Conception de systèmes à partir d'expériences numériques coûteuses

(mots-clés : planification d'expériences, modélisation de systèmes par processus aléatoires gaussiens, optimisation)

Emmanuel Vazquez

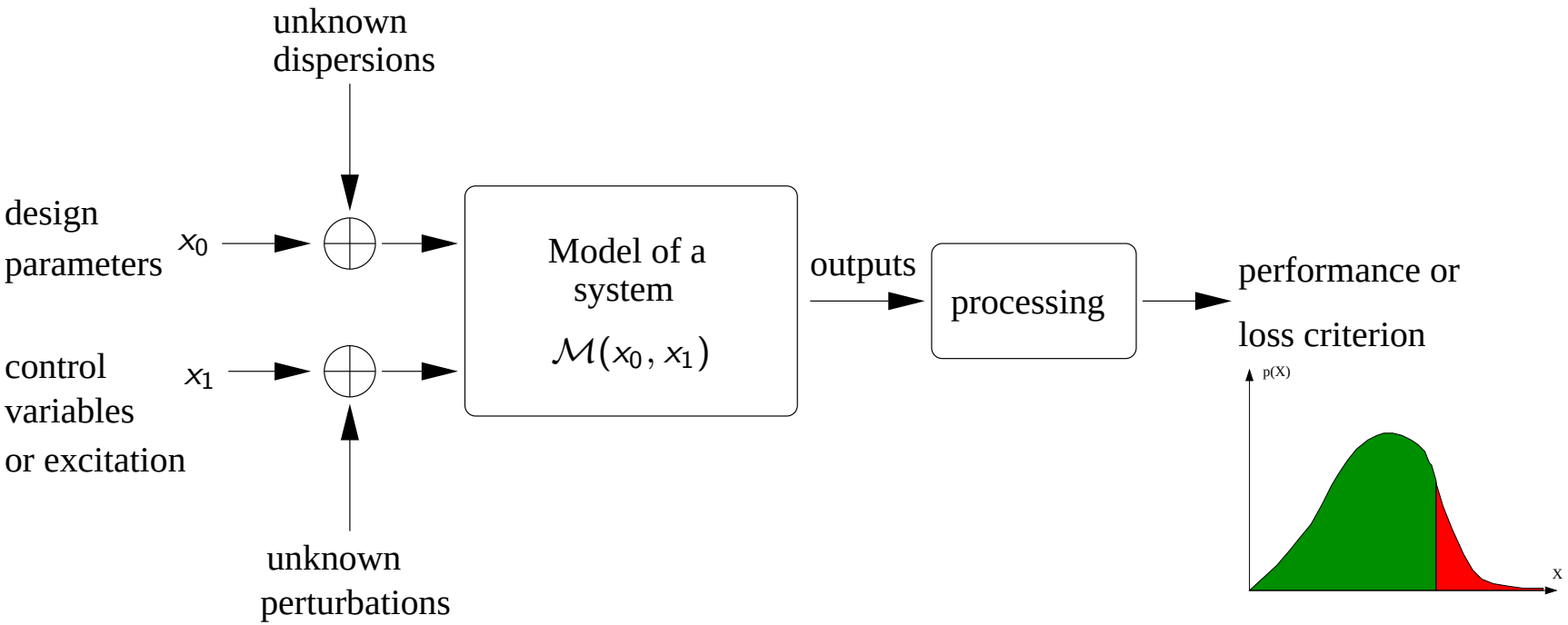
SUPELEC, Gif-sur-Yvette, France

Journée Calcul et Simulation
4 juin 2014

Outline

1. Computer models in engineering

Context overview



Model implemented under the form of a computer program
(e.g., a finite element model).
A single run of the program may be time- and resource-consuming.

Computer models in engineering

- $\mathbb{X} \subseteq \mathbb{R}^d \quad \rightsquigarrow$ factor/parameter space of the system
- $f : \mathbb{X} \to \mathbb{R} \rightsquigarrow$ performance or cost function (function of the outputs of the system)

- Main classes of problems:
  1. Approximation of the performance of a system, from expensive evaluations $x_i \mapsto f(x_i)$, $1 \leq i \leq N$, on a domain of interest
  2. Optimization of the performance of a system, cost minimization...

$$x^\star = \underset{x \in \mathbb{X}}{\operatorname{argmax}} f(x) \qquad \text{or} \qquad x^\star = \underset{x \in \mathbb{X}}{\operatorname{argmin}} f(x)$$

  3. In presence of uncertain factors, estimation of a probability of failure

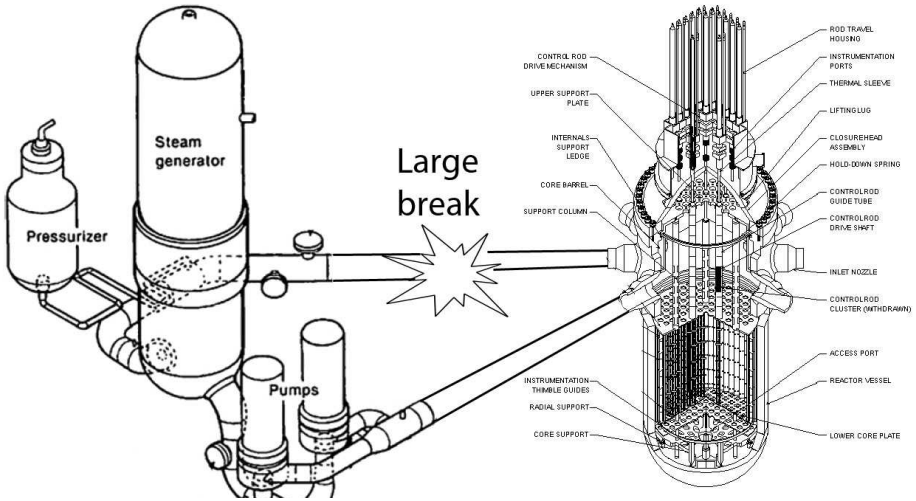
$$\alpha^u(x) \quad := \quad \mathrm{P}_{\mathbb{X}}\{x \in \mathbb{X} : f(x) > u\}$$

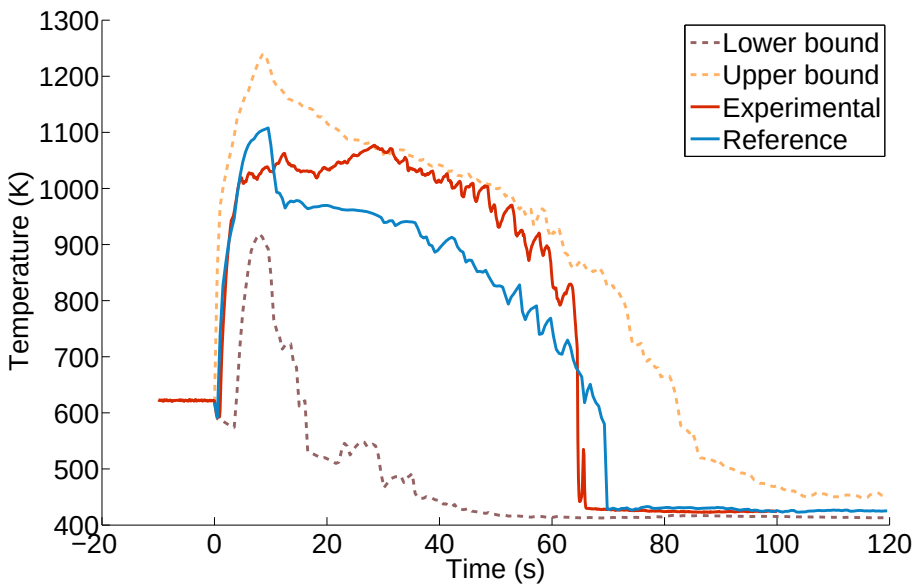
  where $\mathrm{P}_{\mathbb{X}}$ is some probability distribution on $(\mathbb{X}, \mathcal{B}(\mathbb{X}))$

NB: this is a simplified view $\to$ most real problems have several performance functions, and mix different objectives

Example 1/2 – Risk analysis

- Computer simulations to assess the probability of undesirable events





(Courtesy of CEA)

- A serious accident: loss of coolant in a pressurized water nuclear reactor
- Under these conditions, temperature of fuel rods can be described by $\sim$ 50 dimensioning factors, which are not known accurately
- Peak temperature can be estimated using complex and time-consuming simulations
- $f : \mathbb{X} \to \mathbb{R}$ peak temp. as a function of dim. factors
- Objective: estimate a probability of exceeding a critical value

$$\alpha = \mathrm{P}_{\mathbb{X}}\{f \geq u\}$$

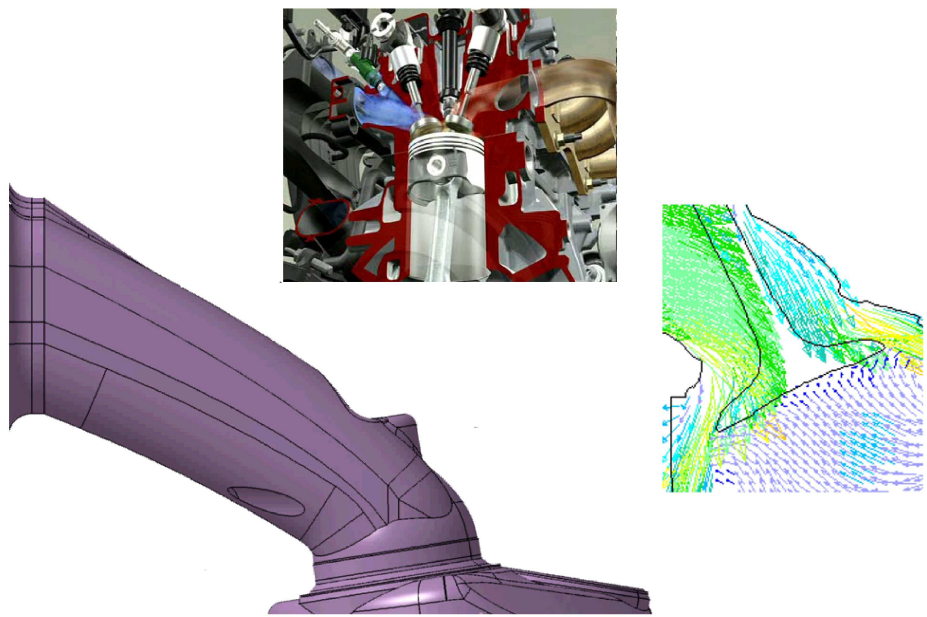
or a worst-case

$$M = \sup_{x \in \mathbb{X}} f(x)$$

Example 2/2 – Design optimization

- Computer simulations to design a product or a process, in particular
  - to find the best feasible values for design parameters (optimization problem)
  - to minimize the probability of failure of a product

- To comply with European emissions standards, the design parameters of combustion engines have to be carefully optimized
- The shape of intake ports controls airflow characteristics, which have direct impact on
  - the performances of the engine
  - emissions of NO_x and CO
- $f : \mathbb{X} \subset \mathbb{R}^d \to \mathbb{R}$ performance as a function of design parameters ($d = 20 \sim 100$)
- Computing $f(x)$ is time-consuming
- Objective: estimate $x^\star = \mathrm{argmax}_x\, f(x)$, or $x^\star = \mathrm{argmax}_x\, f(x)$ subject to $\mathrm{P}\{\text{pollutant emissions} \leq \text{threshold}\} > \gamma$



Simulation of an intake port (Navier-Stokes equations) (courtesy of Renault, Julien Villemonteix)

Distinct properties of computer experiments

- The performance/cost function $f : \mathbb{X} \subseteq \mathbb{R}^d \to \mathbb{R}$ is only known through pointwise evaluations
- An evaluation of f is called a computer experiment. It consists in
  - choosing an $x \in \mathbb{X}$
  - running one or several deterministic computer programs to obtain the value $f(x)$
- ∇f may also be known in rare cases
- The factor space $\mathbb{X}$ may be high-dimensional (typically $10 \sim 100$)
- Evaluation of f may be expensive (e.g., several hours) ➠ budget of experiments is limited (typically < 1000)

2. How to construct a good estimation procedure?

Estimation from computer experiments

- Let $f : \mathbb{X} \to \mathbb{R}$ be a continuous function defined on a compact domain $\mathbb{X}$ with non-empty interior

  (f corresponds to a computer program whose output is not a closed-form expression of the inputs.)

- Objective: from a set of computer experiments, obtain an approximation of

$$f : \mathbb{X} \to \mathbb{R}$$

$$\text{or} \quad m(f) = \min_{x \in \mathbb{X}} f(x) = f(x^\star)$$

$$\text{or} \quad \alpha^u(f) = \mathrm{P}_{\mathbb{X}}\{f > u\} = \int_{\mathbb{X}} \mathbb{1}_{f>u} \mathrm{dP}_{\mathbb{X}}$$

$$\ldots$$

- The result of a pointwise evaluation of f carries information about f and quantities depending on f (in particular, $m(f)$, $\alpha^u(f) \ldots$)
- Expensive computer experiments: the number of evaluations is limited $\to$ $m(f)$, $\alpha^u(f)$, etc. must be estimated using a fixed number, say N, of evaluations of f.

The case of optimization

- In the context of rare events estimation and risk analysis, it is often desirable to assess the worst-case performance of a system, that is, to determine

$$M = \sup_{x \in \mathbb{X}} f(x)$$

or

$$m = \inf_{x \in \mathbb{X}} f(x)$$

  - ➟ f may be non-convex
  - ➟ this is a global optimization problem

- How to design a good optimization algorithm?
- In a context of risk analysis, and also in difficult economic environments, we want to use an optimization algorithm that will provide a robust estimation of the global optimum

Why local optimization methods may not be satisfactory in the domain of computer experiments?

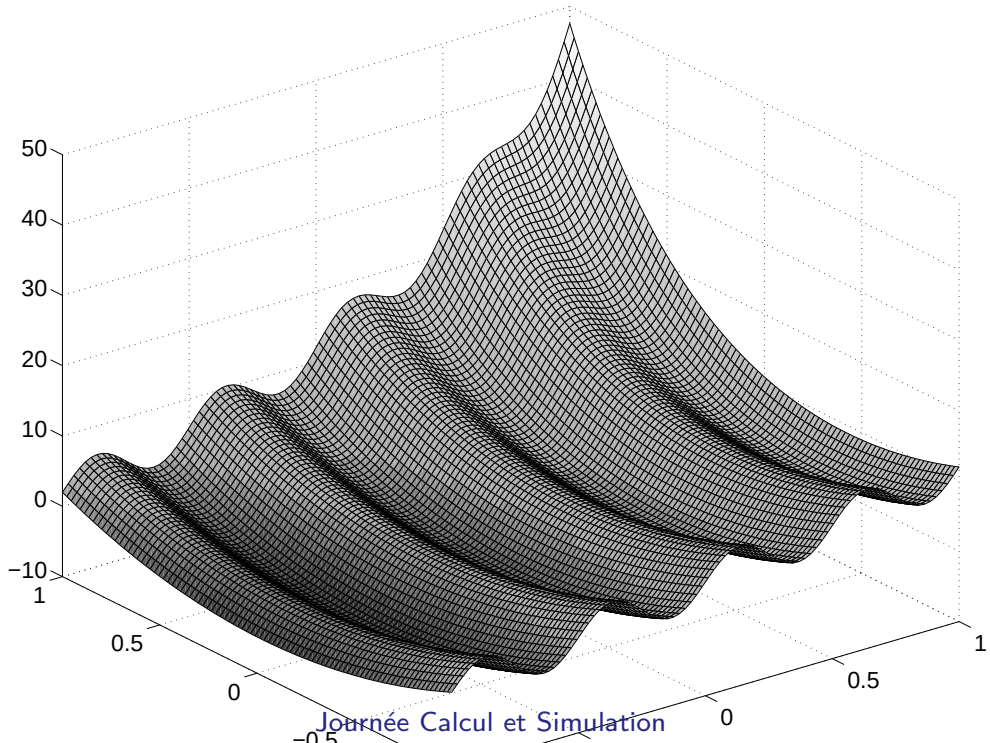
- An illustrative example: consider

$$\begin{array}{rccl} f: & \mathbb{R}^2 & \to & \mathbb{R} \\ & x & \mapsto & f(x) = \exp\left(1.8\left(x_{[1]} + x_{[2]}\right)\right) + 5x_{[1]} + 6{x_{[2]}}^2 + 3\sin\left(4\pi x_{[1]}\right) \end{array}$$

- Objective: find an approximation of

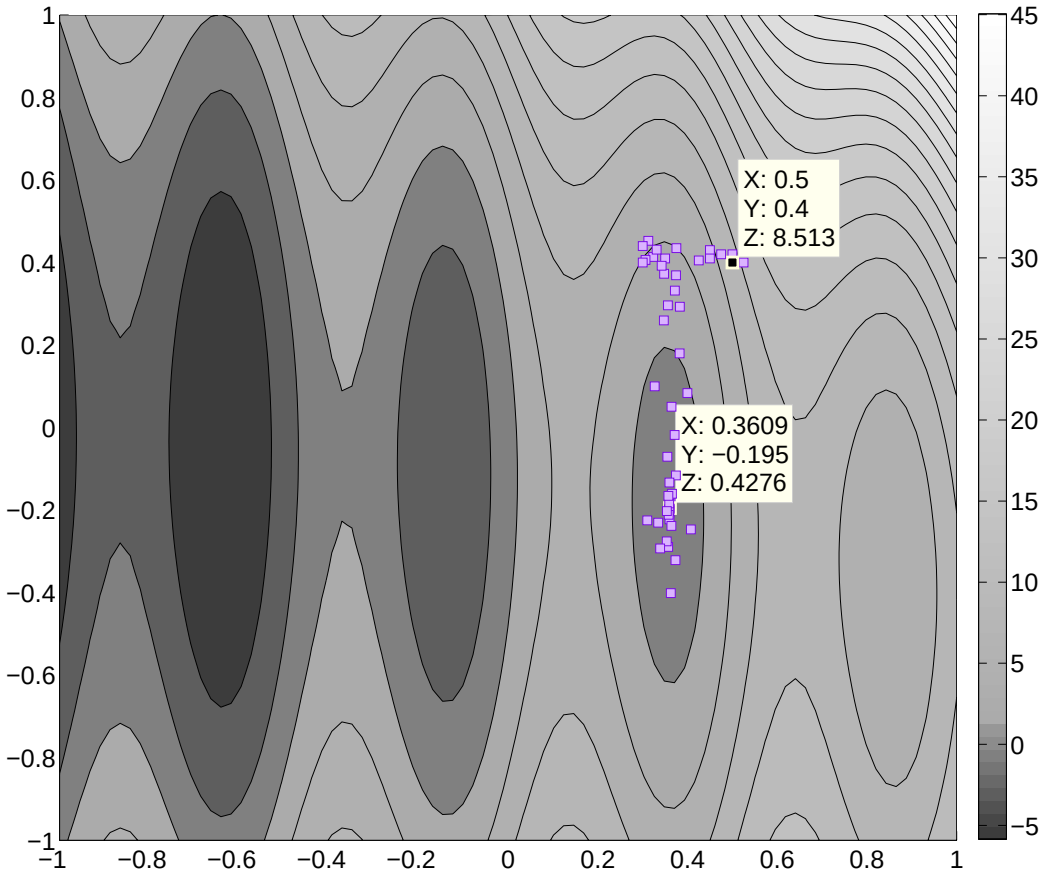
$$x^\star = \underset{x\in[-1,1]^2}{\operatorname{argmin}}\ f(x)\,.$$

with a budget of $N = 60$ experiments



Illustrative example (continued)

Evaluations points using a Nelder-Mead algorithm



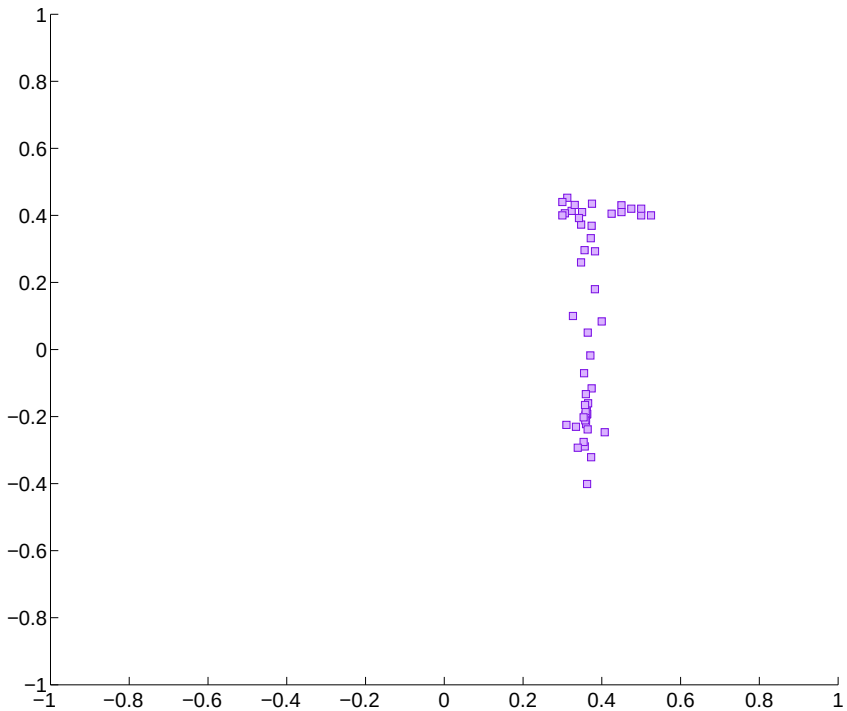
$\rightarrow$ the algorithm converges to a local minimum (≈ 0.427)

This comes as no surprise (local search algorithm). But above all...

- after having spent the budget of (possibly expensive) evaluations, the behavior of the function is only known in a small region of the search domain
- the global behavior of the function is unknown
- potentially interesting regions have not been explored

This comes as no surprise (local search algorithm). But above all...

- after having spent the budget of (possibly expensive) evaluations, the behavior of the function is only known in a small region of the search domain



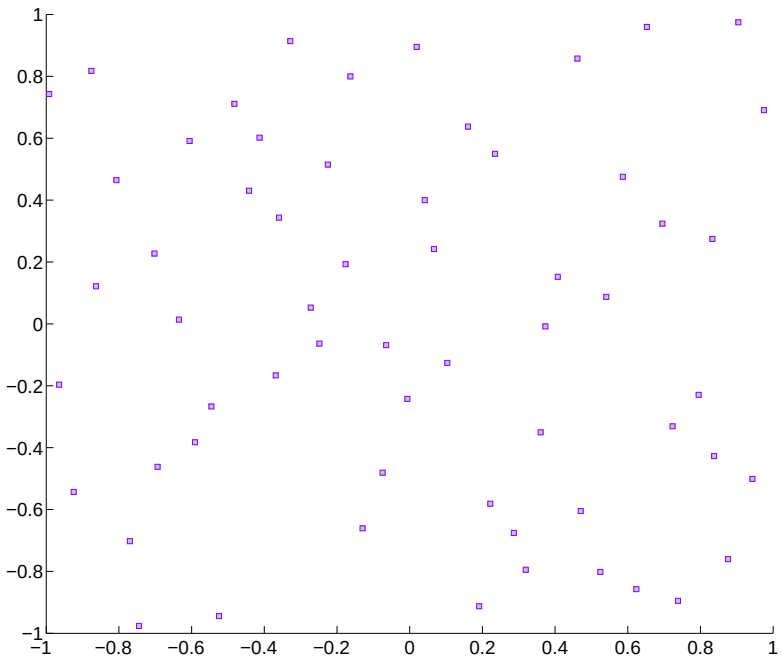
- the global behavior of the function is unknown
- potentially interesting regions have not been explored

- In a context of expensive-to-evaluate functions and a small budget of evaluations, it seems "safer" to achieve a balance between local search and exploration of the search domain
- Uniform random sampling:

$\rightarrow$ minimum of evaluation results is ≈ -5.823 (global minimum is ≈ -5.845)

- What is a robust optimization strategy? How to obtain such a strategy?

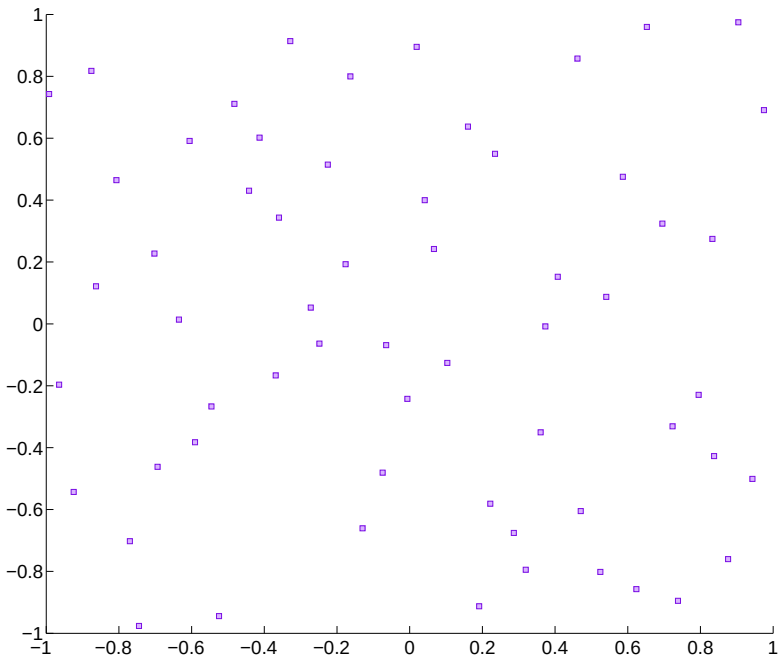
- In a context of expensive-to-evaluate functions and a small budget of evaluations, it seems "safer" to achieve a balance between local search and exploration of the search domain
- Uniform random sampling:



$\rightarrow$ minimum of evaluation results is ≈ -5.823 (global minimum is ≈ -5.845)

- What is a robust optimization strategy? How to obtain such a strategy?

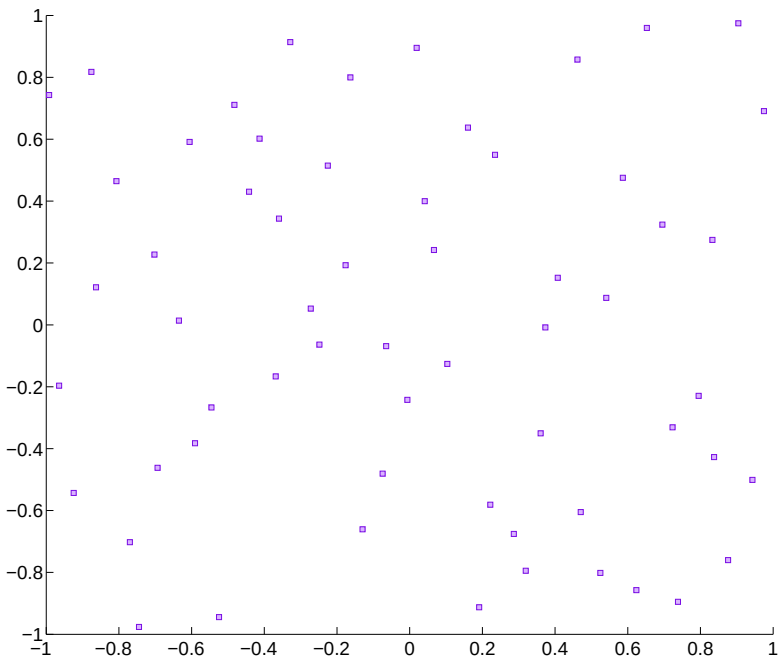
- In a context of expensive-to-evaluate functions and a small budget of evaluations, it seems "safer" to achieve a balance between local search and exploration of the search domain
- Uniform random sampling:



$\rightarrow$ minimum of evaluation results is ≈ -5.823 (global minimum is ≈ -5.845)

- What is a robust optimization strategy? How to obtain such a strategy?

- In a context of expensive-to-evaluate functions and a small budget of evaluations, it seems "safer" to achieve a balance between local search and exploration of the search domain
- Uniform random sampling:



$\rightarrow$ minimum of evaluation results is ≈ -5.823 (global minimum is ≈ -5.845)

- What is a robust optimization strategy? How to obtain such a strategy?

The worst-case approach

- Let $\mathcal{A}_N$ be the class of all optimization strategies $\underline{X}_N$ that query sequentially N evaluations of f at points $X_1, \ldots, X_N$.
- Define the error of approximation of a strategy $\underline{X}_N \in \mathcal{A}_N$ on f as

$$\varepsilon(\underline{X}_N, f) = \widehat{m}_N(f) - m(f)$$

with $\widehat{m}_N(f) = f(X_1) \wedge \cdots \wedge f(X_N)$

- Assume that f belongs to a class of functions $\mathcal{F} \rightarrow$ prior information

➜ A first idea to define a notion of a good strategy is to consider robustness with respect to a worst case

- Define the minimax risk

$$r_{\mathrm{minimax}}(\mathcal{F}) = \inf_{\underline{X}_N \in \mathcal{A}_N} \sup_{f \in \mathcal{F}} \varepsilon(\underline{X}_N, f)$$

- A strategy ${\underline{X}_N}^\star$ that attains $r_{\mathrm{minimax}}(\mathcal{F})$ is called an (optimal) minimax strategy
- ${\underline{X}_N}^\star$ has the best worst-case performance on $\mathcal{F}$

Example of a minimax strategy: case of Lipschitz functions

- Recall that a function $f : \mathbb{X} \to \mathbb{R}$ is Lipschitz continuous if there exists $K \geq 0$ such that, for all x_1 and x_2 in $\mathbb{X}$,

$$|f(x_1) - f(x_2)| \leq K\|x_1 - x_2\|.$$

  (Any such K is referred to as a Lipschitz constant for the function f.)

- Let $\mathcal{F}$ be the class of all Lipschitz continuous functions $\mathbb{X} \to \mathbb{R}$, with Lipschitz constant K

Example of a minimax strategy: case of Lipschitz functions

- For any strategy $\underline{X}_N$, define the fill distance as

$$h_N = \sup_{x \in \mathbb{X}} \min_{i=1,\ldots,N} |x - X_i|$$

- For any $\underline{X}_N \in \mathcal{A}_N$ and any $f \in \mathcal{F}$,

$$\varepsilon(\underline{X}_N, f) = f(X_1) \wedge \cdots \wedge f(X_N) - f(x^\star) \leq f(X_{i^\star}) - f(x^\star) \leq K h_N \,,$$

  where $X_{i^\star}$ is the nearest point to $x^\star$
- Thus, for any $\underline{X}_N \in \mathcal{A}_N$, $\sup_{f \in \mathcal{F}} \varepsilon(\underline{X}_N, f) \leq K h_n$
- For any $\underline{X}_N$, there exists a function $f \in \mathcal{F}$ such that

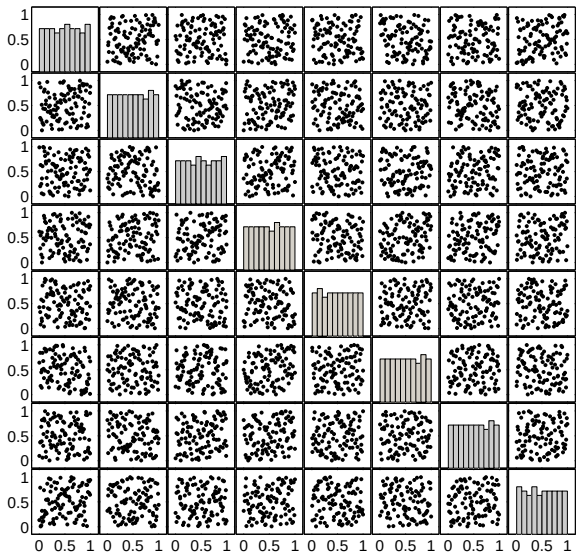
$$\varepsilon(\underline{X}_N, f) = K h_N$$

  Thus,

$$\sup_{f \in \mathcal{F}} \varepsilon(\underline{X}_N, f) = K h_N$$

Example of a minimax strategy: case of Lipschitz functions

- Consequence: a minimax strategy minimizes h_N
  $\rightarrow$ sample points have to be uniformly distributed over the search domain
- For $d = 1$, $\forall\, \underline{X}_N$, $h_N \geq \frac{|X|}{(N+1)} \implies$ the optimal strategy is the uniform sampling: $r_{\text{minimax}}(\mathcal{F}) = K\frac{|X|}{(N+1)}$
- For $d > 1$, use a *space-filling* design, *e.g.*., Maximin Latin Hypercube Sampling [McKay, Conover and Beckman (1979)] is an easy procedure that will generally provide good suboptimal designs



Example of a maximin Latin hypercube sampling of size $n = 100$ in dimension $d = 8$

The worst-case approach

- Consequence: for Lipschitz continuous functions, the minimax strategy consists in having sample points uniformly distributed over the search domain
- Here, the optimal strategy is non-adaptive!
- It may be more satisfying to achieve a balance between exploration of the search domain and local search in promising regions (good performance on worst cases and good convergence rate)
- Worst-case setting: appropriate framework to assess the performance of an optimization algorithm?
- We need to know how an optimization algorithm performs for "typical" functions f not corresponding to worst cases
- A classical approach is to adopt an average-case point of view

3. Average-case approach to the problem of optimization

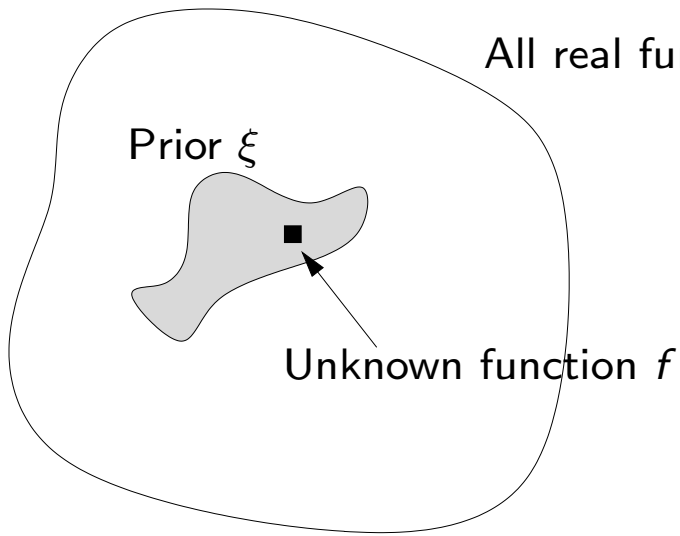
Average-case approach

- Average-case $\rightarrow$ introduction of a probability space $(\Omega, \mathcal{B}, \mathrm{P}_0)$
- We consider methods where f is seen as a sample path of a real-valued random process ξ defined on $(\Omega, \mathcal{B}, \mathrm{P}_0)$ with parameter in $\mathbb{X}$

  $\rightarrow$ there exists $\omega \in \Omega$ such that

$$f = \xi(\omega, \cdot)$$

- From a Bayesian decision-theoretic point of view, ξ represents prior knowledge about f



- A good strategy is a strategy that achieves, or gets close to, the optimal average risk

$$r_{\mathrm{average}} := \inf_{\underline{X}_N \in \mathcal{A}_N} \mathrm{E}_0 \left(\epsilon(\underline{X}_N, \xi) \right)$$

where E_0 denotes the expectation with respect to P_0

Expected Improvement [Mockus et al. 78, Schonlau et al. 96, Jones et al. 98]

- The optimal Bayesian one-step lookahead strategy for the problem of optimization corresponds to choosing each new evaluation point according to

$$\begin{aligned} X_{n+1} &= \operatorname*{argmin}_{x\in\mathbb{X}} \mathsf{E}_n\left(\widehat{m}_{n+1} - m \mid X_{n+1} = x\right) \\ &= \operatorname*{argmin}_{x\in\mathbb{X}} \mathsf{E}_n\left(\widehat{m}_{n+1} \mid X_{n+1} = x\right) \\ &= \operatorname*{argmin}_{x\in\mathbb{X}} \mathsf{E}_n\left(\widehat{m}_n \wedge \xi(X_{n+1}) \mid X_{n+1} = x\right) \\ &= \operatorname*{argmin}_{x\in\mathbb{X}} \mathsf{E}_n\left(0 \wedge (\xi(X_{n+1}) - \widehat{m}_n) \mid X_{n+1} = x\right) \\ &= \operatorname*{argmax}_{x\in\mathbb{X}} \rho_n(x) := \mathsf{E}_n\left((\widehat{m}_n - \xi(X_{n+1}))_+ \;\middle|\; X_{n+1} = x\right) \end{aligned}$$

  with

  - E_n conditional expectation wrt $\xi(X_1), \ldots, \xi(X_n)$
  - $\widehat{m}_n = \xi(X_1) \wedge \cdots \wedge \xi(X_n)$,
  - $z_+ = \max(z, 0)$

- The sampling criterion ρ_n is the expected improvement (EI)

  $\rightarrow$ average excursion of $\xi(x)$ below the current minimum of past evaluation results

- A well-known Bayesian optimization algorithm
  - proposed by Mockus et al.
  - popularized by the EGO algorithm of Jones et al.

Expected Improvement [Mockus 78, Schonlau et al. 96, Jones et al. 98]

- Assume ξ is a Gaussian process, with known mean and covariance functions
- Then, $\rho_n(x)$ has a closed-form expression:

$$\rho_n(x) \;=\; \gamma\left(\, m_n - \widehat{\xi}_n(x;\underline{X}_n),\, \sigma_n^2(x)\,\right),$$

where

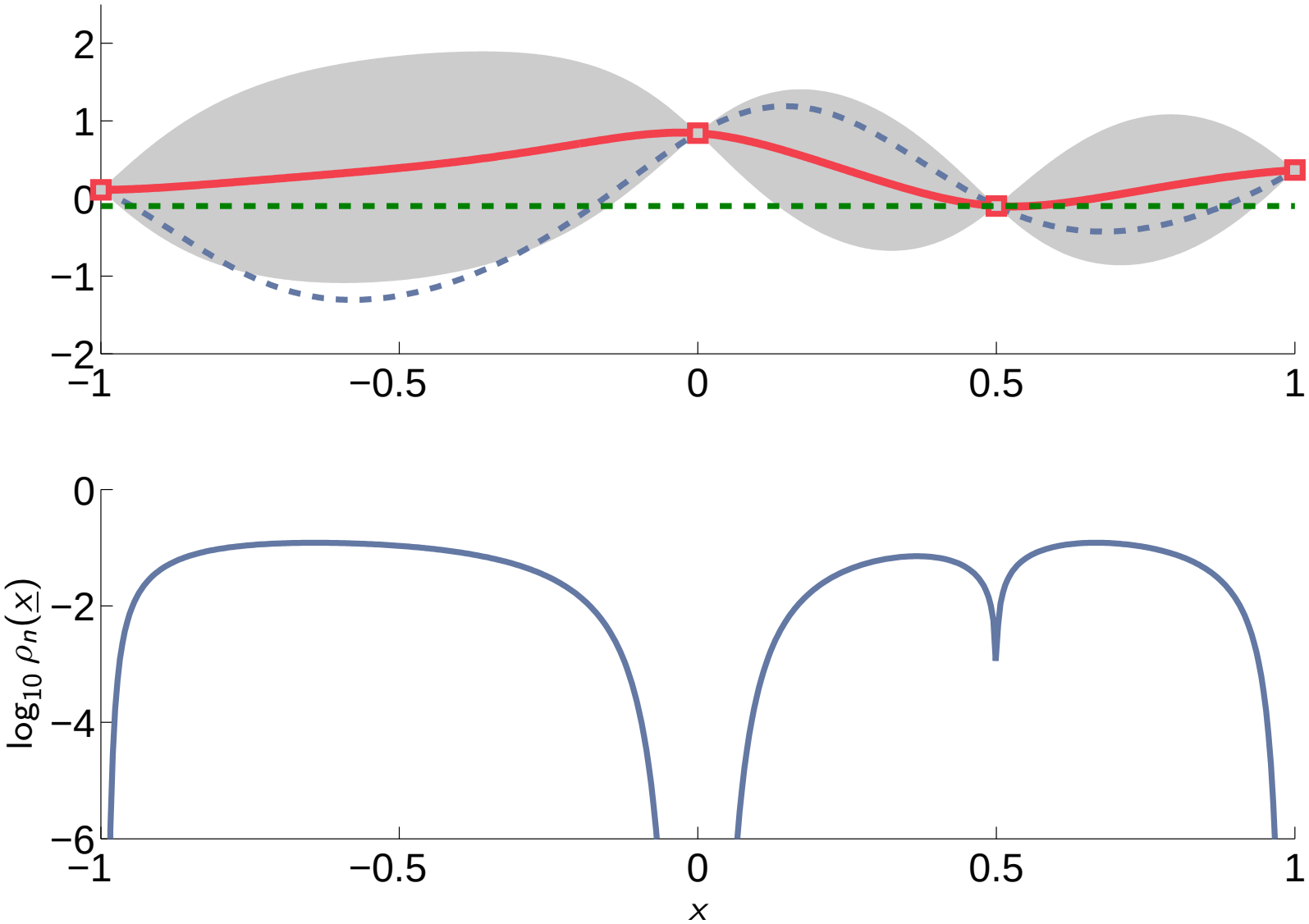
$$\gamma(z,s) \;=\; \begin{cases} \sqrt{s}\,\Phi'\left(\frac{z}{\sqrt{s}}\right) + z\,\Phi\left(\frac{z}{\sqrt{s}}\right) & \text{if } s>0, \\ \max(z,0) & \text{if } s=0. \end{cases}$$

and $\widehat{\xi}_n(x;\underline{X}_n)$ and $\sigma_n^2(x)$ are the kriging predictor and the kriging variance of $\xi(x)$ (Matheron, 1960) $\rightarrow$ see illustrating figure below.

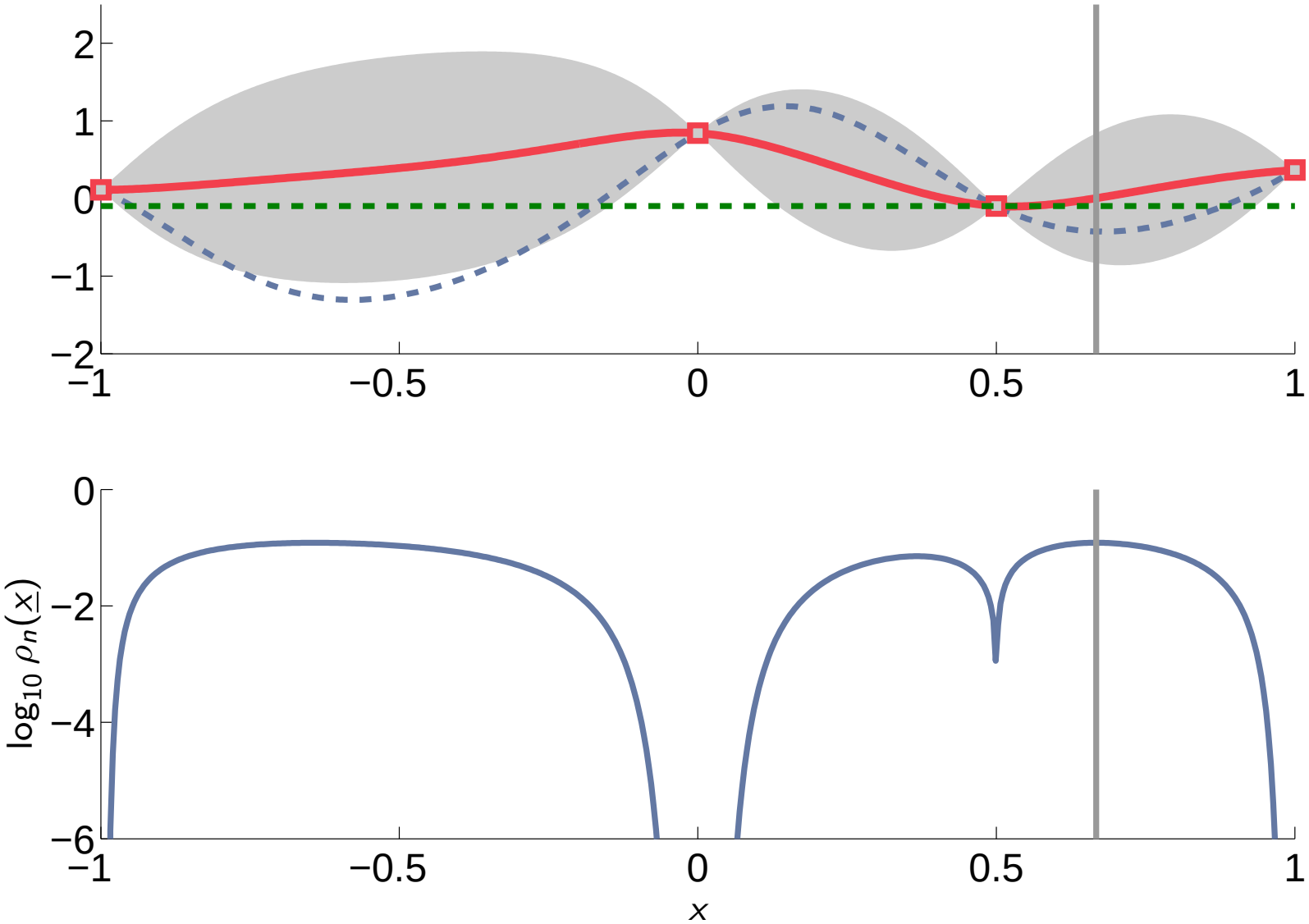
- The EI algorithm:

$$\begin{cases} X_1 & = & x_{\text{init}}\,, \\ X_{n+1} & = & \underset{x\in\mathbb{X}}{\text{argmax}}\ \rho_n(x)\,, \quad n\geq 1\,, \end{cases}$$

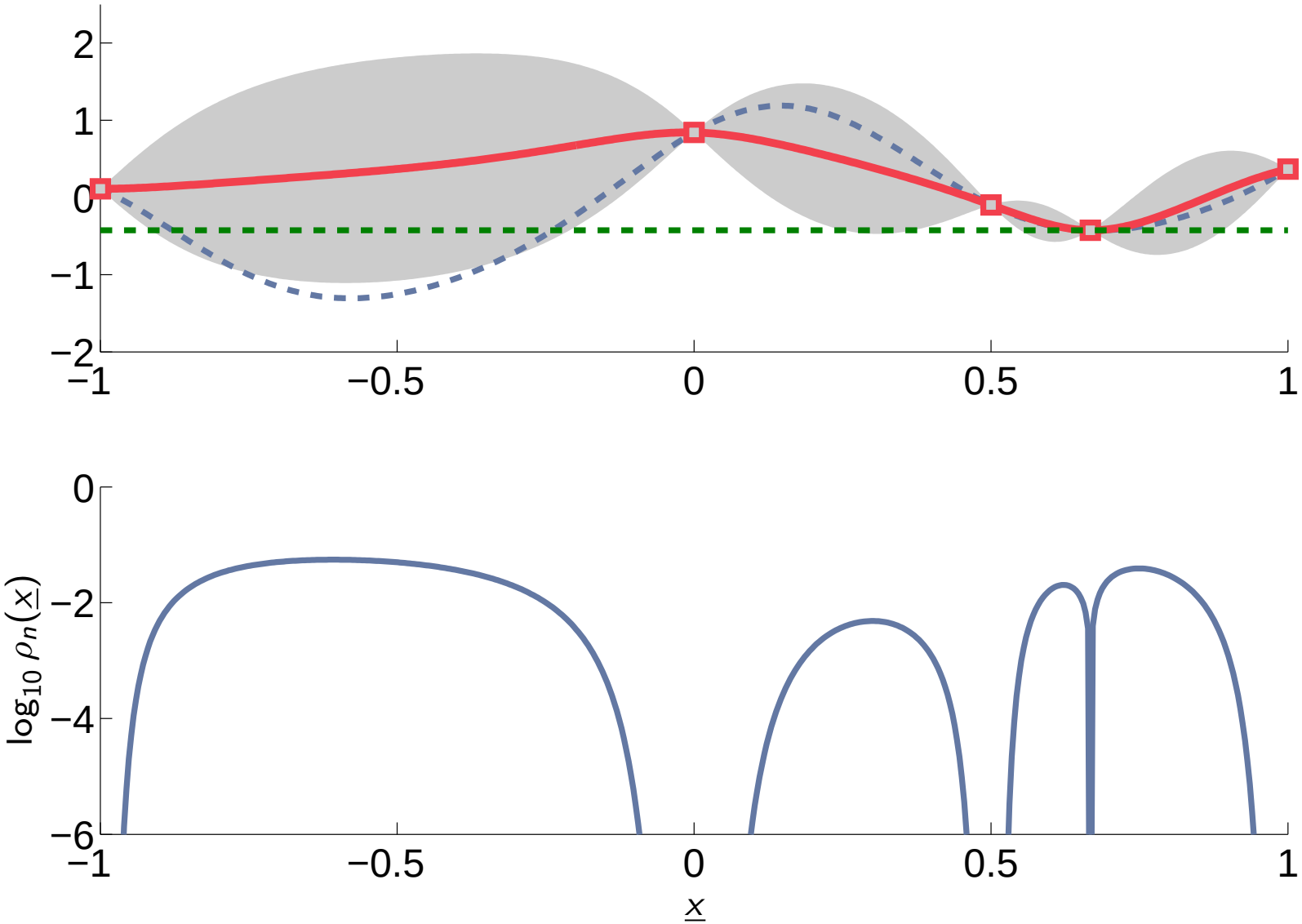
Global optimization based on EI



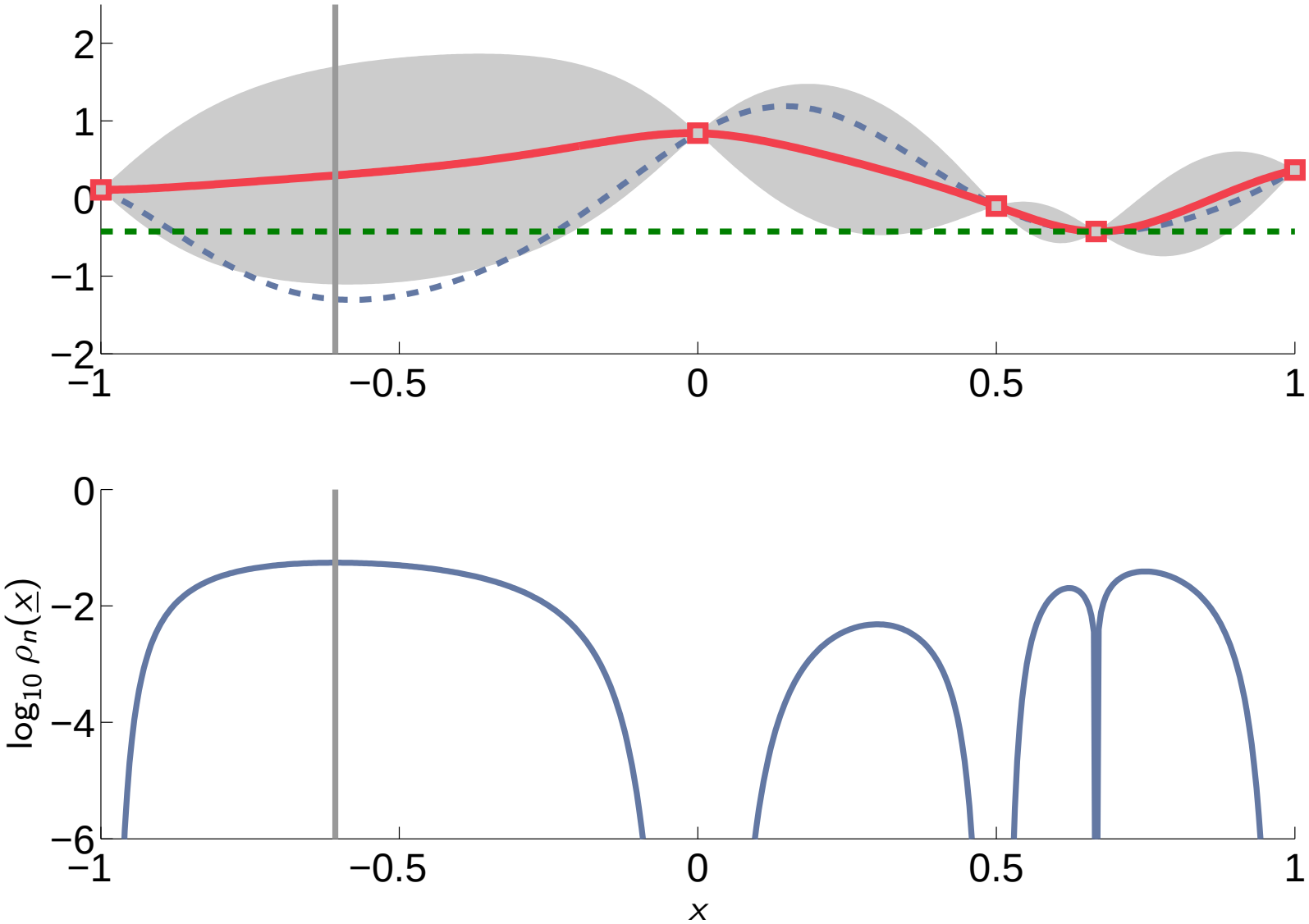
Global optimization based on EI



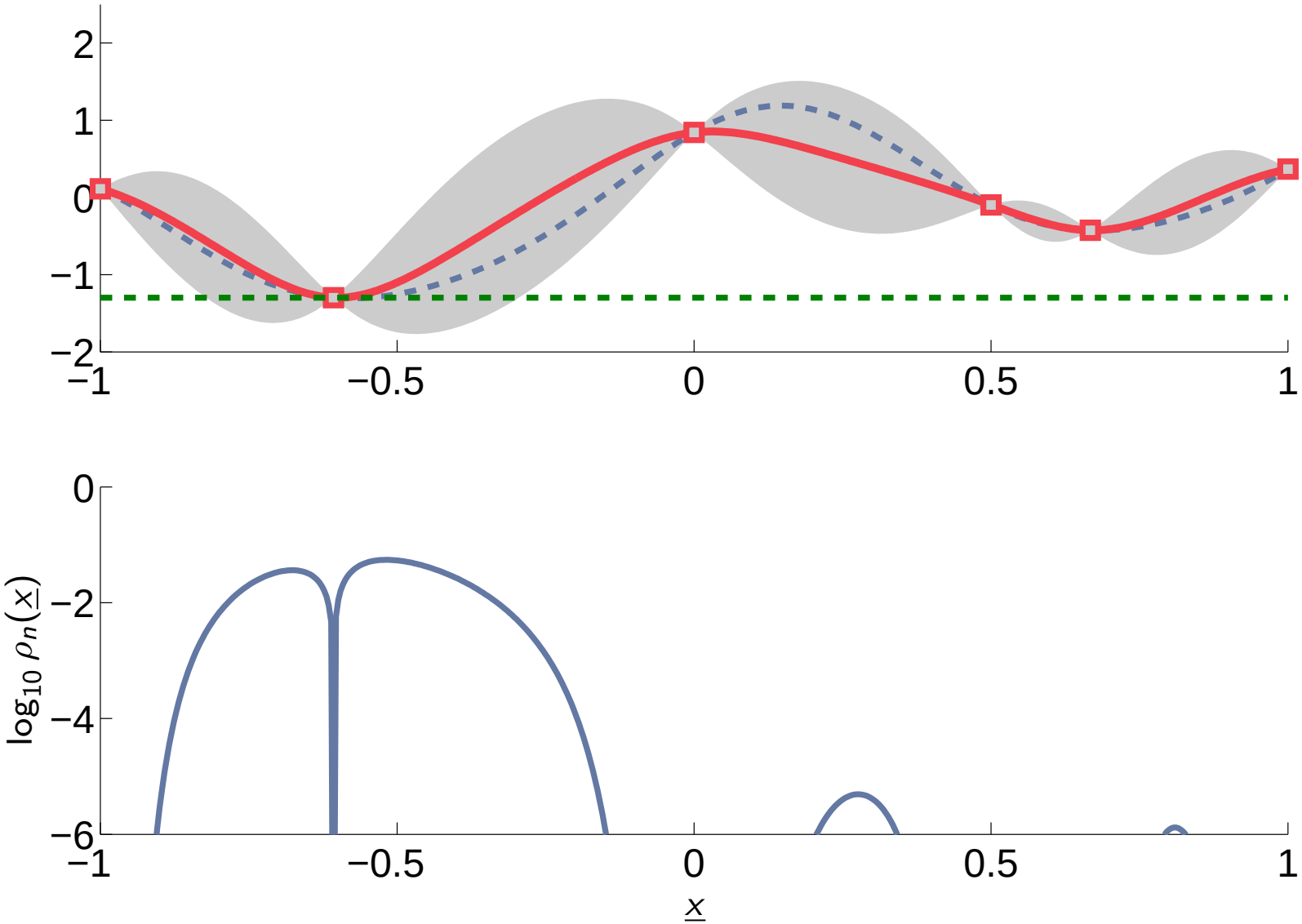
Global optimization based on EI



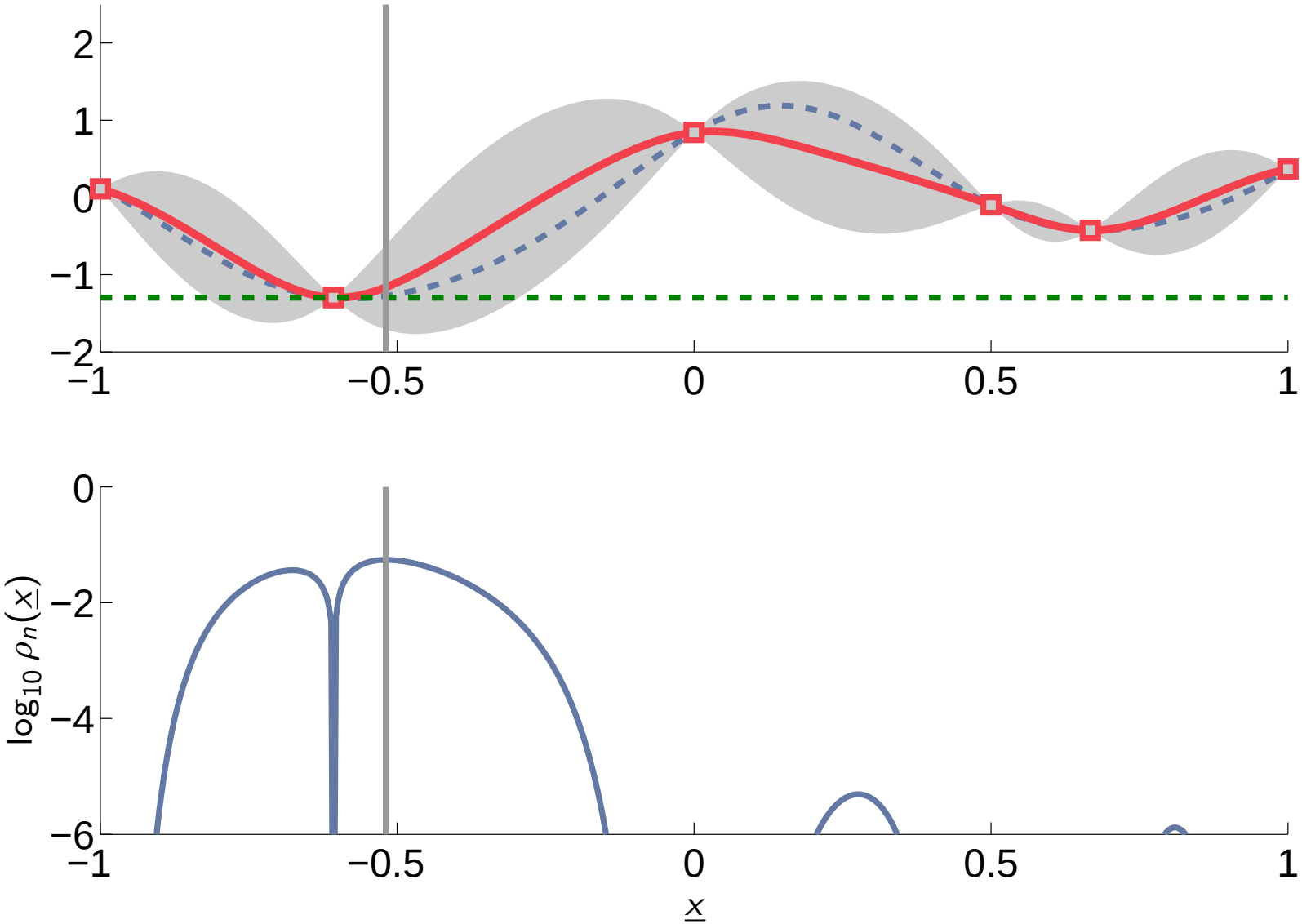
Global optimization based on EI



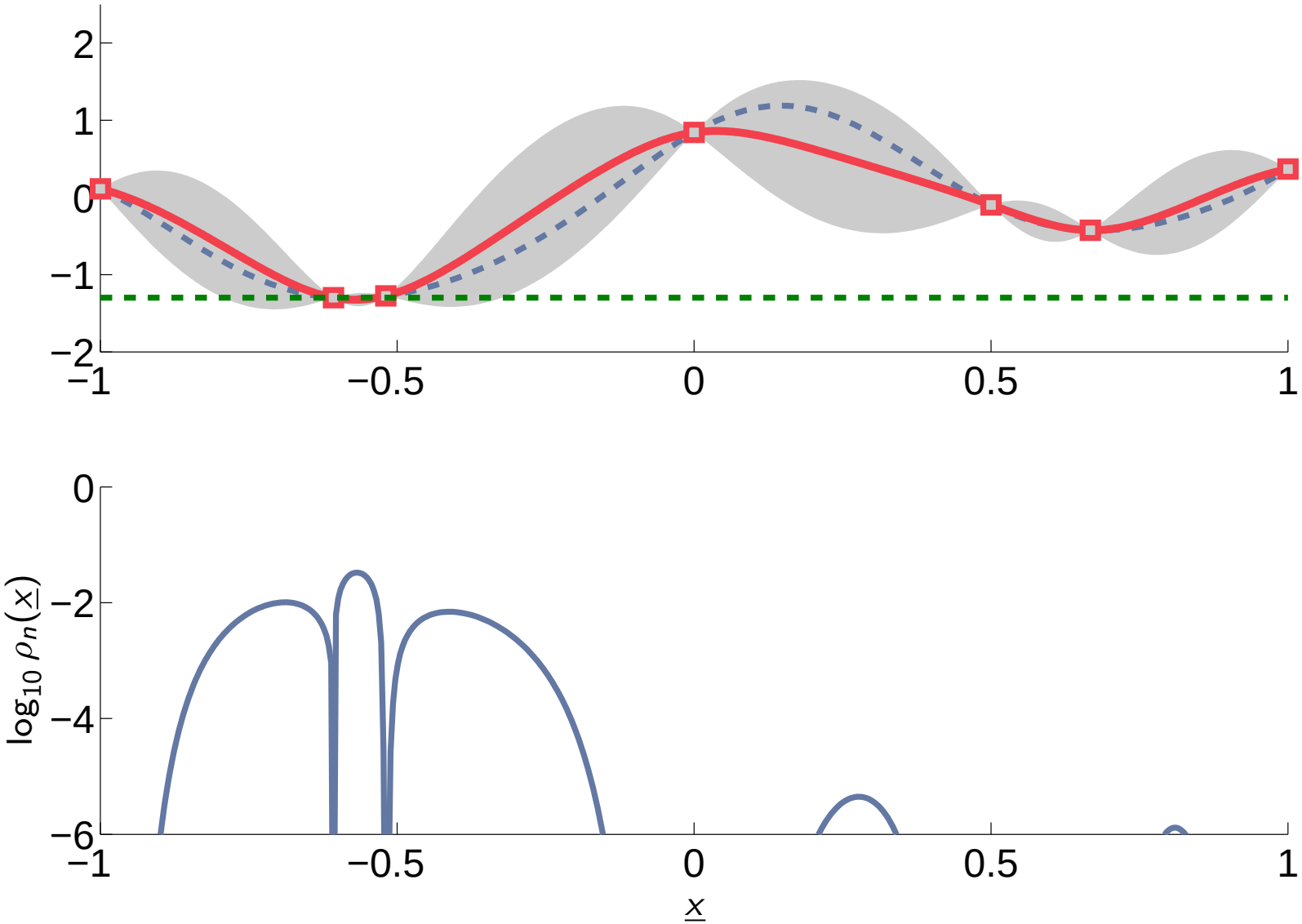
Global optimization based on EI



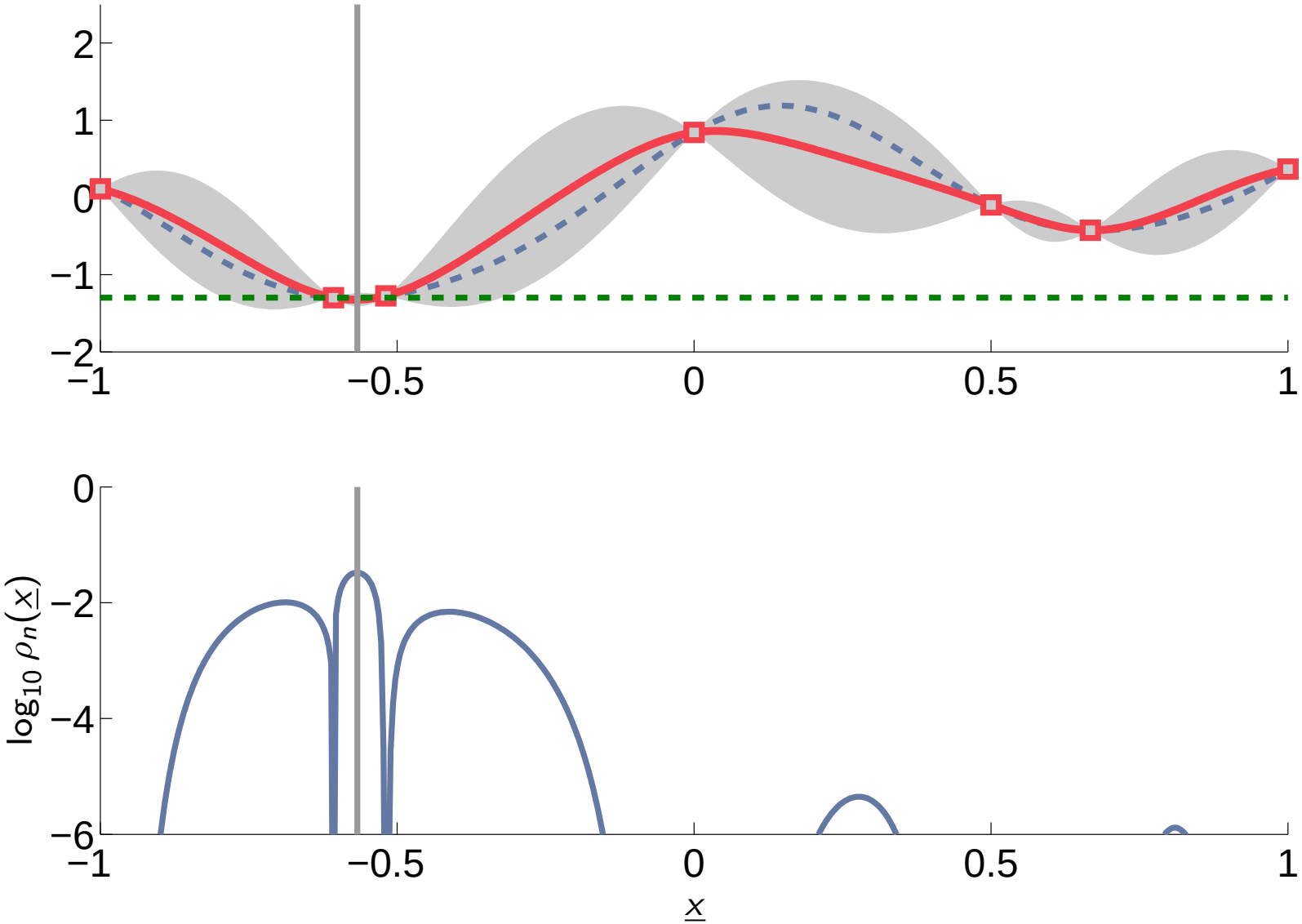
Global optimization based on EI



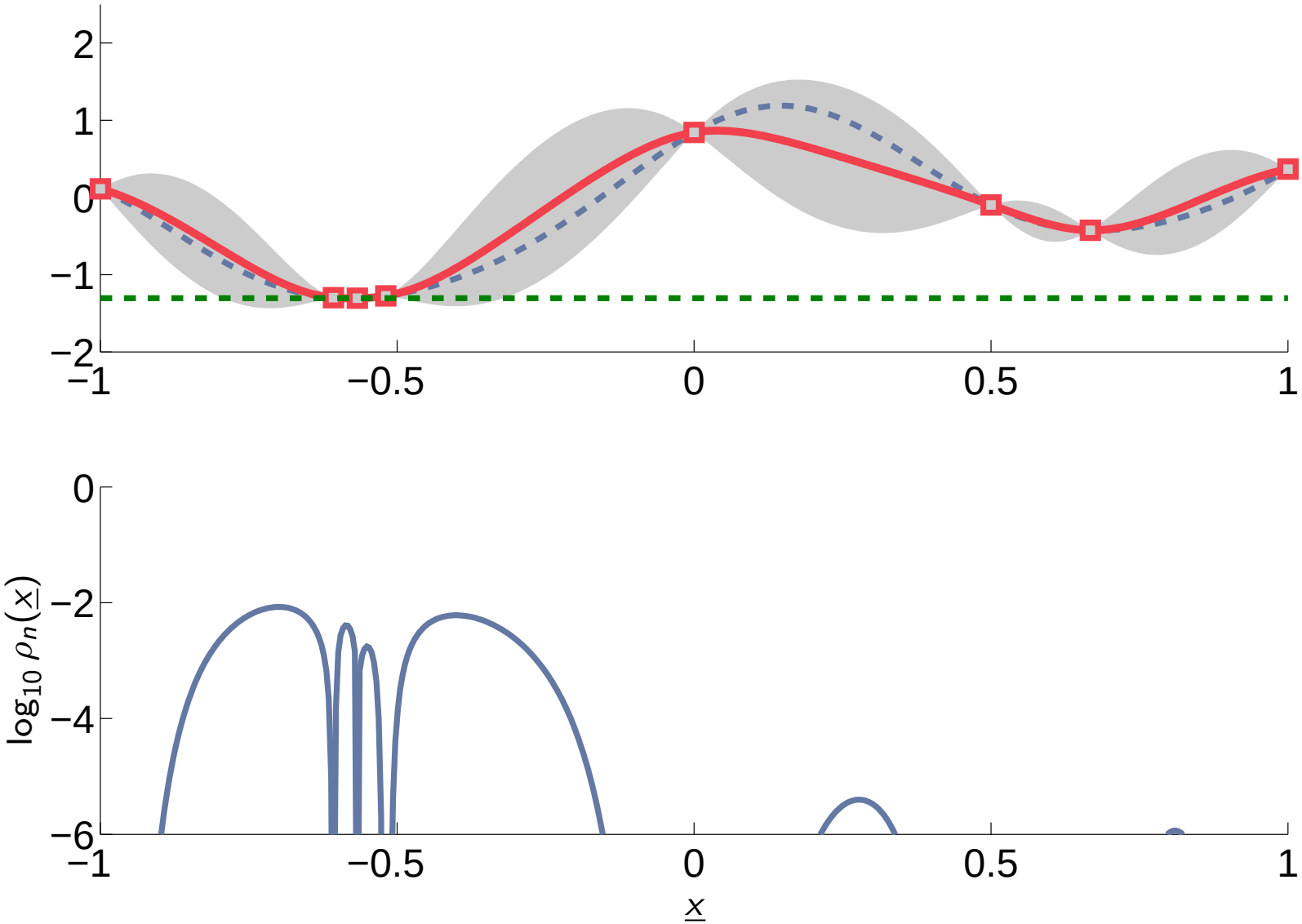
Global optimization based on EI



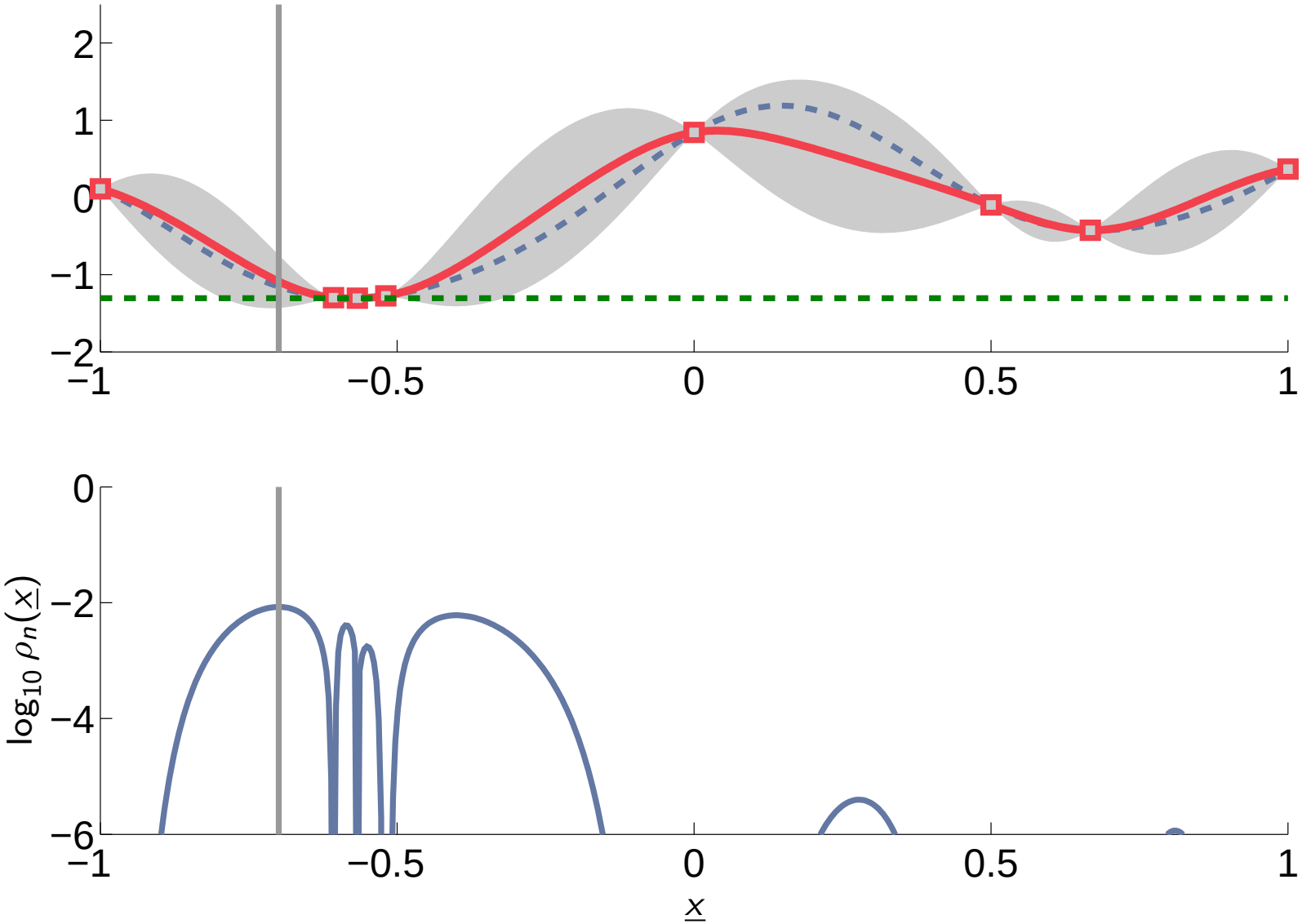
Global optimization based on EI



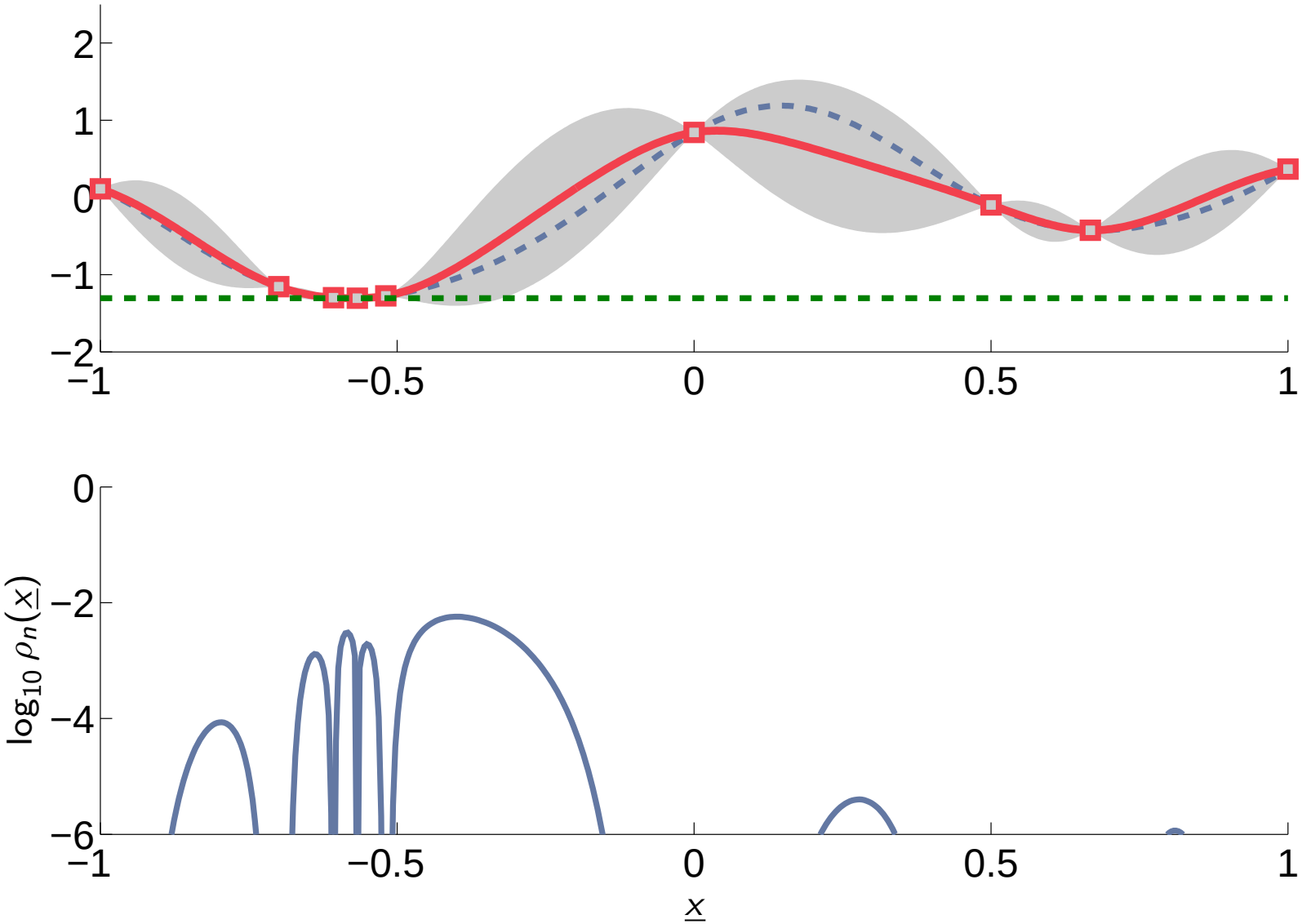
Global optimization based on EI



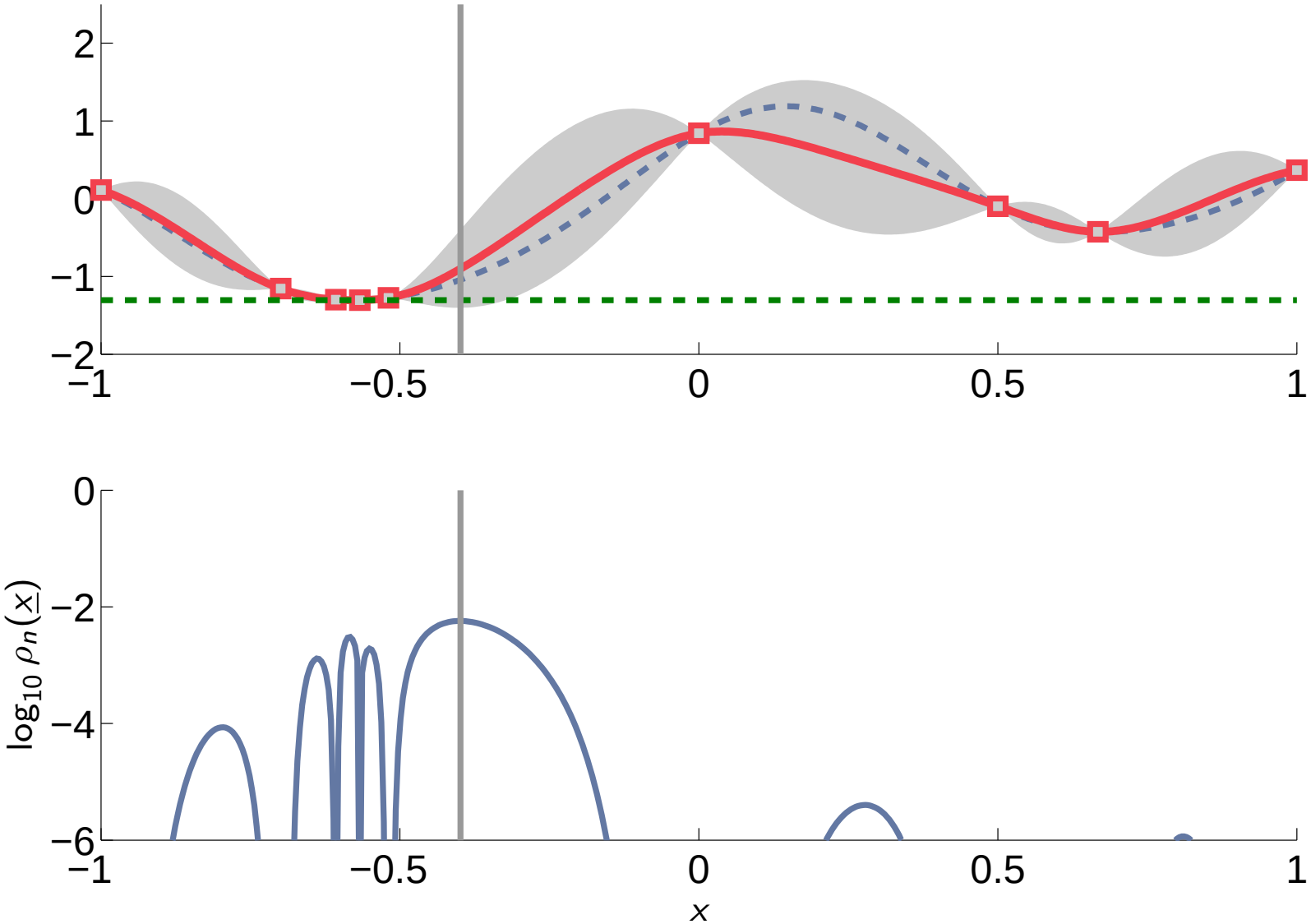
Global optimization based on EI



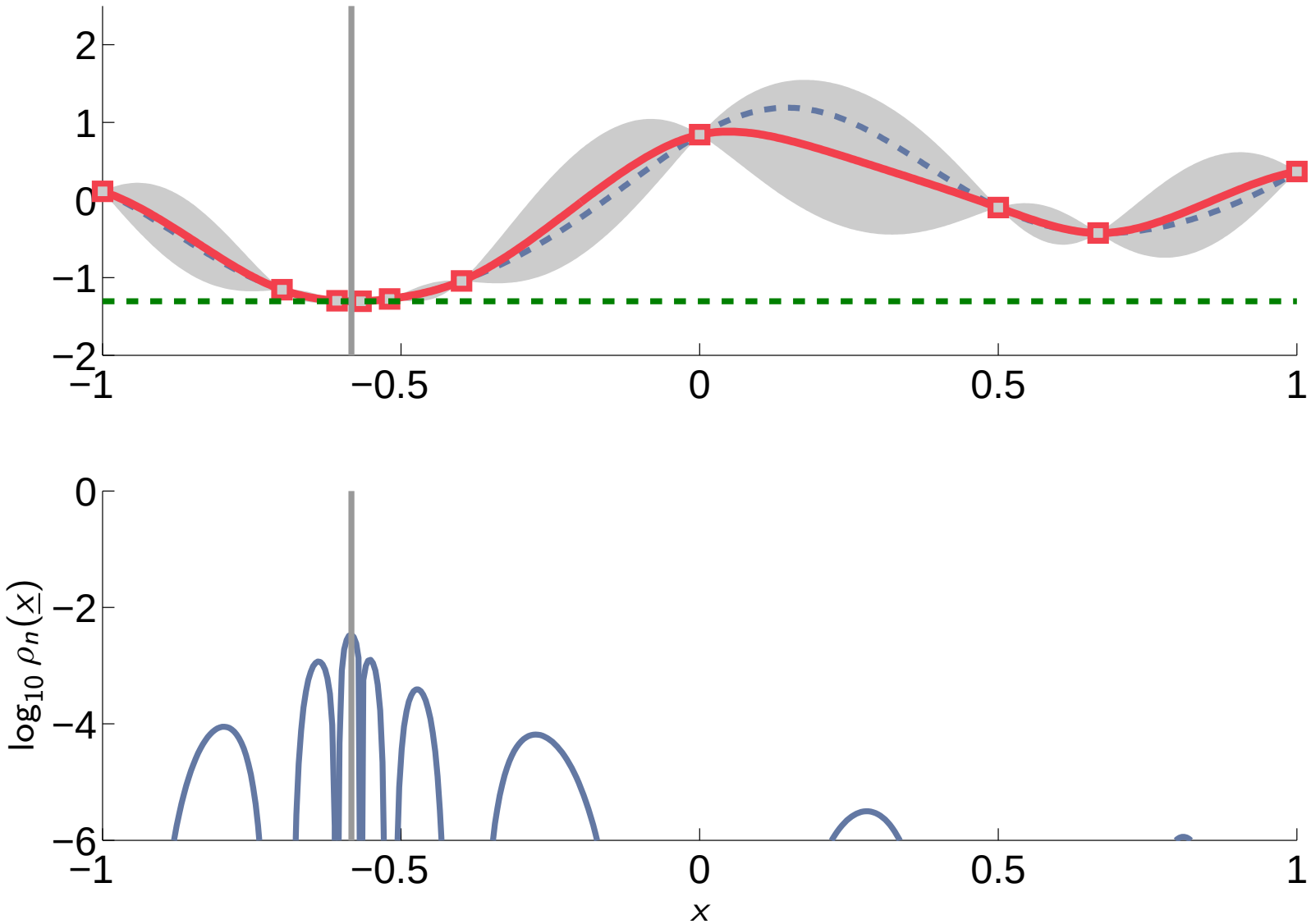
Global optimization based on EI



Global optimization based on EI

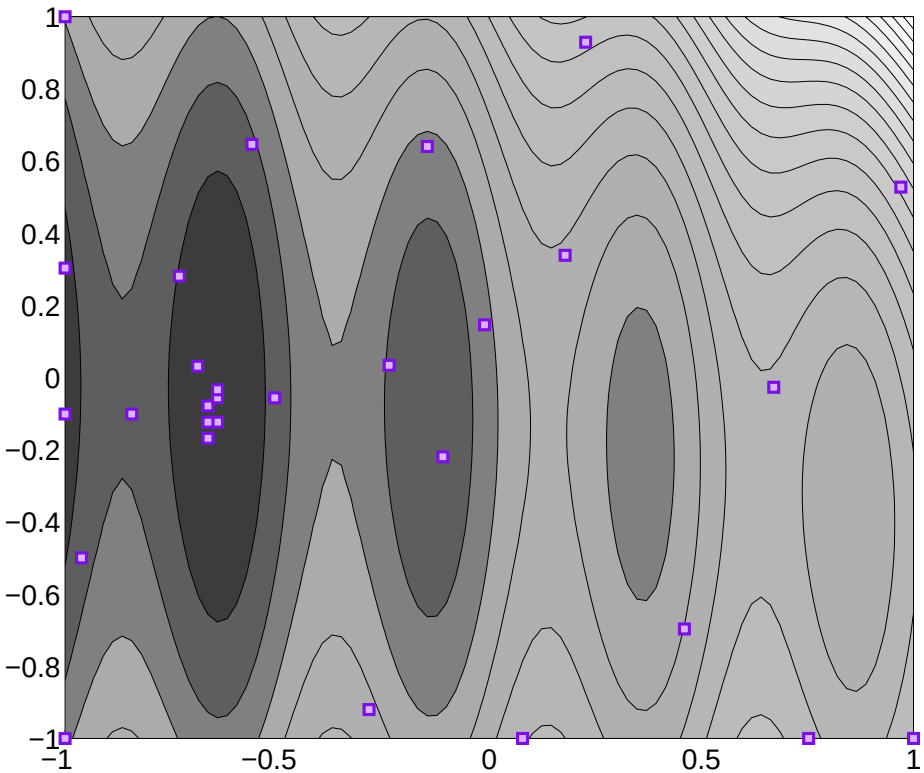


Global optimization based on EI



EI/EGO: 2D illustration

(f defined on Slide 12)



	$\widehat{m}_n$ with $N = 60$
LHS	-5.823
DIRECT	-5.839
EI/EGO	-5.845
Global minimum	-5.845

NB: Global minimum found by the EI algorithm in only 31 evaluations (abs. tol. 1.10^{-4})

Summing up

Global optimization based on EI

- Particularly interesting in the context of expensive-to-evaluate functions, very useful and effective in practical situations
- A great number of applications can be found in the literature (aeronautics, chemistry, energy...)
- Some theoretical results on the convergence of these algorithms: Vazquez & Bect 2010, Bull 2011...
- Efficient implementation based on SMC techniques: Benassi, Bect, Vazquez 2013

Concluding remarks

- In the context expensive simulations, Bayesian strategies show very good performances with respect to alternative approaches
- Bayesian strategies can be used for global optimization, estimation of probabilities of failure, quantile estimation...