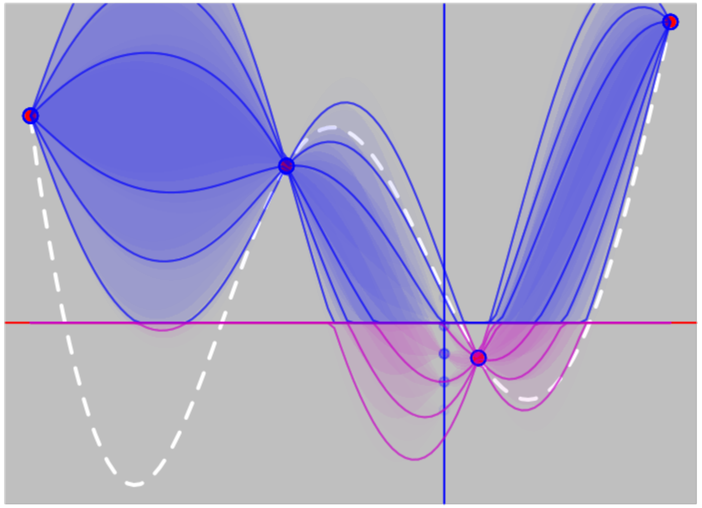


$$y_{new} \sim \mathcal{N}(m(x_{new}), s^2(x_{new}))$$

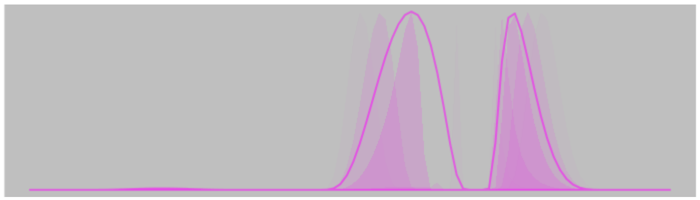
SUR

$$\color{blue}{M}\color{black}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



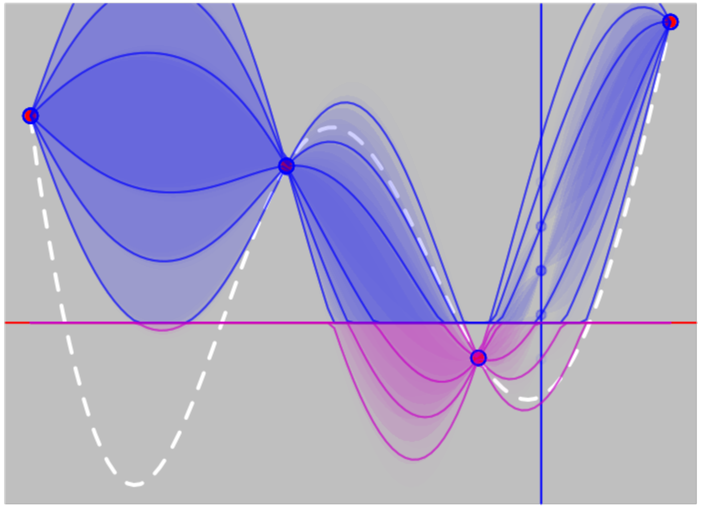
$$y_{new} \sim \mathcal{N}(m(x_{new}), s^2(x_{new}))$$



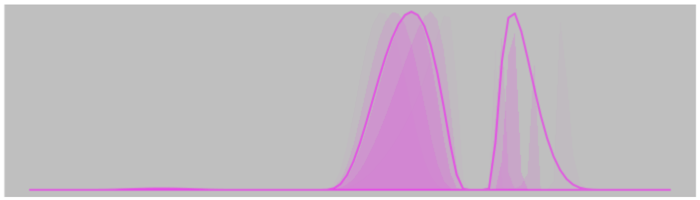
SUR

$$\color{blue}{M}(x) = \mathcal{N}(m(x), s^2(x))$$

- $m(x) = C(x)^T C(X)^{-1} f(X)$
- $s^2(x) = c(x) - C(x)^T C(X)^{-1} C(x)$
- C is the covariance kernel
  $C(.) = C(X, .)$, $c(.) = C(x, .)$



$$y_{new} \sim \mathcal{N}(m(x_{new}), s^2(x_{new}))$$



Reference

KrigInv: An efficient and user-friendly implementation of batch-sequential inversion strategies based on Kriging

- Clément Chevalier, IMSV University of Bern
- Victor Picheny, INRA Toulouse
- David Ginsbourger, IMSV University of Bern

Implements

- Pointwise criterions: Bichon, Ranjan, TMSE
- SUR criterions: TIMSE, SUR, Jn

Conclusions

Pointwise vs. conditional uncertainty

Instead of a *local* criterion (like $SE(x)$),
use a *global* gain which integrates *local* criterion:

$$SE(x) = P[T < M_n(x)] \times P[T > M_n(x)]$$

$$\downarrow$$

$$SUR(x) = \int_S P[T < M_{n+\{x\}}(y)] \times P[T > M_{n+\{x\}}(y)]\, dy$$

i.e. We search for x which, once added to X, most reduces $\int_S SE$.

Pointwise vs. conditional uncertainty

Instead of a *local* criterion (like $EI(x)$),
use a *global* gain which integrates *local* criterion:

$$EI(x) = E[min\{f(X_n)\} - M_n(x)]^+$$
$$\downarrow$$
$$IECI(x) = \int_S E[min\{f(X_n \oplus x)\} - M_{n+\{x\}}(y)]^+ \, dy$$

i.e. We search for x which, once added to X, most reduces $\int_S EI$.

Stepwise Uncertainty Reduction

SUR criterions are:

- more flexible than *local* ones (EI) to handle:
  - constraints:

$$min_{x \in S \subset \mathbb{R}^d} \; \{f(x) : g(x) \leq 0\}$$

  - robustness:

$$min_{x \in S \subset \mathbb{R}^d} \; \{f(x) : g(x, u) \leq 0, \forall u \in U\}$$

- but **much** more costly to evaluate ($\int_S$)

Constrained optimization

Basically using EI:

$$EI(x) = \mathbf{1}_{g(x) \leq 0} \; E[min\{f(X)\} - M(x)]^+$$

… but this excludes to sample x if $g(x) > 0$.

Or using $IECI$:

$$IECI(x) = \int_S \mathbf{1}_{g(y) \leq 0} \; E[min\{f(X_n \oplus x)\} - M_{n+\{x\}}(y)]^+ \; dy$$

Tips & tricks

On-shelf available (tech.)

General optimization

Languages:

- R, Python, Matlab/Octave
- C++, Java, C, Fortran

Libraries:

- **NLOpt** (C, C++, Fortran, Matlab/Octave, Python, Julia, R)
  DiRect, CRS, COByLA, Nelder-Mead, BFGS, …
- SciPy.optimize (Python) BFGS, COByLA, Nelder-Mead, …
- optim (R) Nelder-Mead, BFGS, Simulated-Annealing, …

State-of-the-Art algorithms: EGO, CMA-ES, NSGA2, PSO
(C, C++, Fortran, Matlab/Octave, Python, R, Scilab, Julia, …)

On-shelf available (tech.)

Languages:

- R, Python, Matlab/Octave
- C++, Java,

Libraries:

- **DiceKriging**/KerGP (R)
- SKL (Matlab/Octave)
- geoR (R)
- GPy/GPflow (Python)
- OpenTURNS (Python)

If problem remains *too* hard…

Go deeper in f.

1. Get more runs
    - numerical model: more CPU, HPC, cloud
2. Get more prior informations
    - boundary conditions
    - trending hypothesis
    - simplified model (multi-level opt.)
3. Get derivatives
    - physical model: intrinsic physics
    - numerical model: integrated|auto differentiation
        - source code differentiation