(Re)Dice⋆ Packages for Computer Experiments

Olivier Roustant
Mines Saint-Étienne

Yves Deville
statistical consultant

representing all (Re)Dice⋆ package builders:

M. Binois, C. Chevalier, G. Damblin, D. Dupuy, J. Franco, D. Ginsbourger, C. Helbert, B. Iooss, V. Picheny, Y. Richet

and all other contributors linked to the (Re)Dice projects

Troisièmes rencontres R, Montpellier June 26-27, 2014.

Special acknowledgements

Some R package contributors also contributed to *this* talk:

- Mickaël Binois (1D EGO slides)
- David Ginsbourger (3D EGO slides)
- Yann Richet

Outline

- Computer Experiments and the (Re)Dice consortiums
- Designs for Computer Experiments
- Kriging Models and Gaussian Processes
- Kriging-based Optimization
- Playing with Kernels
- Practical Considerations

Part I

Computer experiments and (Re)Dice consortiums

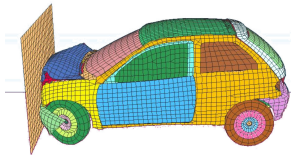
What are Computer Experiments?

design

What are Computer Experiments?

What are Computer Experiments?

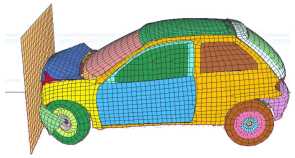
design → **experiment + measurements** → **meas. output**



simulator inputs $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ → **simulator** *physical model, numerical methods* → **simulator output** $f_{\text{sim}}(\mathbf{x}^1), f_{\text{sim}}(\mathbf{x}^2), \ldots, f_{\text{sim}}(\mathbf{x}^n)$

What are Computer Experiments?

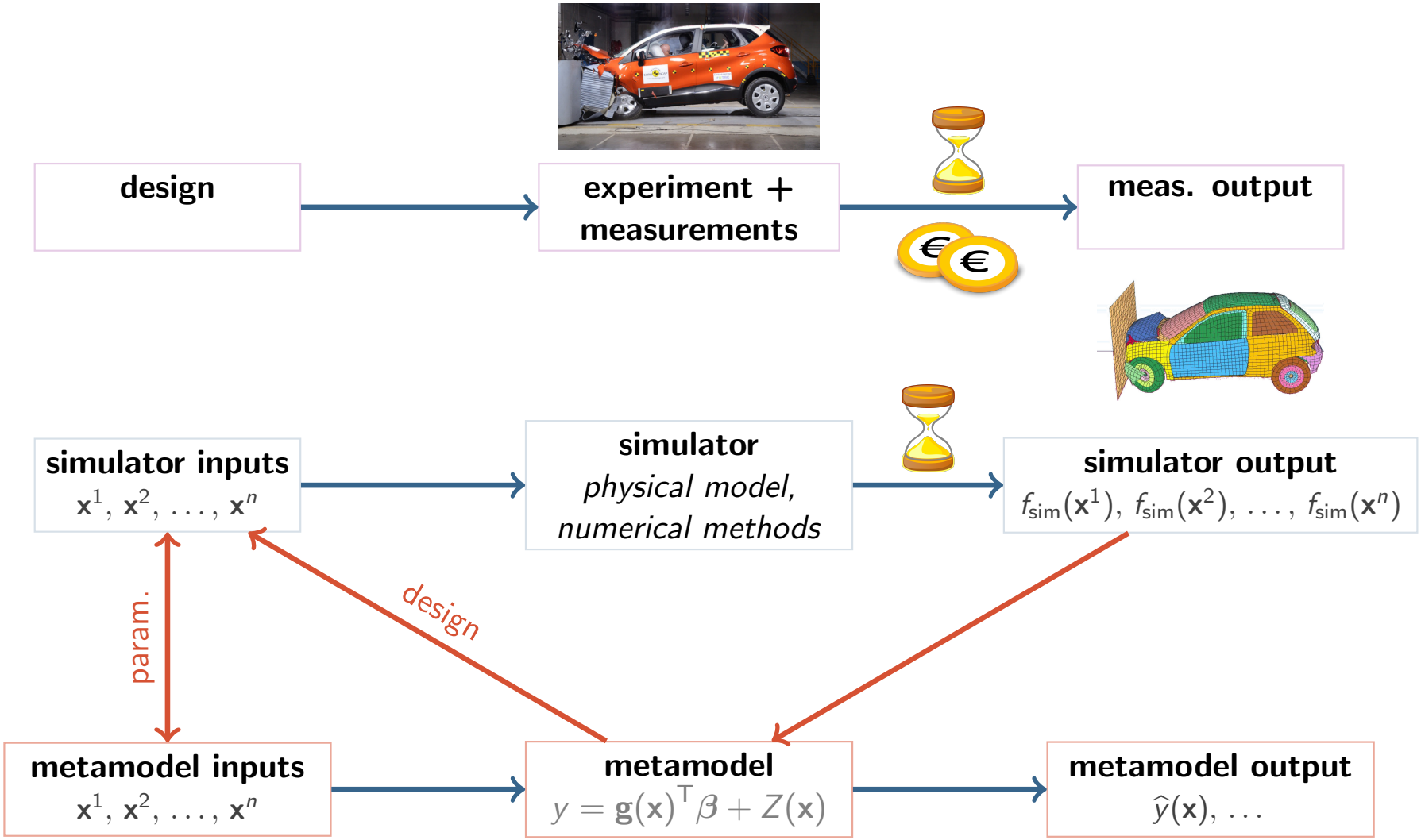
design → **experiment + measurements** → **meas. output**



simulator inputs $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ → **simulator** *physical model, numerical methods* → **simulator output** $f_{\text{sim}}(\mathbf{x}^1), f_{\text{sim}}(\mathbf{x}^2), \ldots, f_{\text{sim}}(\mathbf{x}^n)$

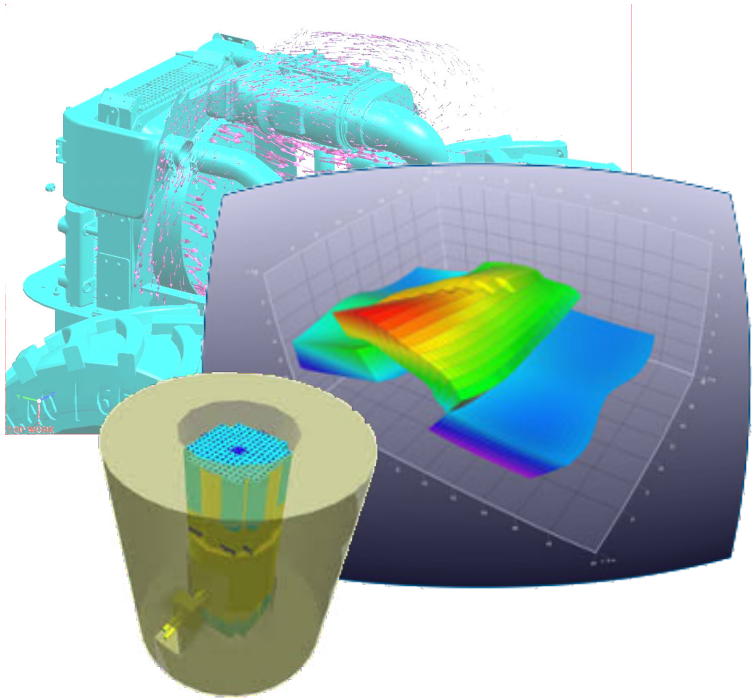
metamodel inputs $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ → **metamodel** $y = \mathbf{g}(\mathbf{x})^\top \beta + Z(\mathbf{x})$ → **metamodel output** $\widehat{y}(\mathbf{x}), \ldots$

What are Computer Experiments?



Who, where?

- Automotive
- Aerospace
- Oil
- Nuclear industry
- Hydrology, climate analysis
- Carbon sequestration
- ...



See e.g. the books by Fang et al [FLS06] or by Santner et al. [SWN03]

What for? (unclosed list...)

- Design of Computer Experiments

  $\longrightarrow$ choose inputs levels in order to best approximate the output
- Interpolation/approximation
- Global Optimization
- Sensitivity Analysis

  $\longrightarrow$ quantify the global effects of inputs on one output
- Inversion

  $\longrightarrow$ find regions where f_{sim} remains below some fixed threshold
- Risk Assessment

  $\longrightarrow$ e.g. estimate a quantile of an output for random inputs

How?

Metamodels come from old and new branches of statistics

- Polynomial models

  $\longrightarrow$ response surface
- Gaussian Processes and Kriging

  $\longrightarrow$ widely used in spatial statistics or spatio-temporal statistics
- Models from Statistical Learning

  $\longrightarrow$ SVM, RBF, Neural Networks, GAM, Mars, PolyMars, ...
- Polynomial Chaos

Dice and ReDice consortiums

Principle: get together with partners willing to share efforts on computer experiments

- Open problems
- Case studies
- Ideas, tips
- Code
- Training
- ...

Dice Consortium (2006-2009)

dice.emse.fr

- 5 industrials (EDF, IRSN, ONERA, Renault, TOTAL)
- 4 academics (EMSE, Univ. Aix-Marseille, Grenoble and Orsay)
- 3 PhD thesis funded (J. Franco, D. Ginsbourger, V. Picheny)

Outputs

- +30 internal reports, 8 publications
- 4 R packages released on CRAN

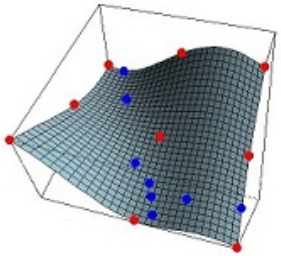
  → **DiceDesign**, **DiceEval**, **DiceKriging**, **DiceOptim**

Dice Consortium (2006-2009)

Industry	
EDF R&D	
IRSN	
Onera	
Renault	
Total	

Academics	
EMSE St Étienne	
Univ. PC Aix-Marseille	
Univ. PMF Grenoble	
Univ. PSud Orsay	

ReDice Consortium (2011-2015)



www.redice-project.org

- 5 industrials (EDF, CEA, IFPEN, IRSN, Renault)
- 5 academics (EMSE, Univ. Bern, Grenoble, Toulouse, Nice)
- 3 PhD thesis funded (M. Binois, C. Chevalier, F. Zertuche)

Differences with Dice

- Publication is enhanced by light property constraints

  $\longrightarrow$ $\emptyset$ for methodological articles, 6 month delay for R packages
- Training sessions (maths + software)

ReDice Consortium (2011-2015)

Industry	Academics
EDF R&D	**Univ. Berne**
IRSN	EMSE St Étienne
Cea	IMT Toulouse
Ifpen	Univ. Nice
Renault	Univ. PMF Grenoble
	Univ. J. Monnet St Étienne
Inria	

Why R?

Existing software in computer experiments: Written in C, Python, popular matrix languages, . . .

R was immediately well-accepted by the Consortiums

- *Lingua Franca* of statistical computing.
- Works on major platforms.
- Open-source and easy to extend.
- Can be used in many modes or fashions.
- Huge choice of updated CRAN packages.

→ write new packages and/or enhance existing ones!

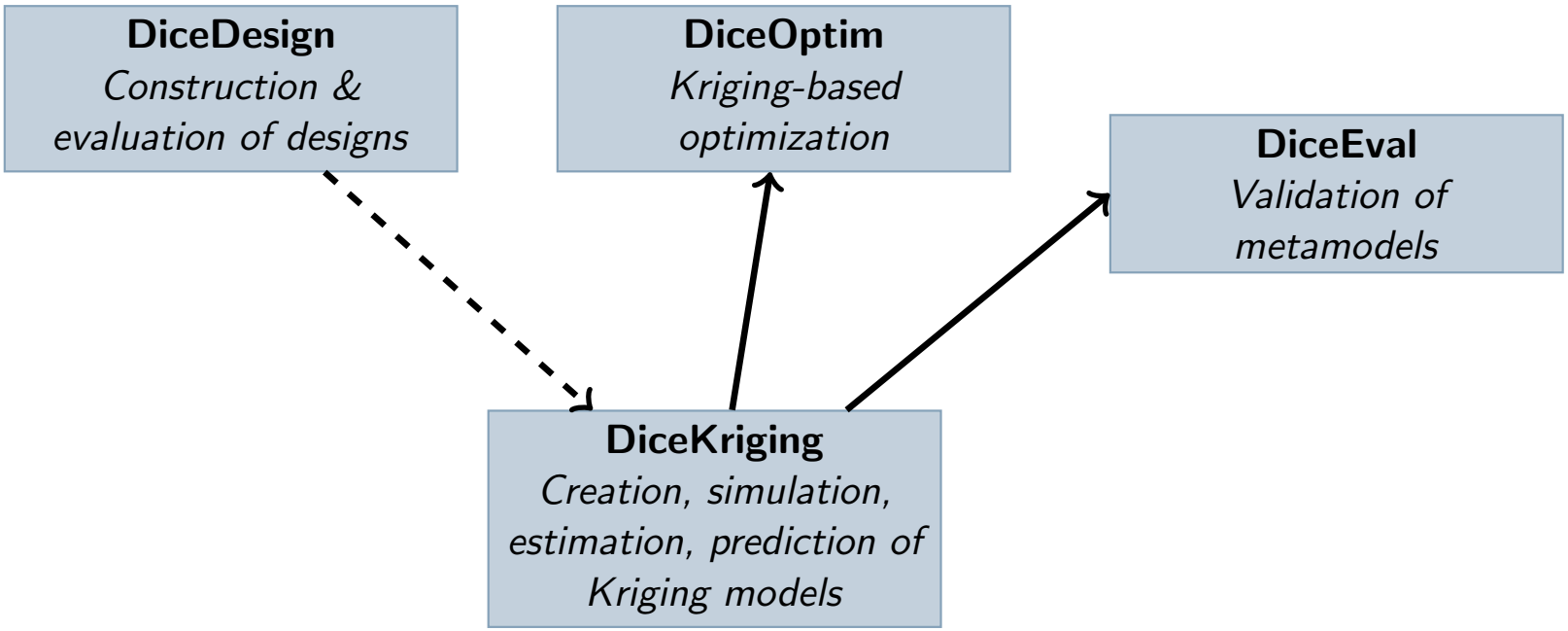
How new R packages are used

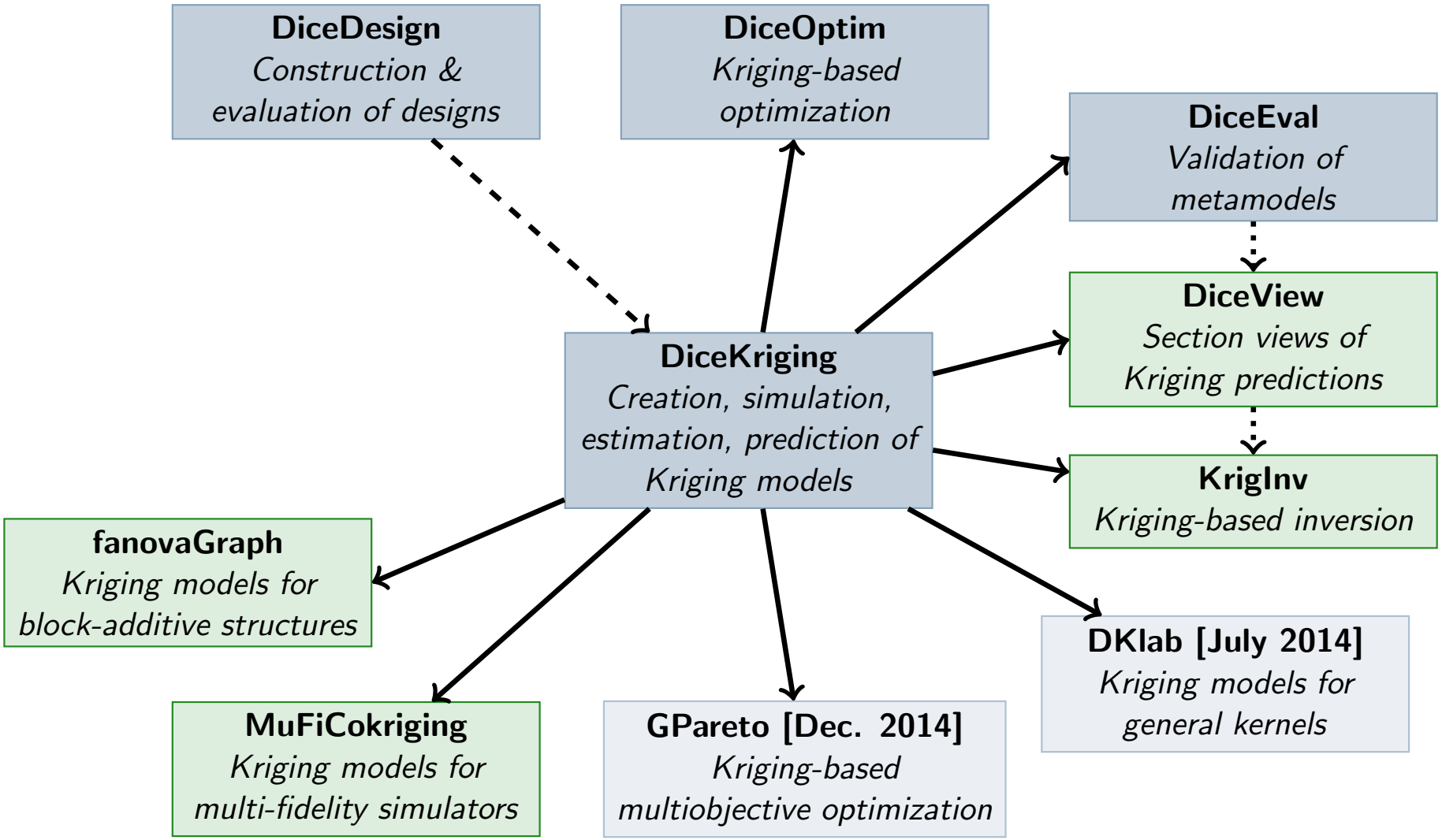
Industrials

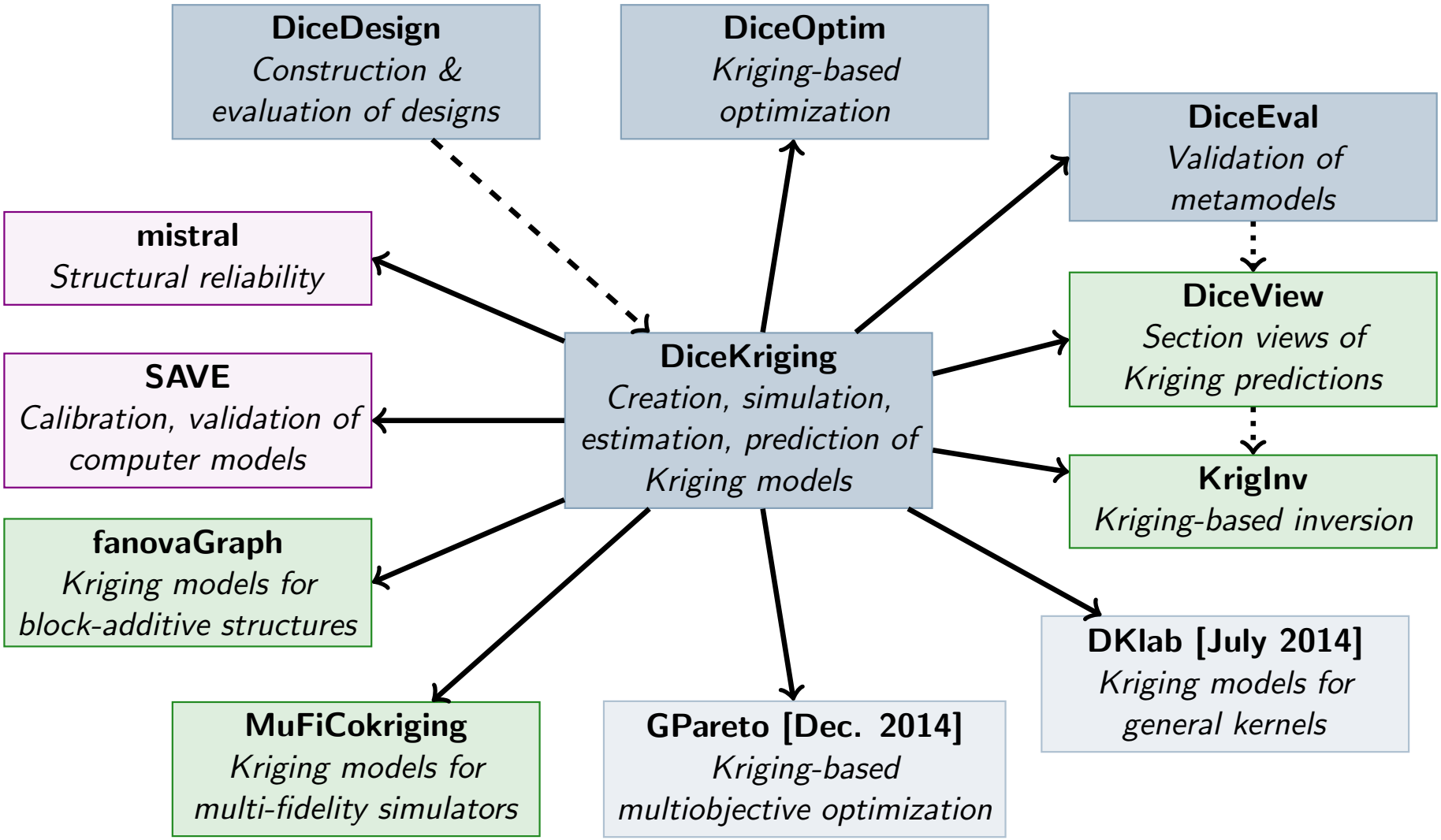
- To solve real-life problems with new methods.
- Often used in a complex computing environment, e.g. Java.
  $\longrightarrow$ *Prométhée* at IRSN, see later

Academics

- For methodological developments.
- Case studies, toy examples, benchmarks.
- Construction efforts are often valorized by publications.


DiceDesign
Construction & evaluation of designs
DiceOptim
Kriging-based optimization
DiceEval
Validation of metamodels
DiceKriging
Creation, simulation, estimation, prediction of Kriging models


DiceDesign
Construction & evaluation of designs
DiceOptim
Kriging-based optimization
DiceEval
Validation of metamodels
DiceView
Section views of Kriging predictions
DiceKriging
Creation, simulation, estimation, prediction of Kriging models
KrigInv
Kriging-based inversion
fanovaGraph
Kriging models for block-additive structures
DKlab [July 2014]
Kriging models for general kernels
MuFiCokriging
Kriging models for multi-fidelity simulators
GPareto [Dec. 2014]
Kriging-based multiobjective optimization


DiceDesign
Construction & evaluation of designs
DiceOptim
Kriging-based optimization
DiceEval
Validation of metamodels
mistral
Structural reliability
DiceView
Section views of Kriging predictions
SAVE
Calibration, validation of computer models
DiceKriging
Creation, simulation, estimation, prediction of Kriging models
KrigInv
Kriging-based inversion
fanovaGraph
Kriging models for block-additive structures
DKlab [July 2014]
Kriging models for general kernels
MuFiCokriging
Kriging models for multi-fidelity simulators
GPareto [Dec. 2014]
Kriging-based multiobjective optimization

Part II

Designs for Computer Experiments

Design specificities for computer experiments

Two main constraints and their consequences.

1. For deterministic simulators, running twice the simulator at the same location gives the same result

   $\longrightarrow$ Avoid replications

2. A simulator generally models a complex phenomenon with strongly non-linear behavior

   $\longrightarrow$ Fill the space, in order to avoid missing an area

Additional constraint:

3. The aforementioned principles should apply in lower subspaces, since the *true* dimensionality can be lower.

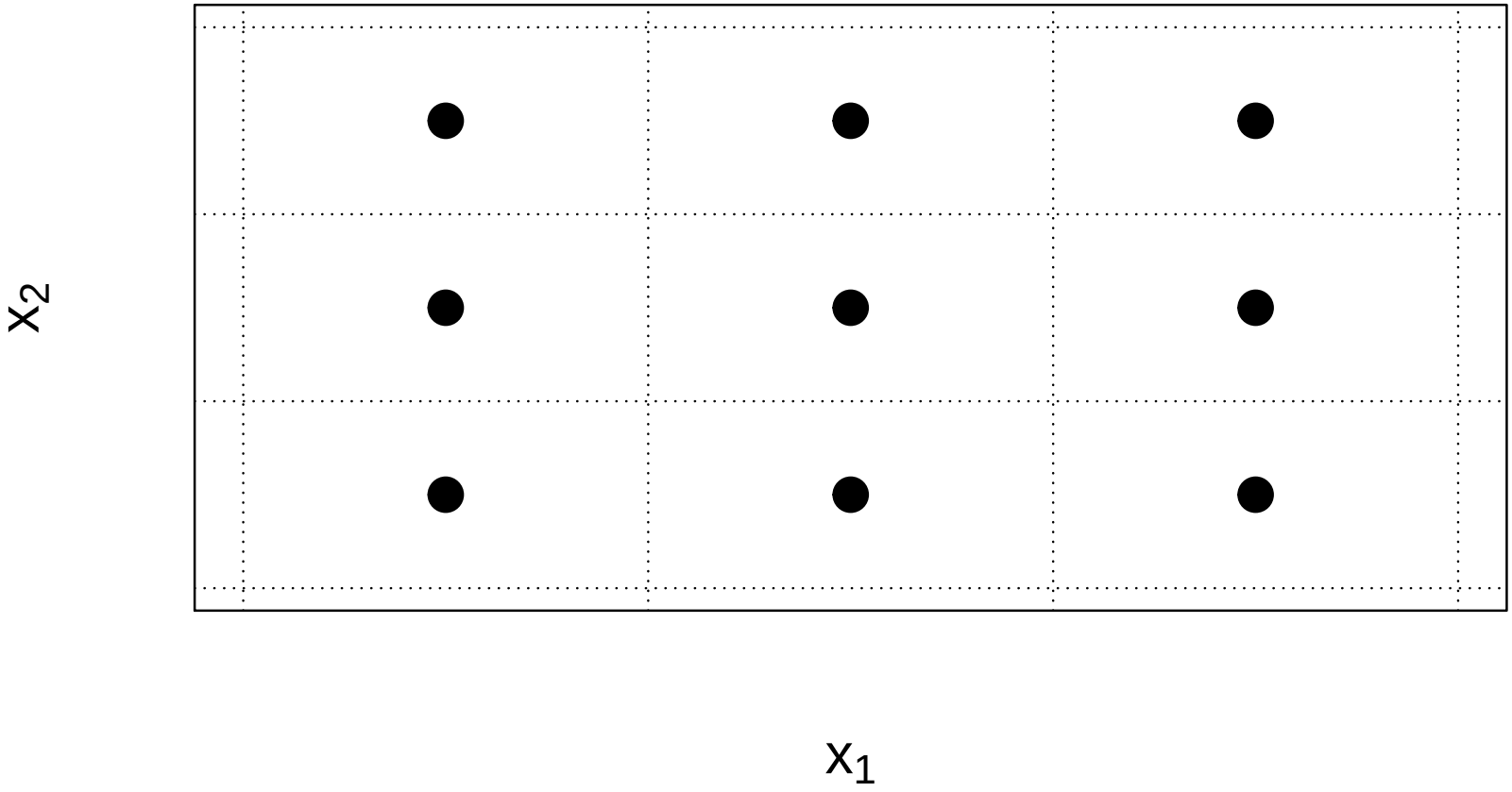
Example: $f_{\text{sim}}(x_1, x_2) = g(x_1)$

$\longrightarrow$ It is useless to run the simulator with same x_1 locations!

$\longrightarrow$ The x_1 locations should fill the space

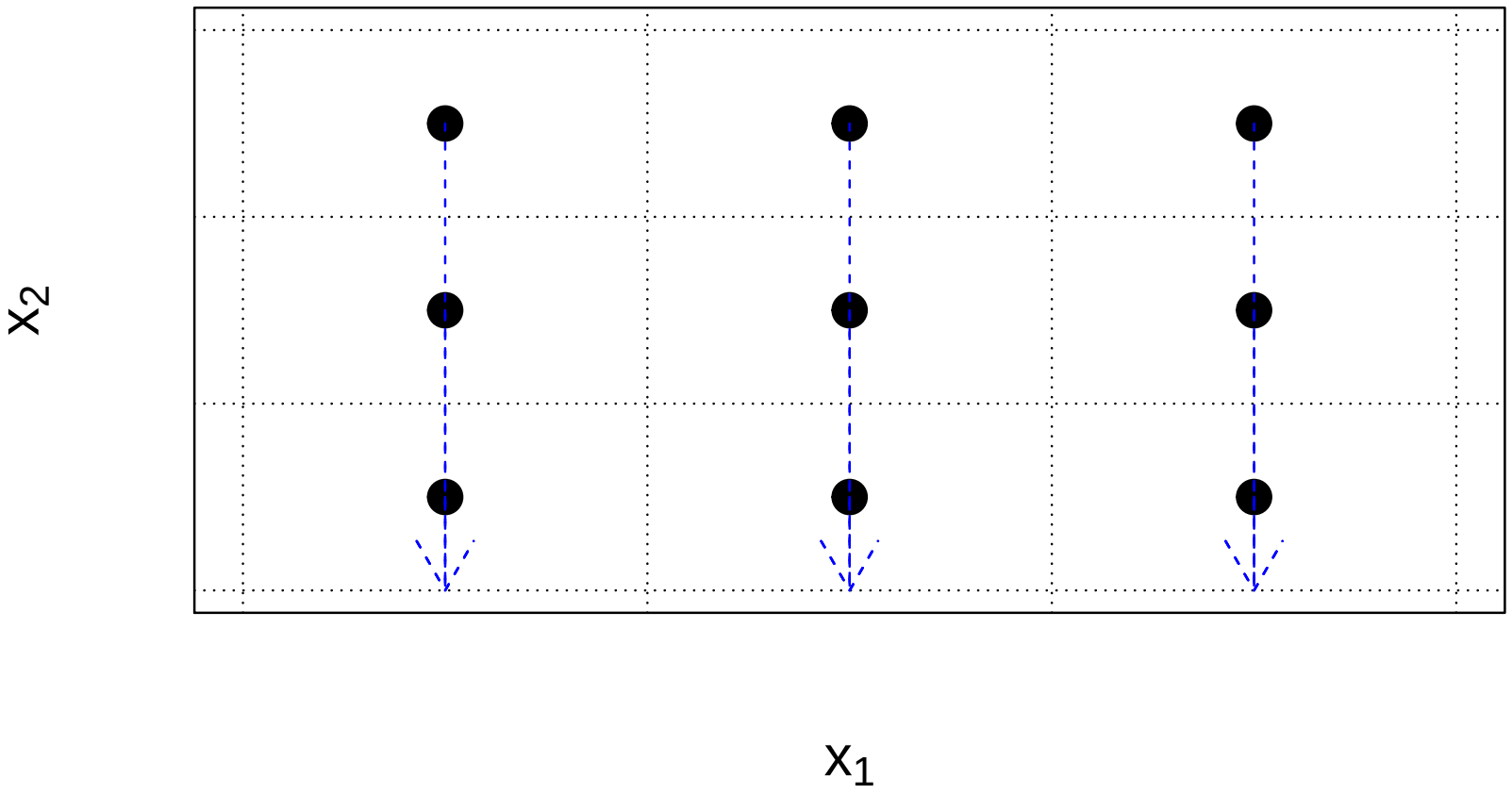
A potentially bad design: How to waste 2/3 of runs!

Only 3 points among 9 are useful if f_{sim} depends only on x_1 (or x_2).



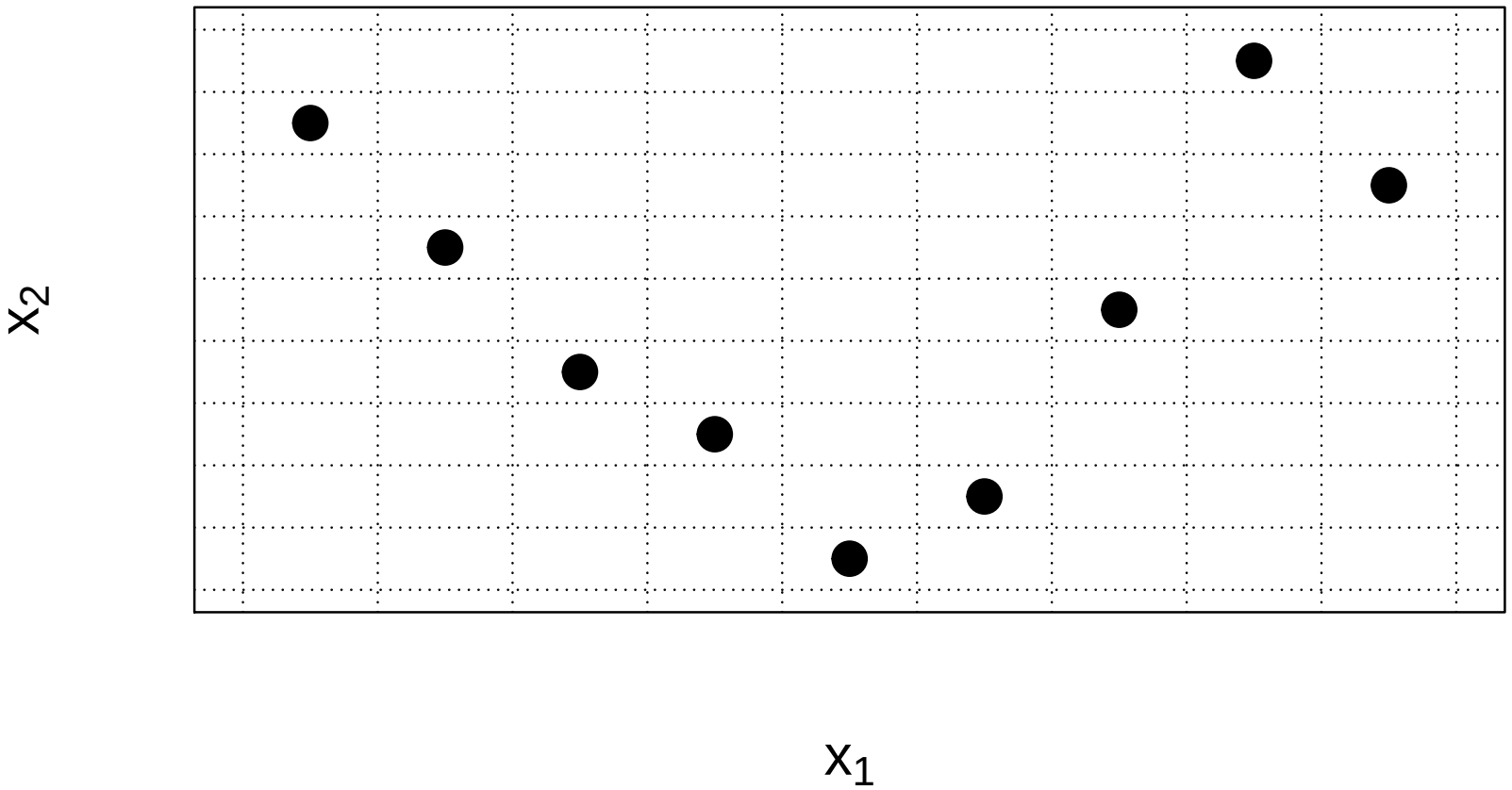
A potentially bad design: How to waste 2/3 of runs!

Only 3 points among 9 are useful if f_{sim} depends only on x_1 (or x_2).



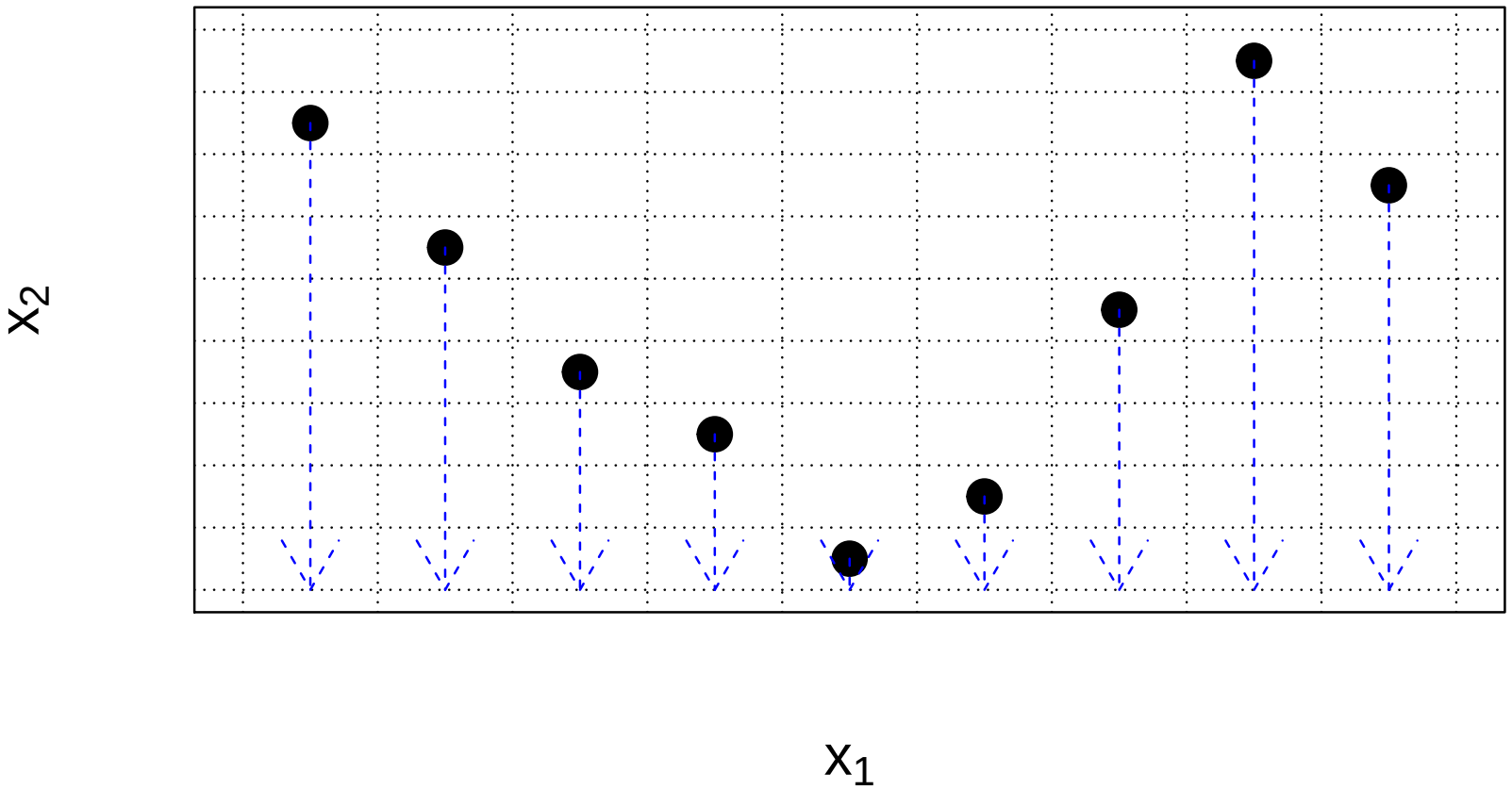
Better designs: Latin hypercubes

```
R> X <- lhsDesign(n = 9, dimension = 2, seed = 3,
randomized = FALSE)$design
```



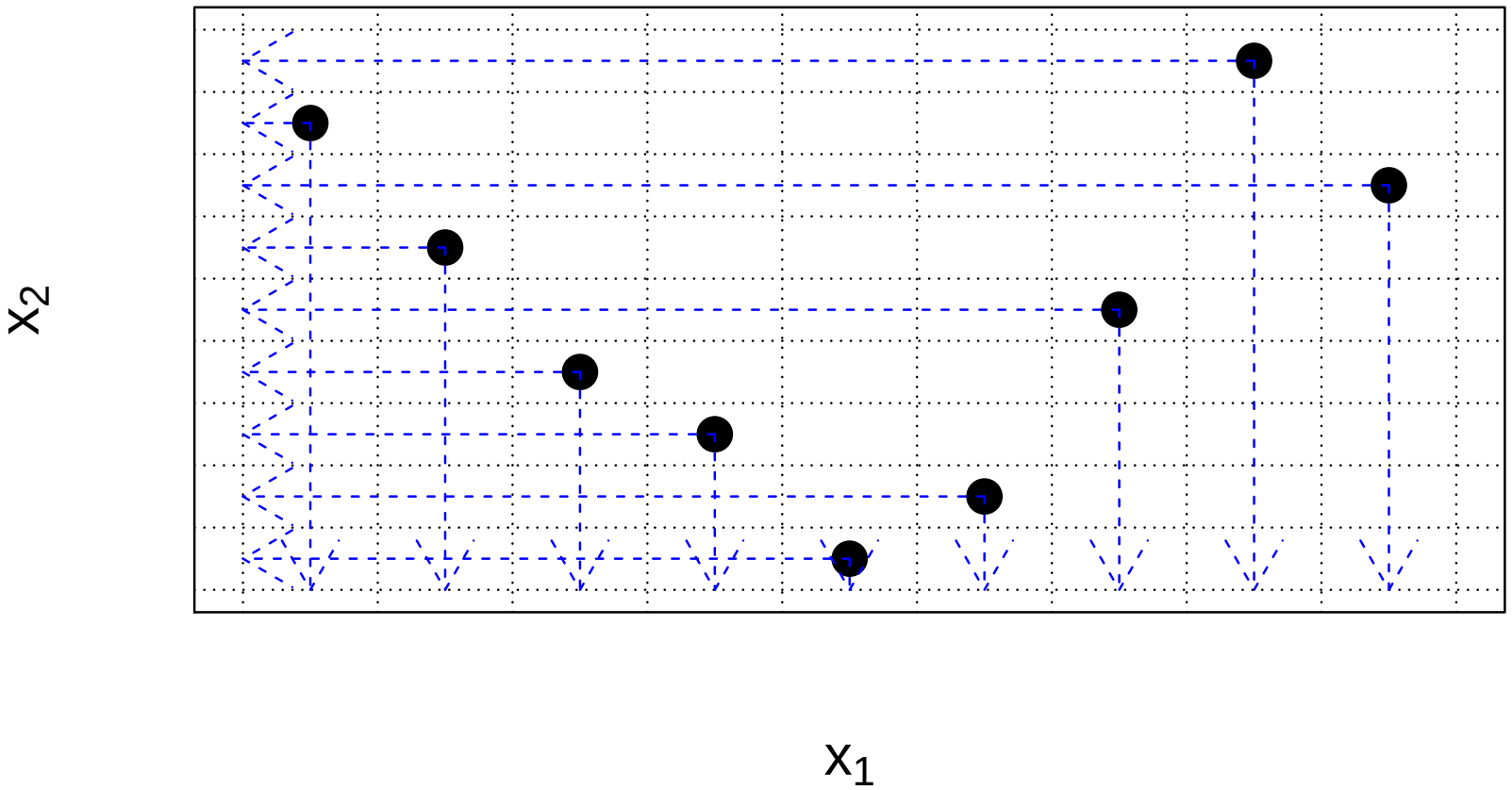
Better designs: Latin hypercubes

```
R> X <- lhsDesign(n = 9, dimension = 2, seed = 3,
randomized = FALSE)$design
```



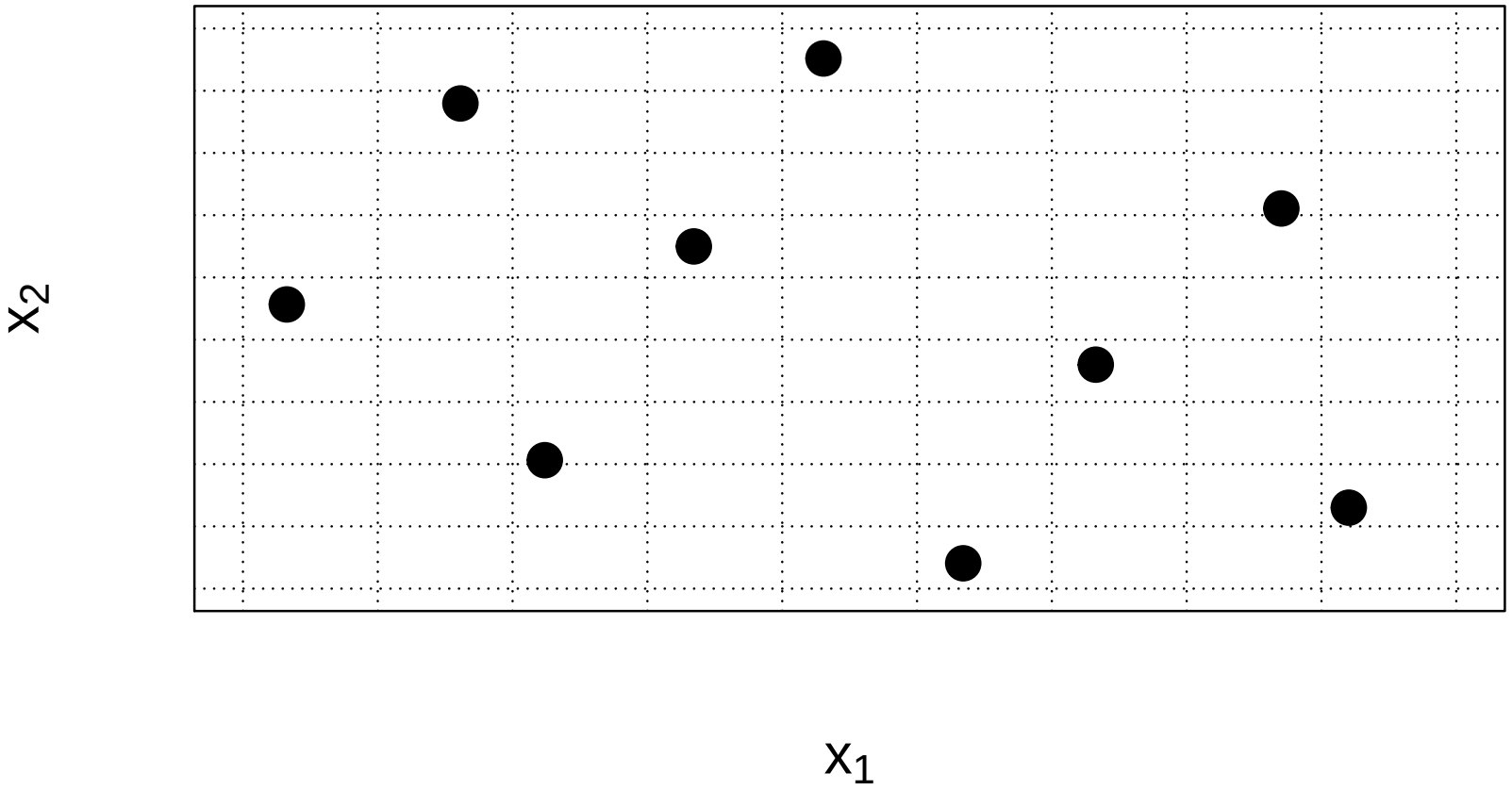
Better designs: Latin hypercubes

```
R> X <- lhsDesign(n = 9, dimension = 2, seed = 3,
randomized = FALSE)$design
```



Good(?) designs: space-filling Latin hypercubes

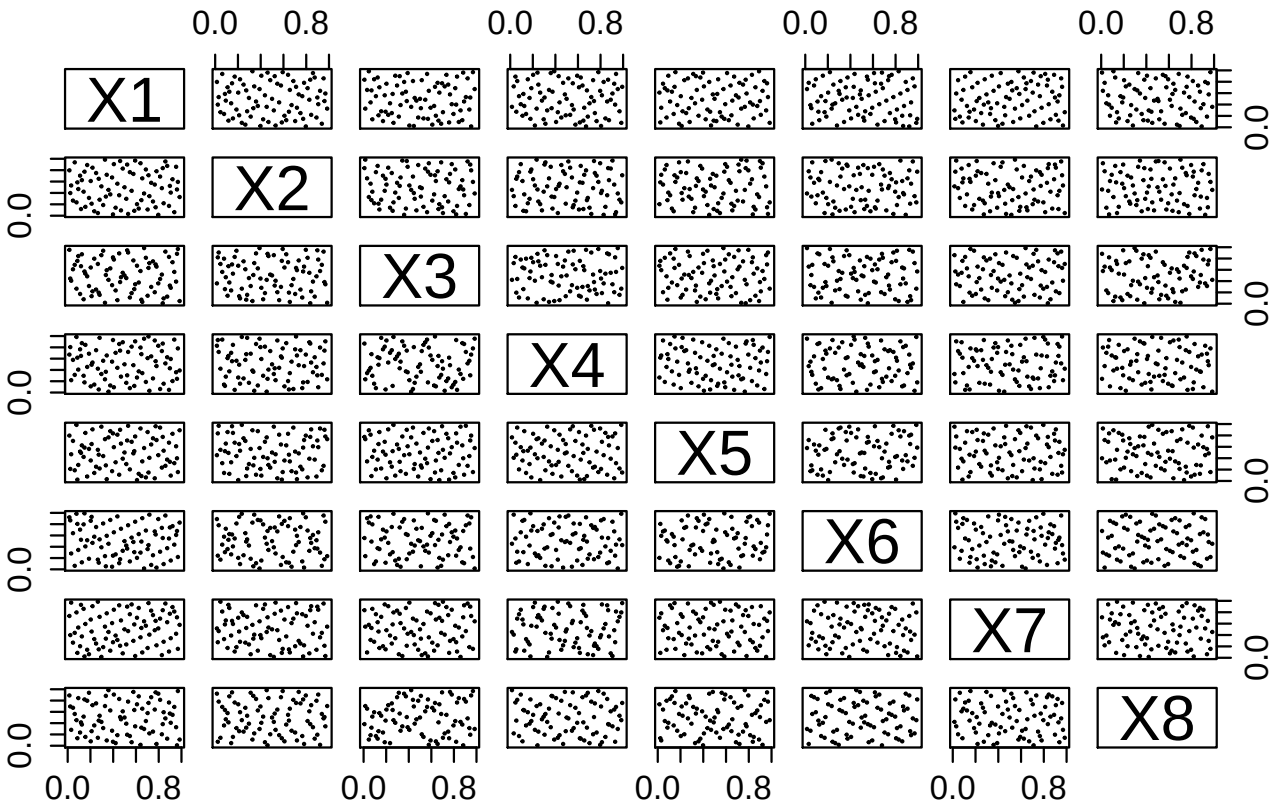
```
R> X0 <- lhsDesign(n = 9, dimension = 2)$design
R> X <- maximinESE_LHS(X0, it=2)$design
```



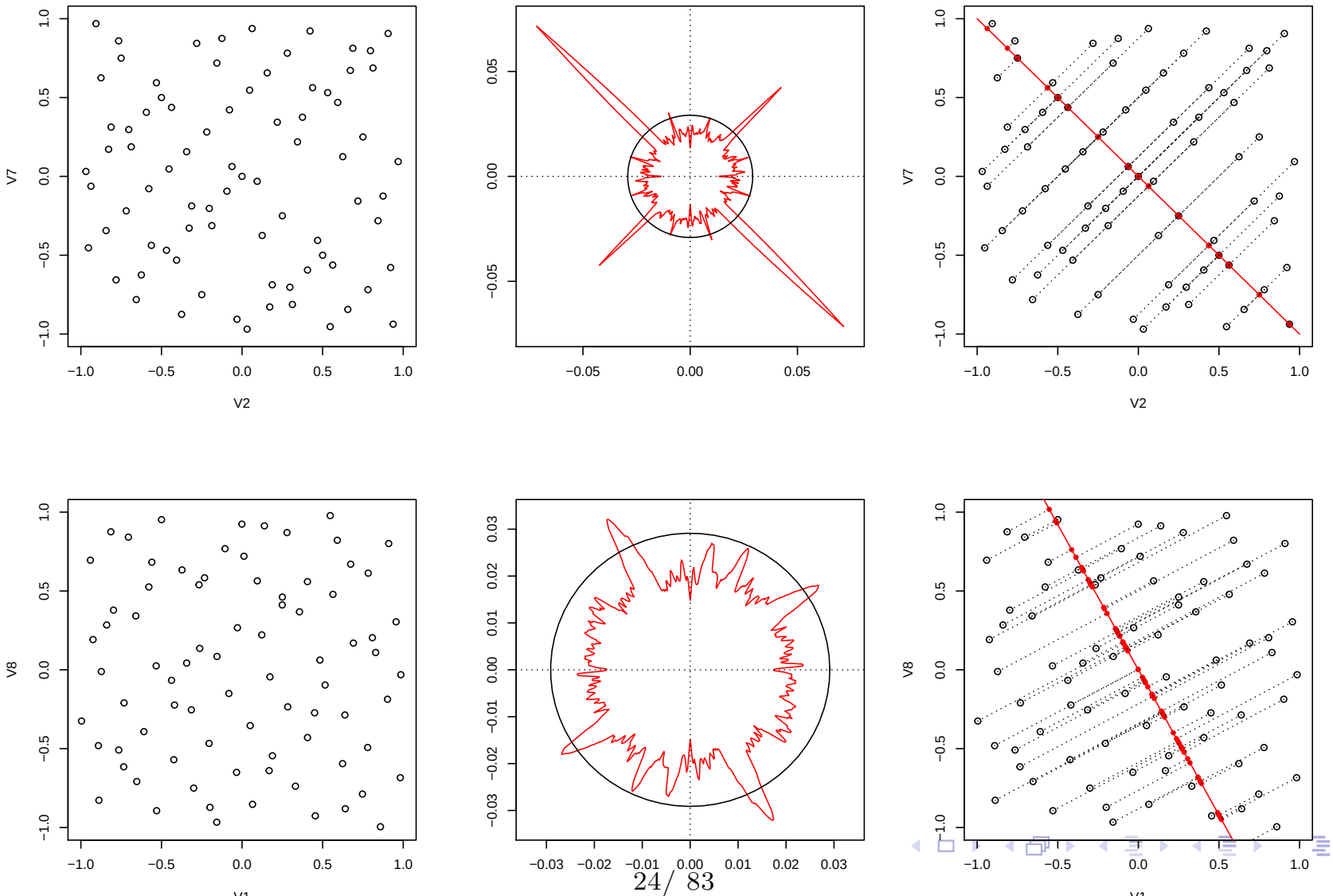
Evaluation of designs with the radial scanning statistic

The radial scanning stat. tests if the projections onto a straight line are compatible with uniformity distribution in the whole domain.

Example with a 80-point low discrepancy (Sobol) sequence, obtained with package **randtoolbox**.



Top: Radial scanning statistic for the Sobol sequence, indicating potential problems if $f_{\text{sim}}(\mathbf{x}) = g(x_2 - x_7)$.
Bottom: The problem disappears with scrambling.



DiceDesign features

1. Construction of space-filling designs
   - Maximin Latin hypercubes
   - Other designs: Strauss, maximum entropy, WSP, ...

   $\longrightarrow$ With a special care on optimization procedures

DiceDesign features

1. Construction of space-filling designs
   - Maximin Latin hypercubes
   - Other designs: Strauss, maximum entropy, WSP, ...

     $\longrightarrow$ With a special care on optimization procedures

2. Evaluation of designs
   - Computation of criteria: distances, discrepancies

     $\longrightarrow$ Discrepancies evaluate the distance to uniformity
   - Graphical tool: radial scanning statistic

     $\longrightarrow$ Detect (oblique) alignments, that may induce a lost of information

Part III

Kriging Models and Gaussian Processes

Kriging models and Gaussian processes for deterministic simulators

A *Kriging model* is a Gaussian process (GP) of the form:

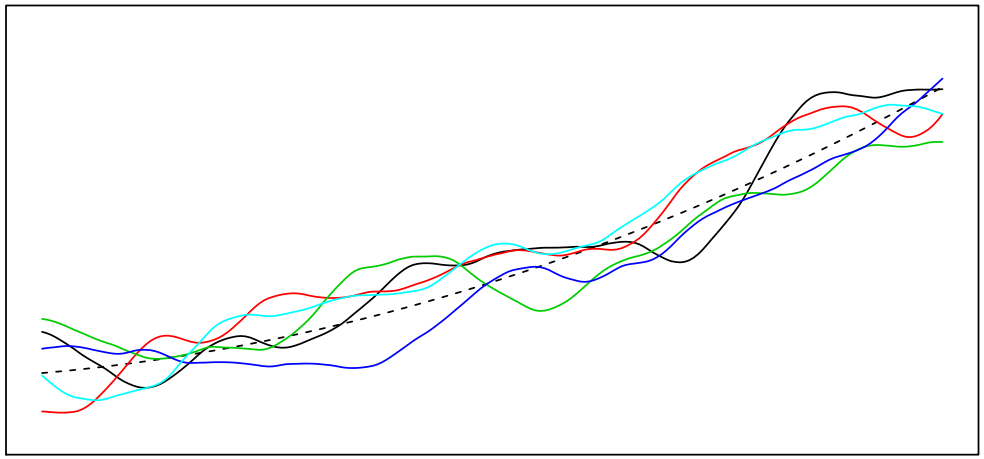
$$Y(\mathbf{x}) = \underbrace{\beta_0 + \beta_1 g_1(\mathbf{x}) + \cdots + \beta_{p-1} g_{p-1}(\mathbf{x})}_{m(\mathbf{x})} + Z(\mathbf{x})$$

where:

- $\mathbf{x}$ is the *location*, typically a vector of length d
- $m(\mathbf{x})$ is a deterministic *trend*, linear comb. of given g_i's
- $Z \sim \mathrm{GP}(0, k)$ where k is the *covariance kernel*,

$$k(\mathbf{x}, \mathbf{x}'; \Theta) = \mathrm{cov}(Z(\mathbf{x}), Z(\mathbf{x}'))$$

$\longrightarrow$ A deterministic simulator is viewed as a particular path of Y



x

Figure: Simulated paths of a Kriging model (Matérn kernel)

Link to geostatistics

The Kriging model originates from geostatistics and the work of D. Krige [Kri51], followed by G. Matheron [Mat63] and many others. The problem dimension d is then 2 or 3. Examples:

- $\mathbf{x}$ is the 3D coordinate in the subsoil, $Y(\mathbf{x})$ is the amount of gold per volume
- $\mathbf{x}$ is the 2D location in France, $Y(\mathbf{x})$ the rainfall

Link to time series

- Linear model with AR(1) errors:

$$Y(t) = \beta_0 + \beta_1 x_{1,t} + \cdots + \beta_{p-1} x_{p-1,t} + Z(t), \quad t = 1, 2, \ldots$$

where $Z(t) = \phi Z(t-1) + \varepsilon(t)$,
with $\varepsilon(t)$ is a Gaussian white noise.

$\longrightarrow$ Here, k is an *exponential* kernel $k(t, t') = \sigma^2 \exp(-|t - t'|/\theta)$

Kriging: An interpolator with prediction bounds!

Prediction with Kriging model is obtained by *Gaussian vector conditioning*. Given:

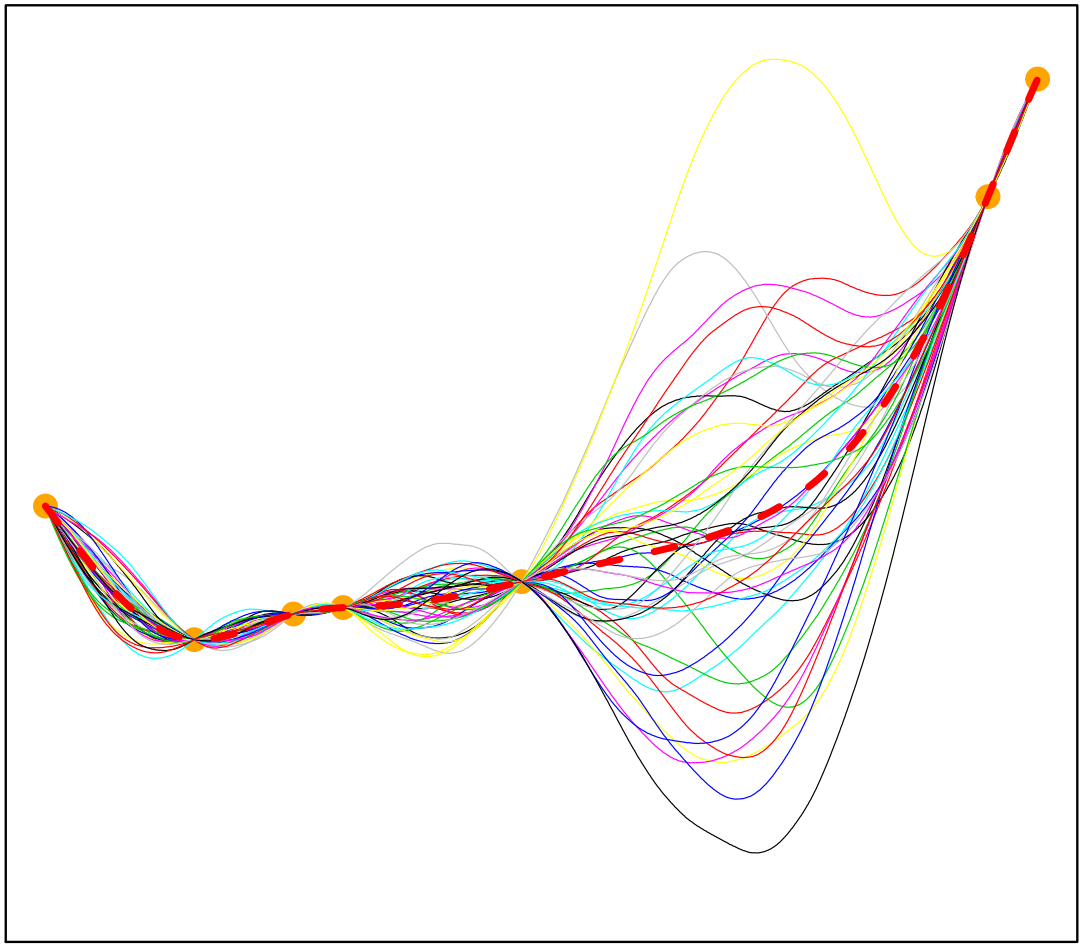
- A set of locations $\mathbf{X} = \mathbf{x}^1, \ldots, \mathbf{x}^n$,
- A set of observations $\mathbf{y} = y^1, \ldots, y^n$, with $y^i = (f_{\text{sim}}(x^i))$.

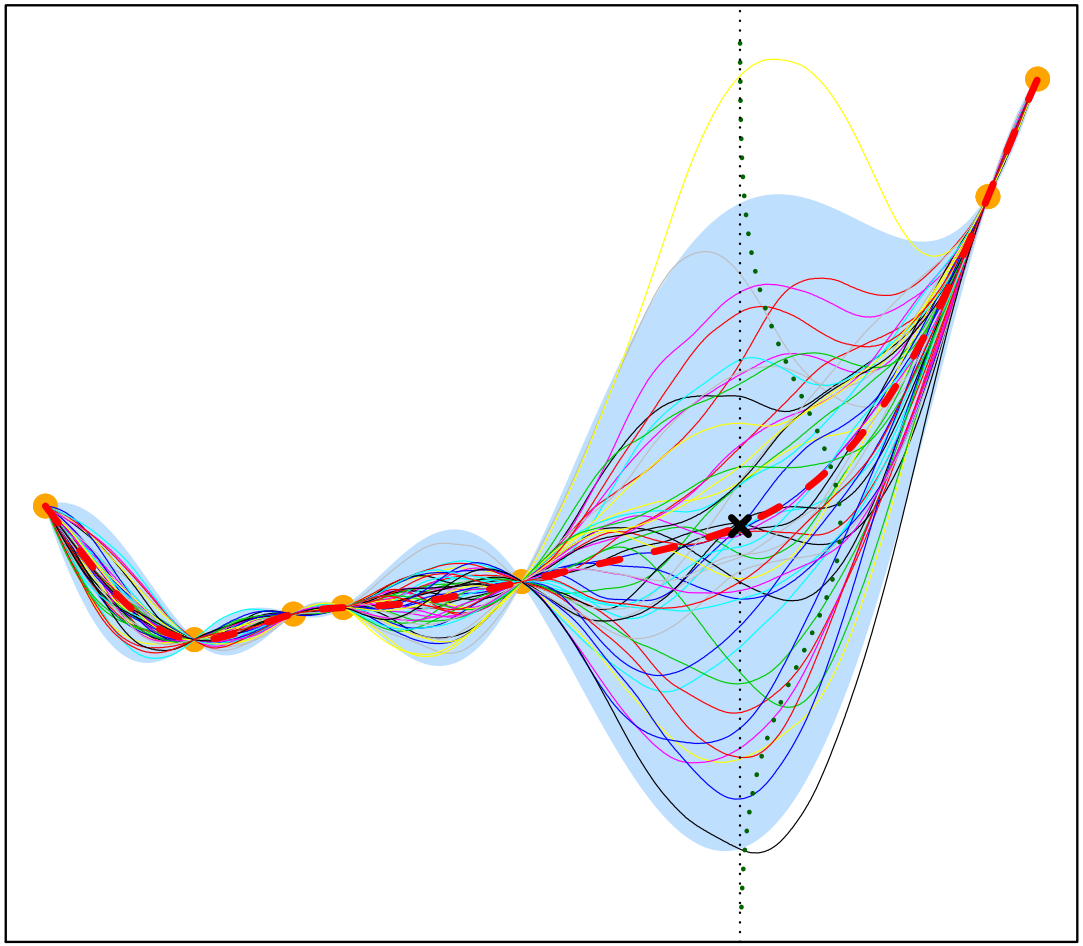
By conditioning on $Y(\mathbf{x}^1) = y^1, \ldots, Y(\mathbf{x}^n) = y^n$, we obtain a GP:

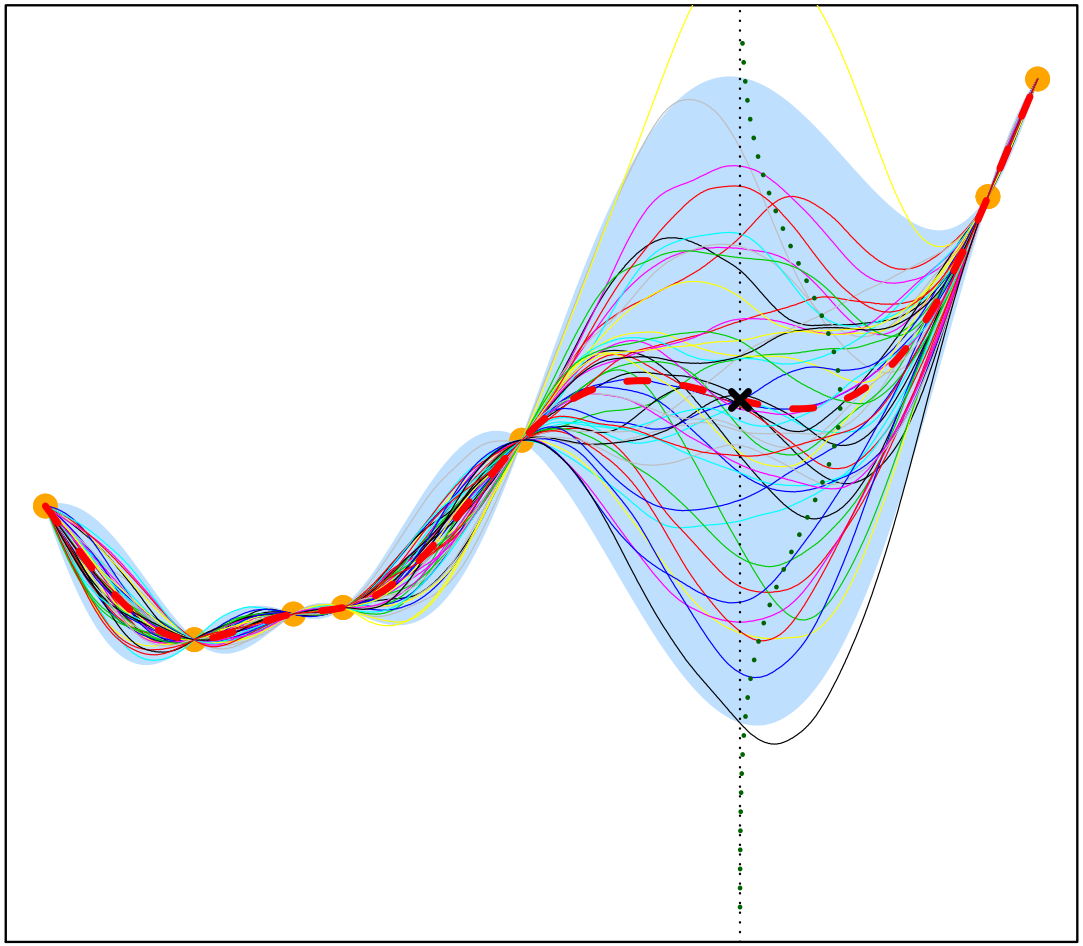
- Whose paths interpolate the observations

  $\longrightarrow$ The mean – *Kriging mean* – is an interpolator
- Whose kernel does not depend on the observations

  $\longrightarrow$ The conditional variance – *Kriging variance* – does not depend on $y^1, \ldots, y^n$







Kernels

GP can be customized by choosing appropriate covariance kernels,

$$k(\mathbf{x}, \mathbf{x}') = \mathrm{cov}(Z(\mathbf{x}), Z(\mathbf{x}'))$$

Below some operations to construct new kernels from old:

- Product: $k_1(\mathbf{x}, \mathbf{x}') \times k_2(\mathbf{x}, \mathbf{x}')$
- Tensor product: $k_1(\mathbf{x}_1, \mathbf{x}'_1) \times k_2(\mathbf{x}_2, \mathbf{x}'_2)$
- Sum, tensor sum
- Change of scale: $k(g(\mathbf{x}), g(\mathbf{x}'))$
- ...

$\rightarrow$ Usual d-dim. kernels are tensor product of 1D kernels

Remark about regularity

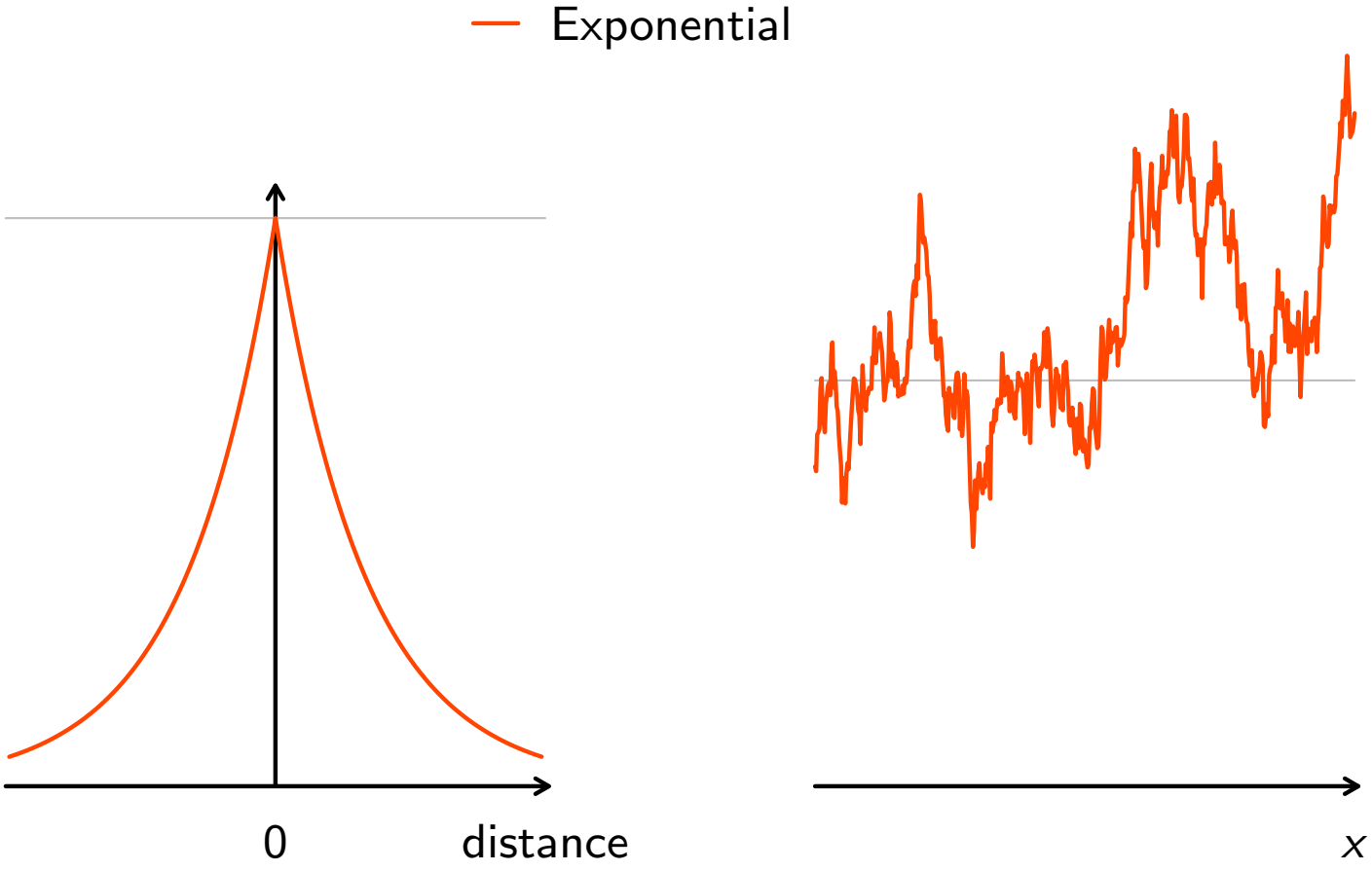


Figure: The regularity of the paths (right) is linked to the regularity of the covariance kernel (left).

Remark about regularity

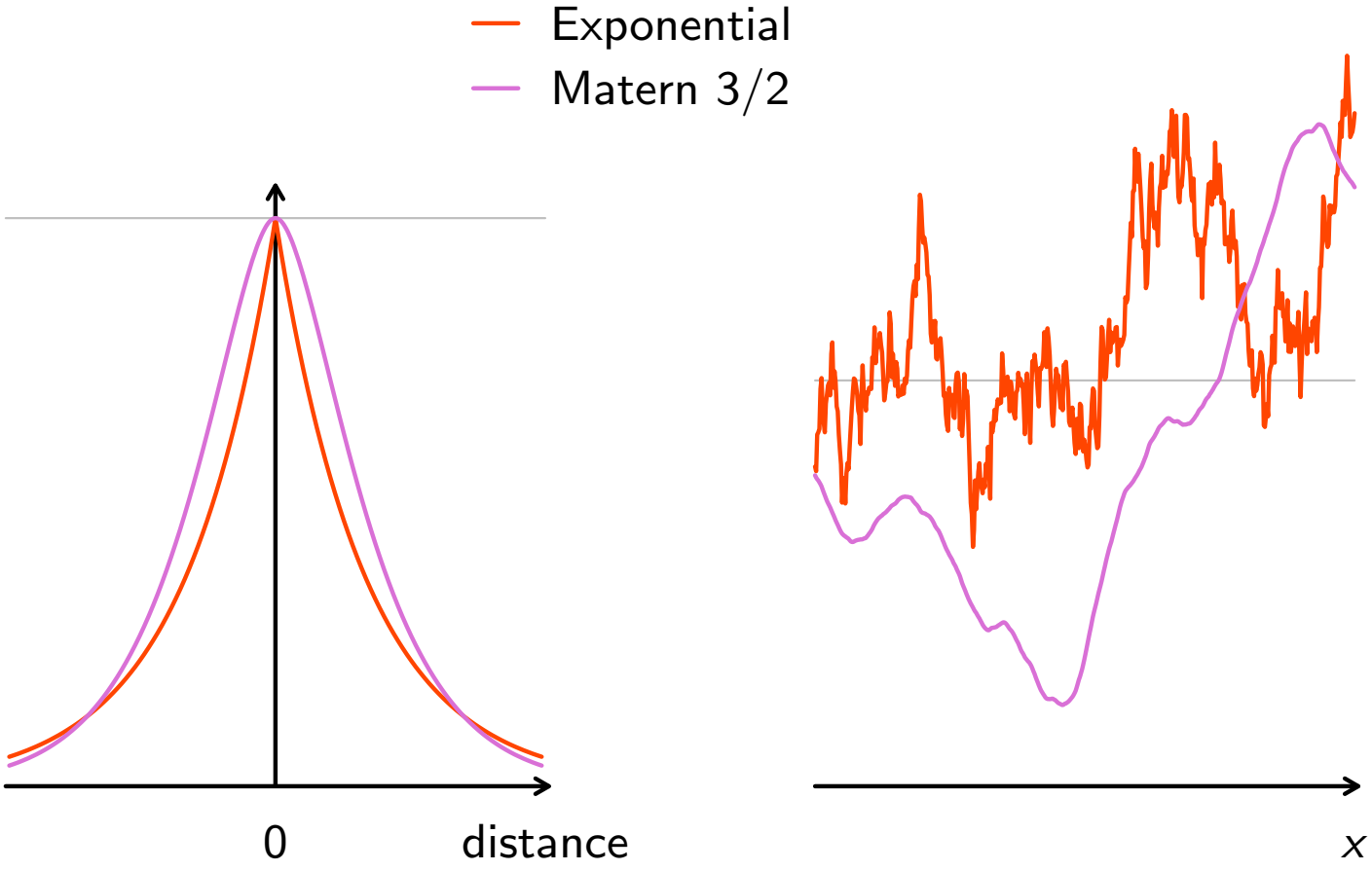


Figure: The regularity of the paths (right) is linked to the regularity of the covariance kernel (left).

Remark about regularity

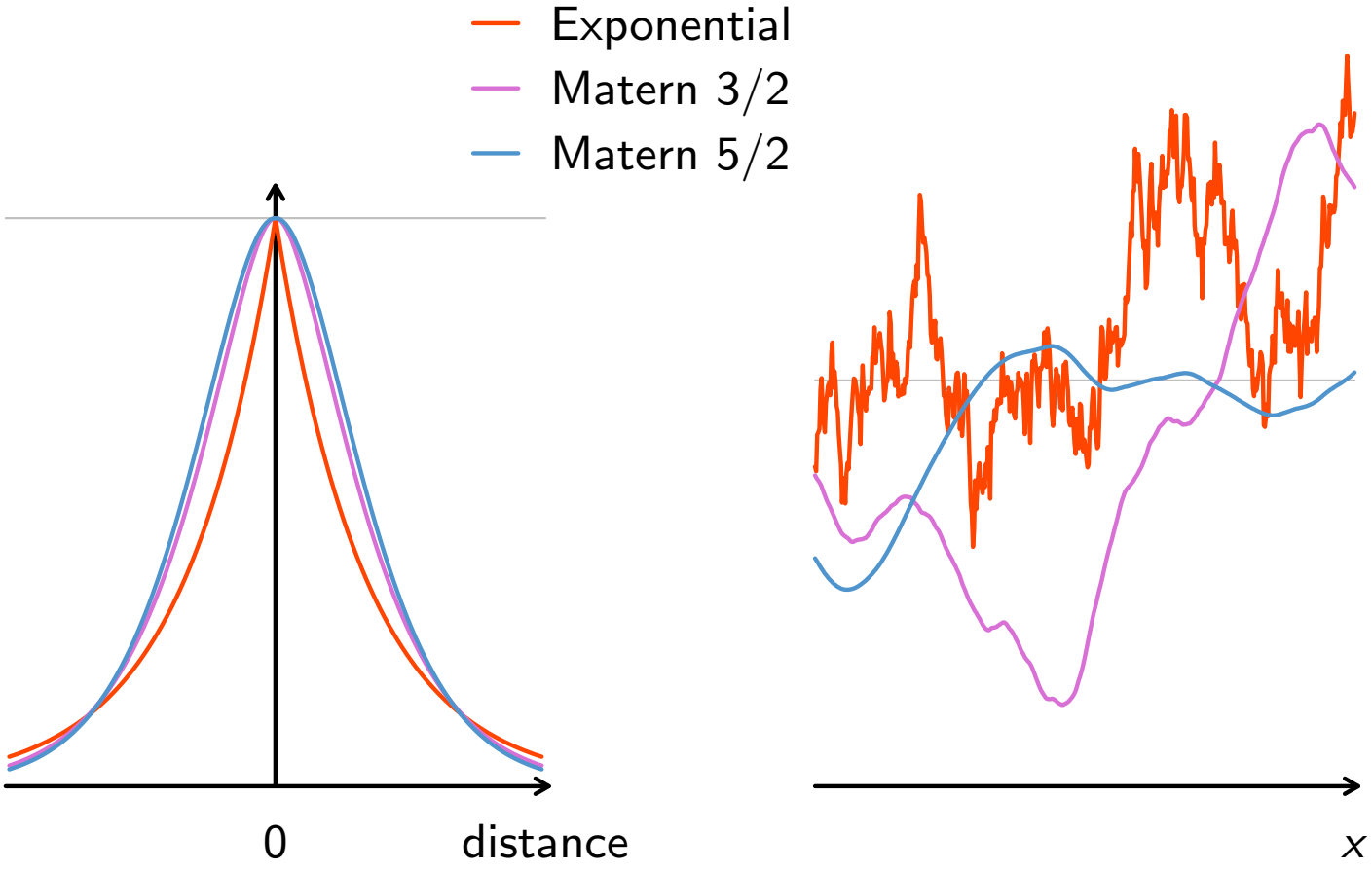


Figure: The regularity of the paths (right) is linked to the regularity of the covariance kernel (left).

Remark about regularity

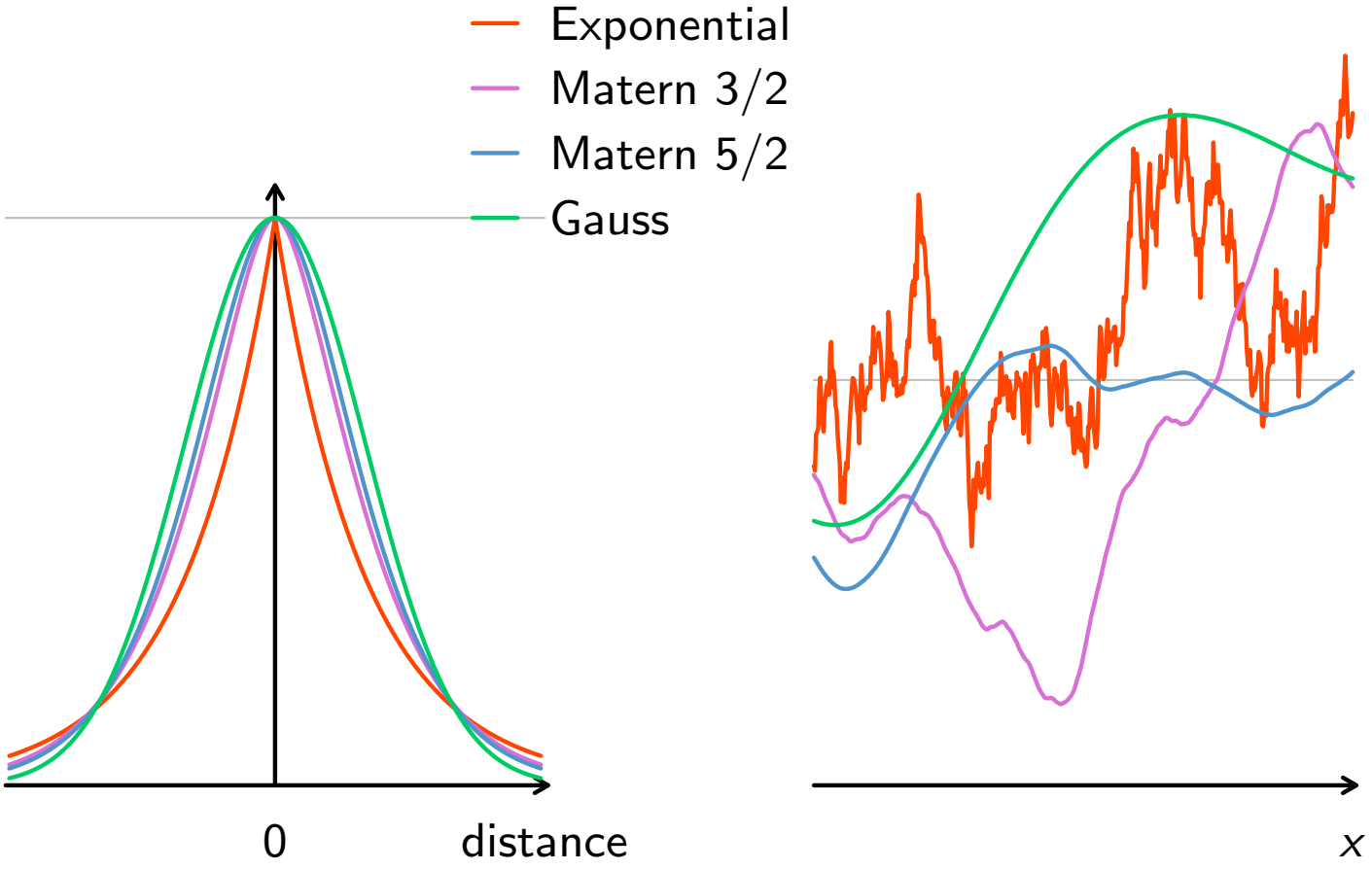


Figure: The regularity of the paths (right) is linked to the regularity of the covariance kernel (left).

Extension to noisy observations

In various situations (ex: Neutronics), the simulator is stochastic.

- The Kriging model has to be adapted:

$$\zeta_i = Y(\mathbf{x}^i) + \varepsilon_i, \qquad i = 1, \ldots, n$$

  where Y is a GP and ε_i are ind. Gaussian r.v. $N(0, \sigma_i^2)$.
  $\longrightarrow$ We can have several observations at the same location

- The aim is to predict $Y(\mathbf{x})$ conditionaly on $\zeta_1, \ldots, \zeta_n$
  $\longrightarrow$ A filtering problem, also solved by Gaussian conditioning, which only slightly modifies the Kriging formulas
- A.k.a. *Gaussian Process Regression*.

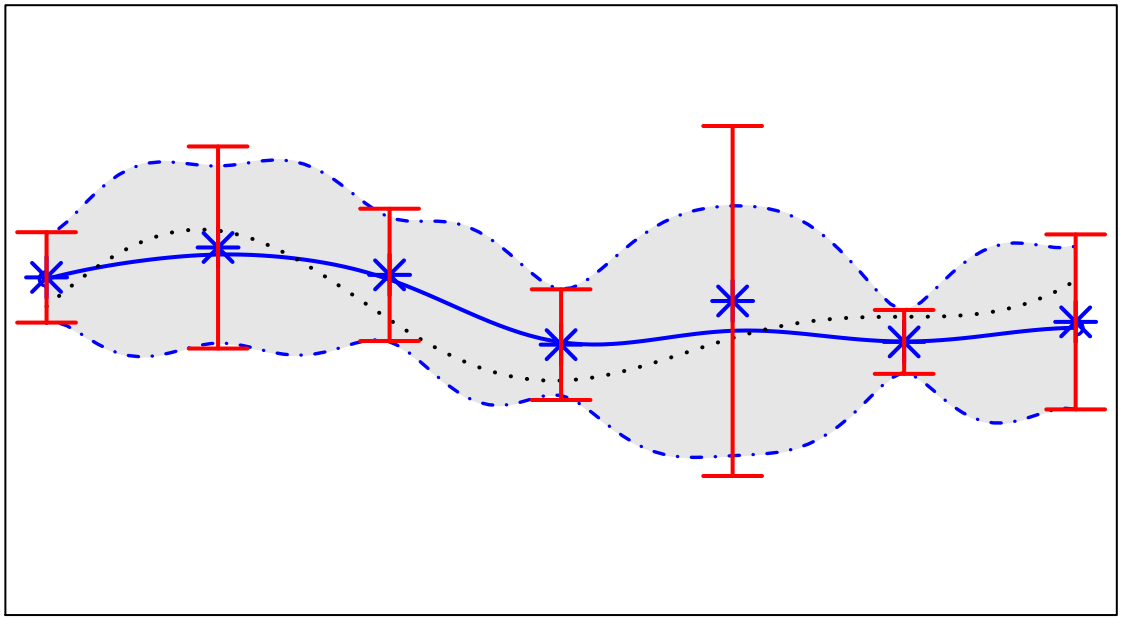


Figure: The noisy observations (stars) are obtained from the true function $Y(\mathbf{x})$ (dotted line) by adding an heterogeneous noise (vertical bars). Confidence bounds in shaded grey.
The Kriging mean is no more an interpolator.

Parameter estimation

The trend and covariance parameters are often unknown.

- Estimation can be done numerically by MLE or by Cross-Validation (CV)

  $\longrightarrow$ The analytical gradient of the criteria is supplied to the optimizers, enhancing their efficiency

Parameter estimation

The trend and covariance parameters are often unknown.

- Estimation can be done numerically by MLE or by Cross-Validation (CV)

  $\longrightarrow$ The analytical gradient of the criteria is supplied to the optimizers, enhancing their efficiency

- The complexity for computing the criteria is $O(n^3)$

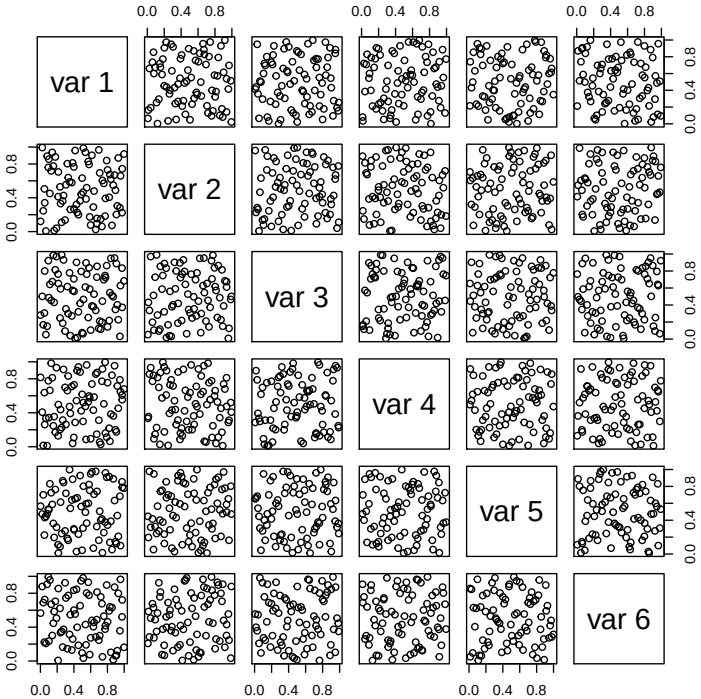
  $\longrightarrow$ A problem when n is large, but often not in our context of time-consuming simulators.

A 6-dimensional (toy) example

Let us choose the Hartman function, transf. by $y \mapsto -\log(-y)$.

1. Choose a 80-point maximin LHS design.

```
R> X0 <- lhsDesign(n = 80, dimension = 6)$design
R> X <- maximinSA_LHS(X0)$design
```



2. Fit a Kriging model to the observations y

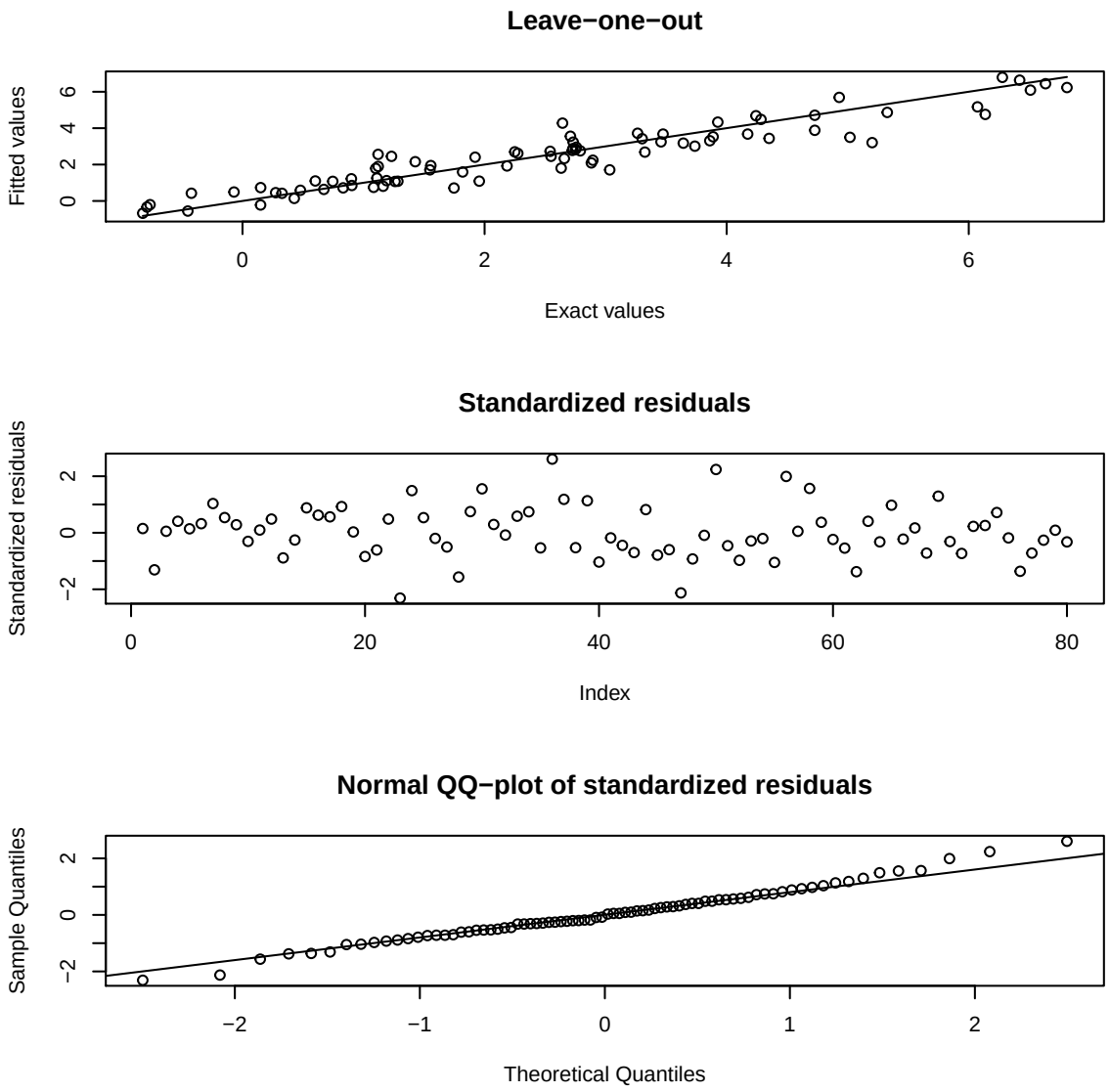
```
R> m <- km(design = data.frame(X), response = y,
+           control = list(trace=F))
```

$\longrightarrow$ we use the default formula

```
R> m

Call:
km(design = data.frame(X), response = y, control = list(trace = F))

Trend  coeff.:
               Estimate
 (Intercept)     4.4372

Covar. type  : matern5_2
Covar. coeff.:
               Estimate
   theta(X1)     0.6640
   theta(X2)     0.9639
   theta(X3)     1.9630
   theta(X4)     0.7540
   theta(X5)     0.6893
   theta(X6)     0.5600

Variance estimate: 4.812747
```

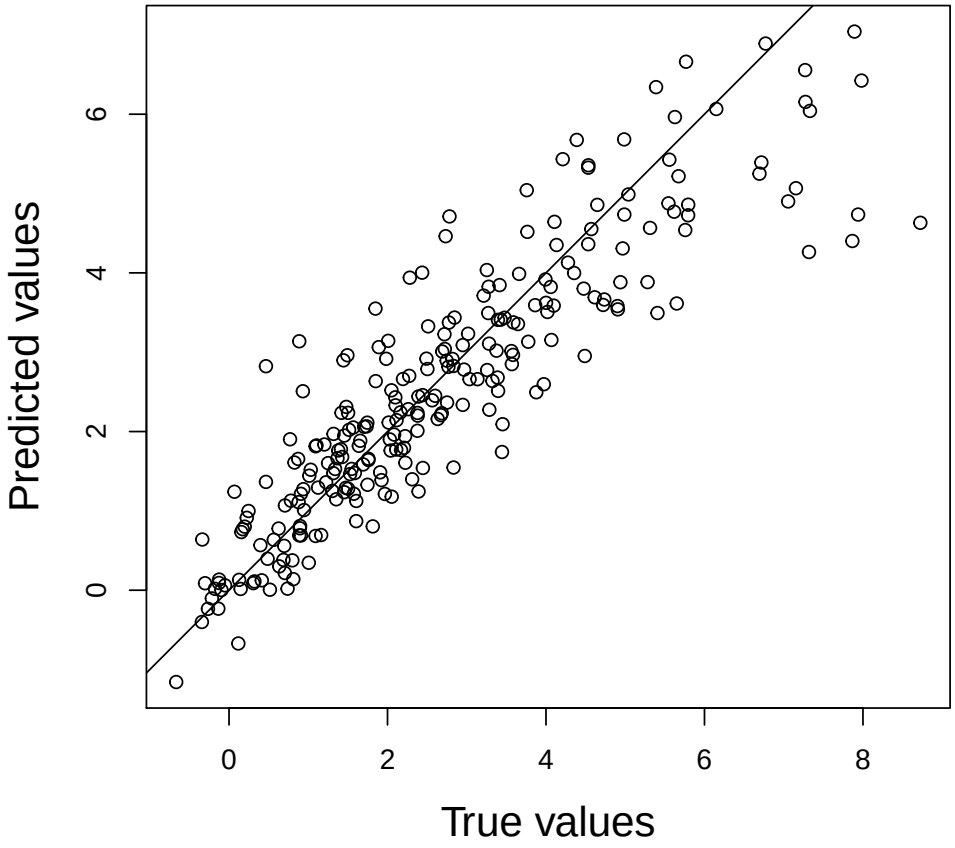
3 Validation with method `plot`

```
R> plot(m)
```



4. Predict at new locations, here a 250-point random Latin hypercube, with method `predict`. Compare with true values.

```
R> y.pred <- predict(m, newdata = X.test, type = "UK")
```



DiceKriging features

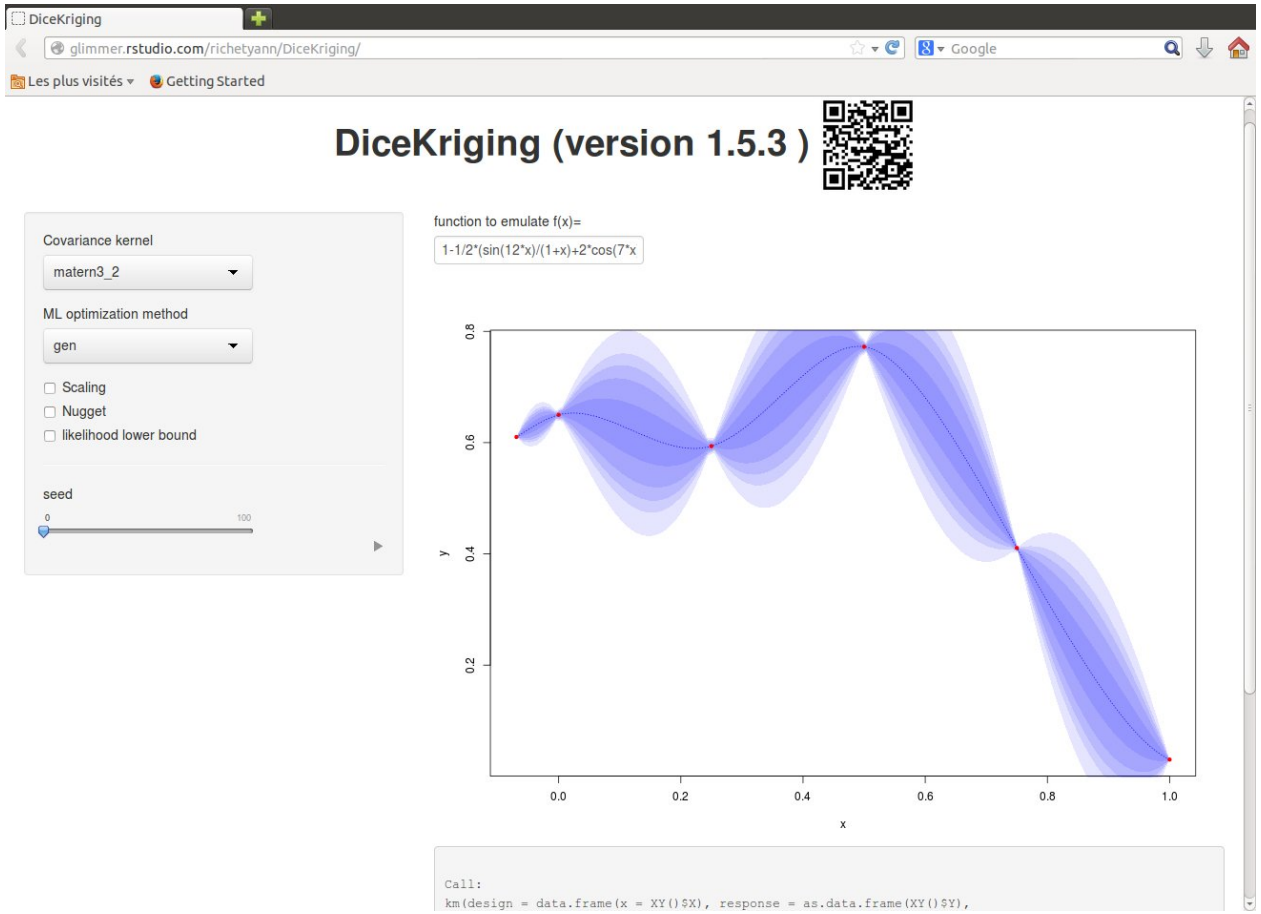
- For both deterministic or noisy observations
- Accepts a general linear trend

  $\longrightarrow$ With the `formula` mechanism
- Various classes of kernels already implemented

  $\longrightarrow$ Possibility to add one's kernel (no parameter estimation)
- Parameter optimization: MLE or CV, with known trend or known covariance, classic or genetic optimization, choice of control parameters, ...
- Methods: `simulate`, `predict`, `plot`

DiceKriging: a shiny demo

Yann Richet has written a shiny animation showing a `km` fit, see it at http://glimmer.rstudio.com/richetyann/DiceKriging.



Part IV

Kriging-based Optimization

Metamodel-based optimization

The aim is to minimize f_{sim}, with the help of a Kriging model Y.

- Wrong way: To minimize the Kriging mean.
  $\longrightarrow$ Highly depends on the quality of the first interpolation!
- Right way: To use both the Kriging mean & variance
  $\longrightarrow$ Results in efficient sequential strategies

Expected Improvement EI

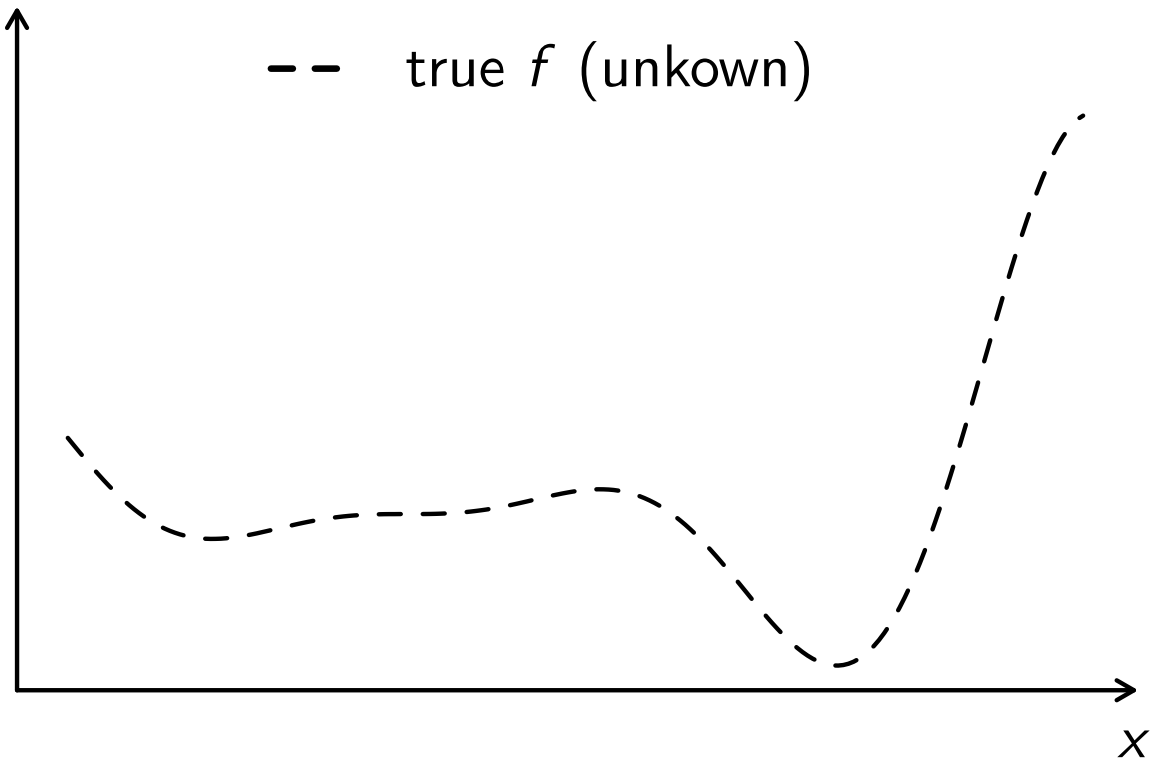
- **Improvement**: What is below the current minimum $f_{\min}$

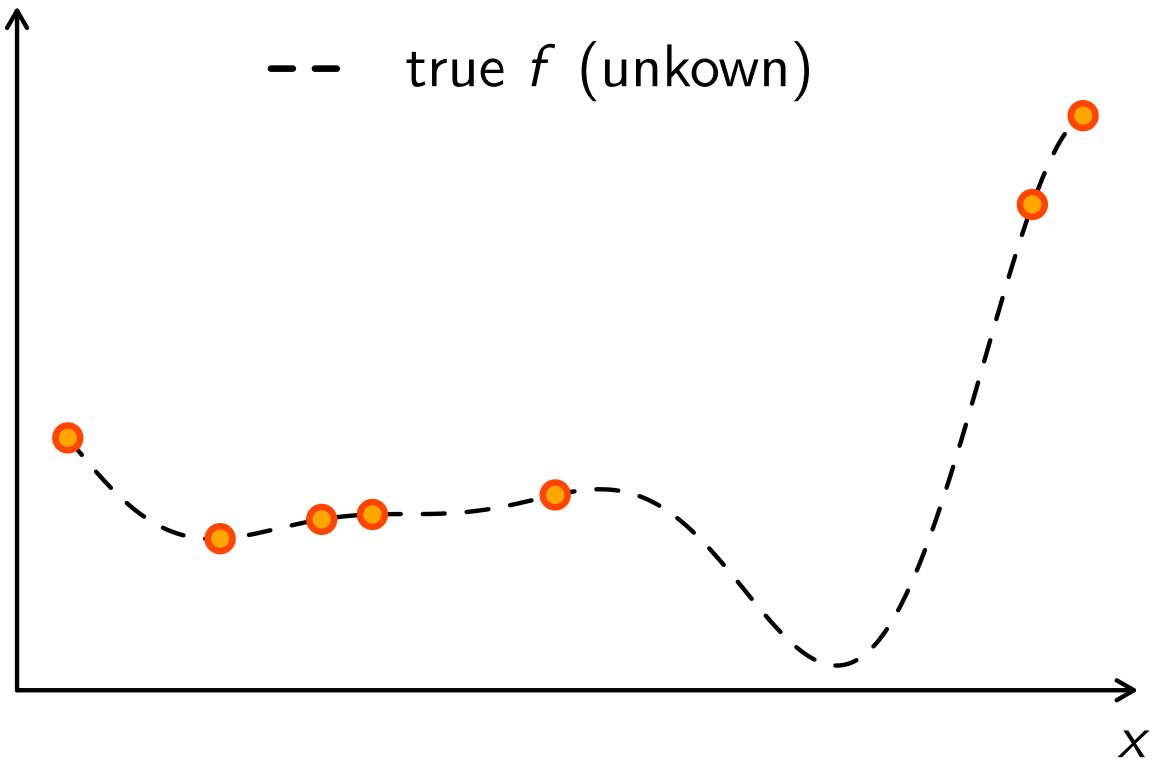
$$I(\mathbf{x}) := \max(f_{\min} - Y(\mathbf{x}), 0)$$

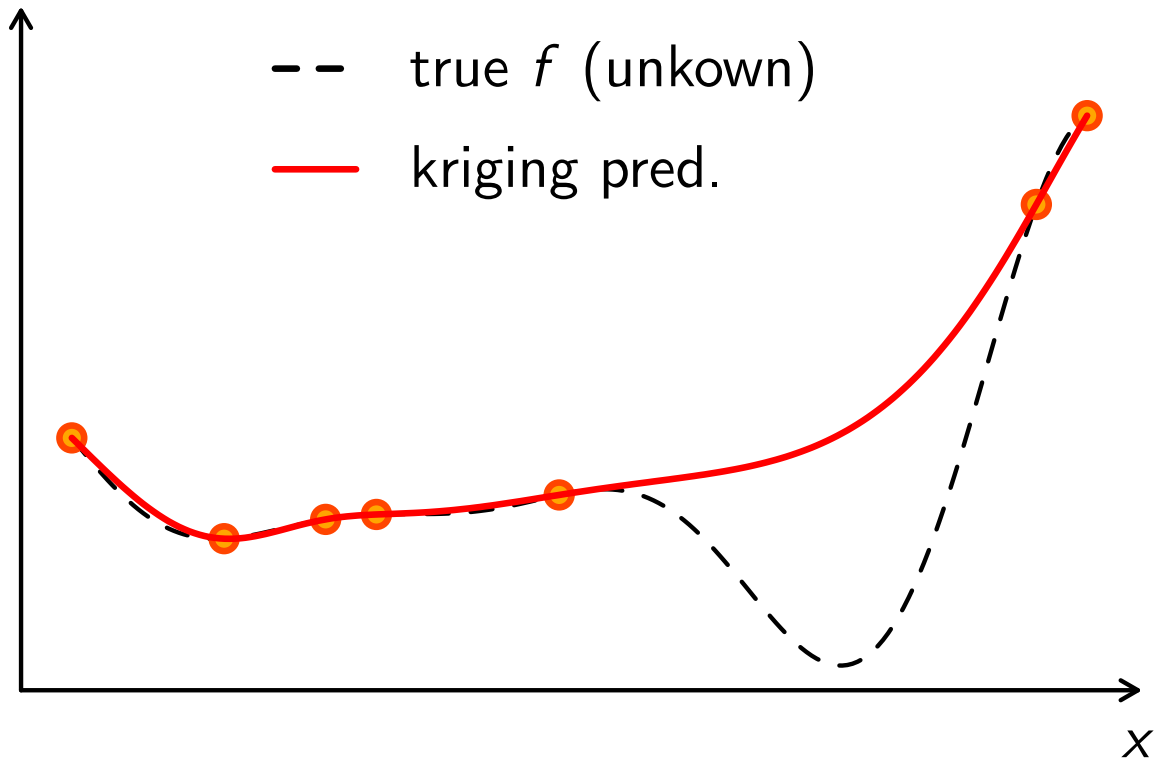
- **Expected Improvement**: Expectation of the r.v. $I(\mathbf{x})$, conditionaly on the observations $\mathbf{y}$ at $\mathbf{X} = (\mathbf{x}^1, \ldots, \mathbf{x}^n)$:

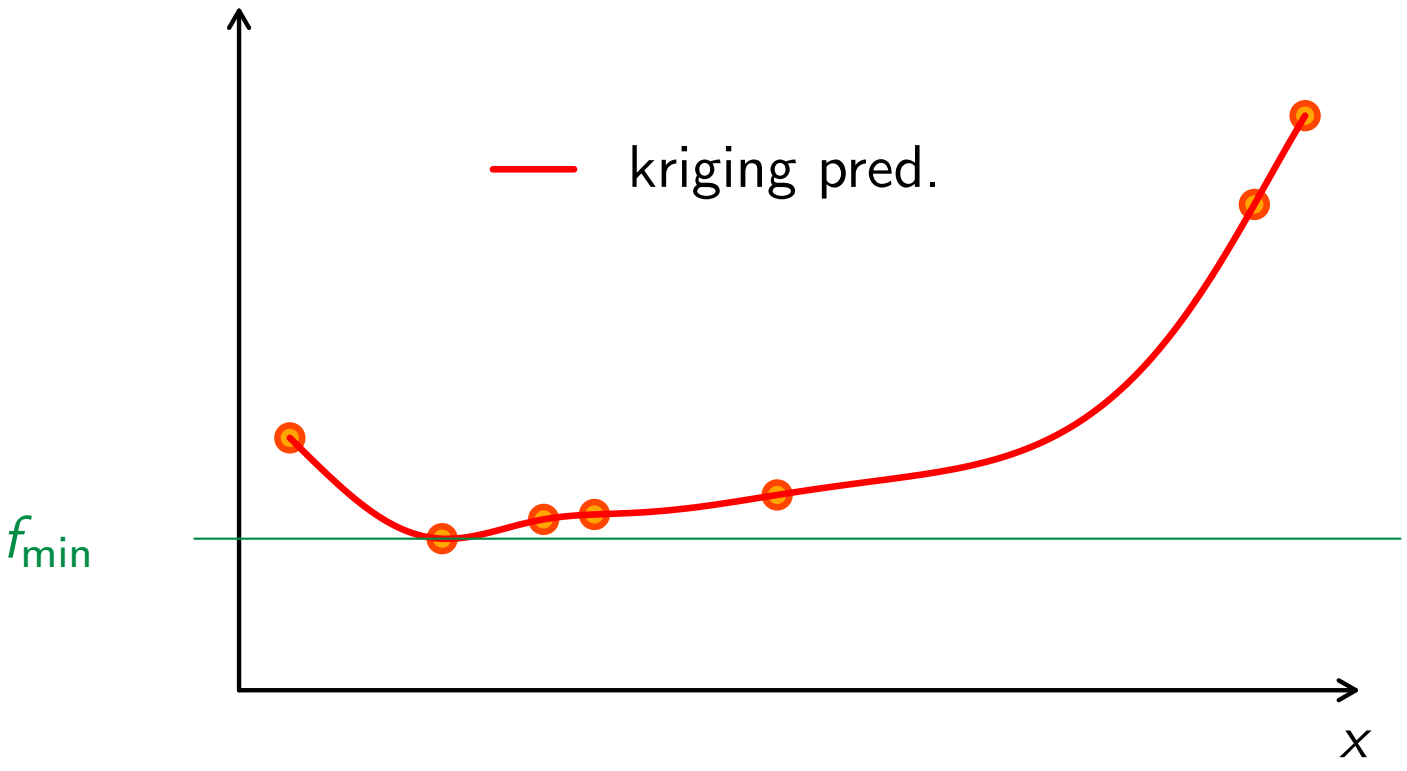
$$\mathrm{EI}(\mathbf{x}) := \mathbb{E}\left[I(\mathbf{x}) | Y(\mathbf{x}^1) = y^1, \ldots, Y(\mathbf{x}^n) = y^n\right]$$

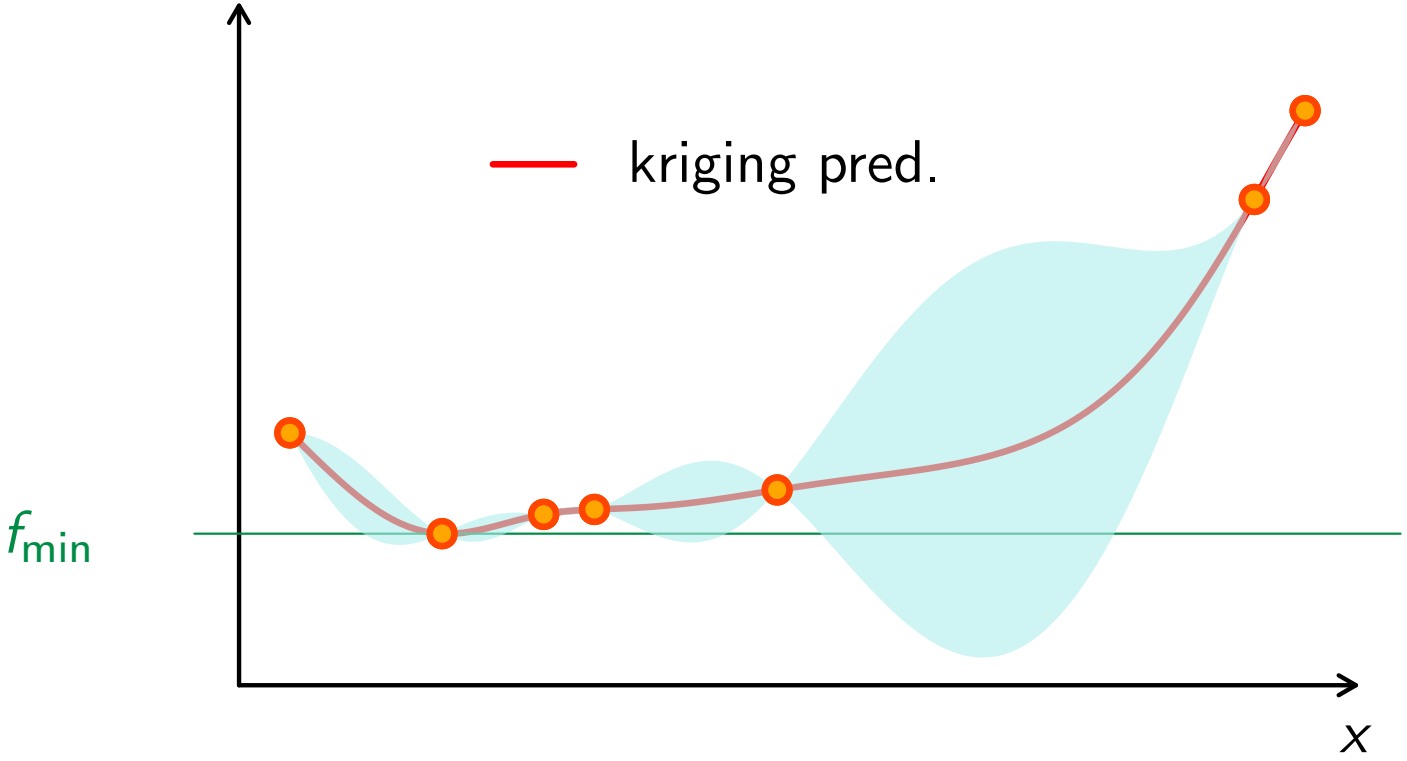
$\rightarrow$ $\mathrm{EI}(\mathbf{x})$ has an analytical expression depending on the Kriging mean and Kriging variance

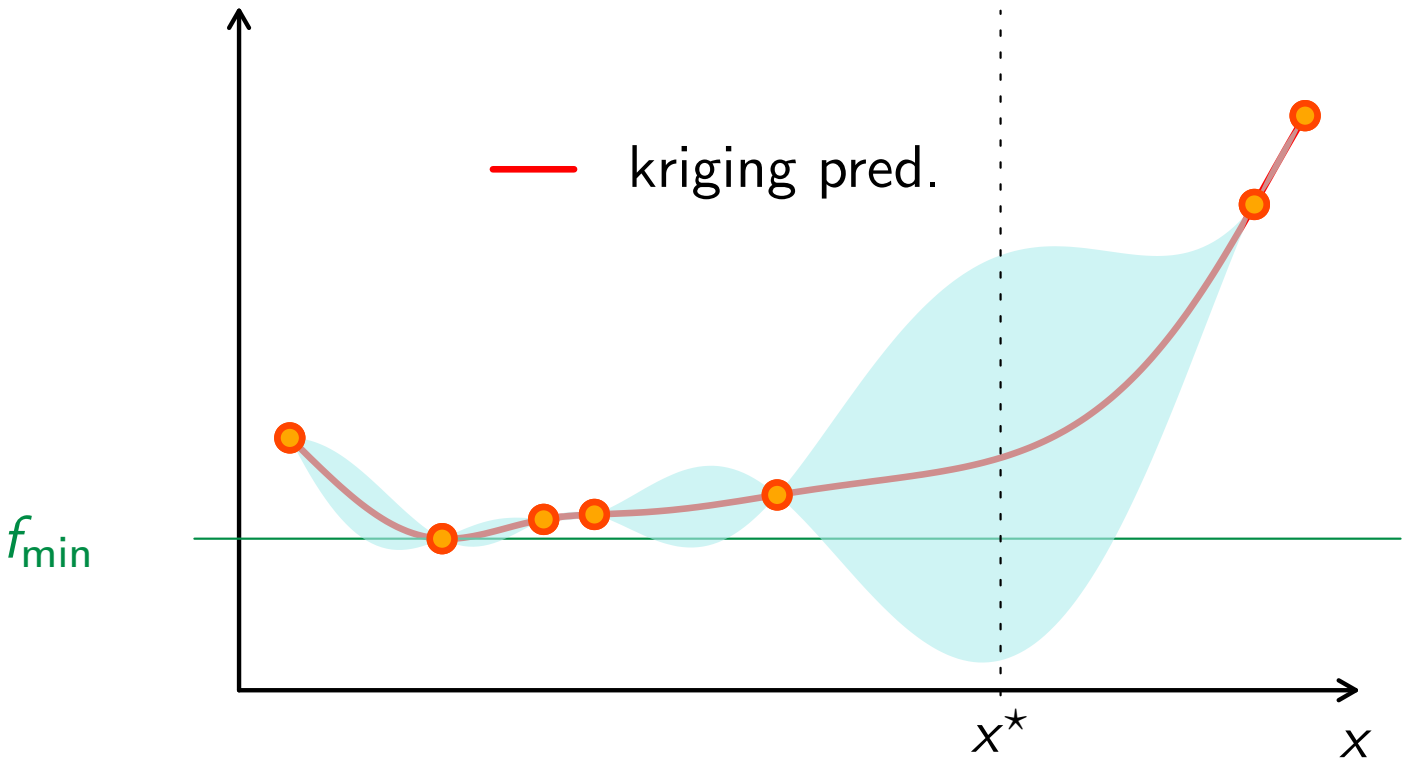

true f (unkown)
x

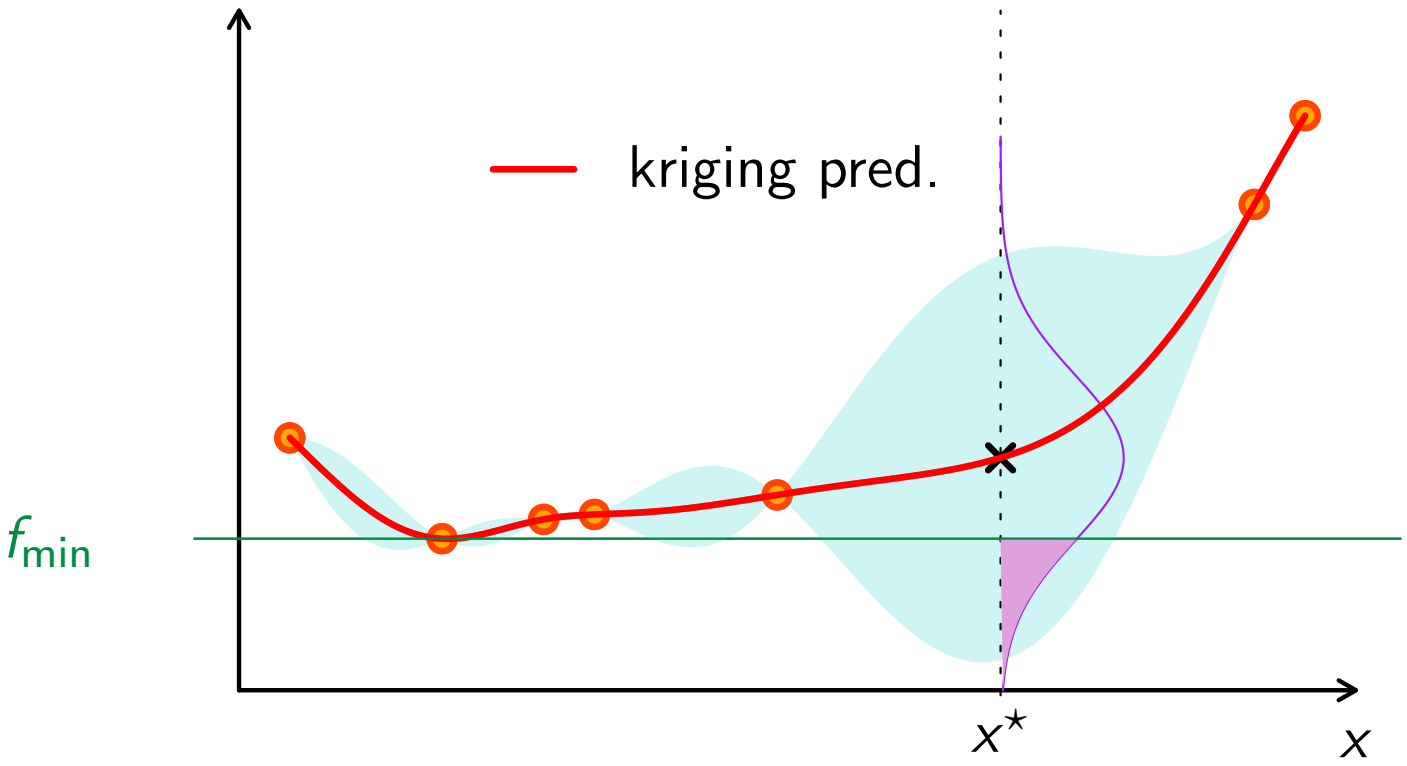

true f (unkown)
x


true f (unkown)
kriging pred.
x


kriging pred.
$f_{\min}$
x


kriging pred.
$f_{\min}$
x


kriging pred.
$f_{\min}$
$x^\star$
x

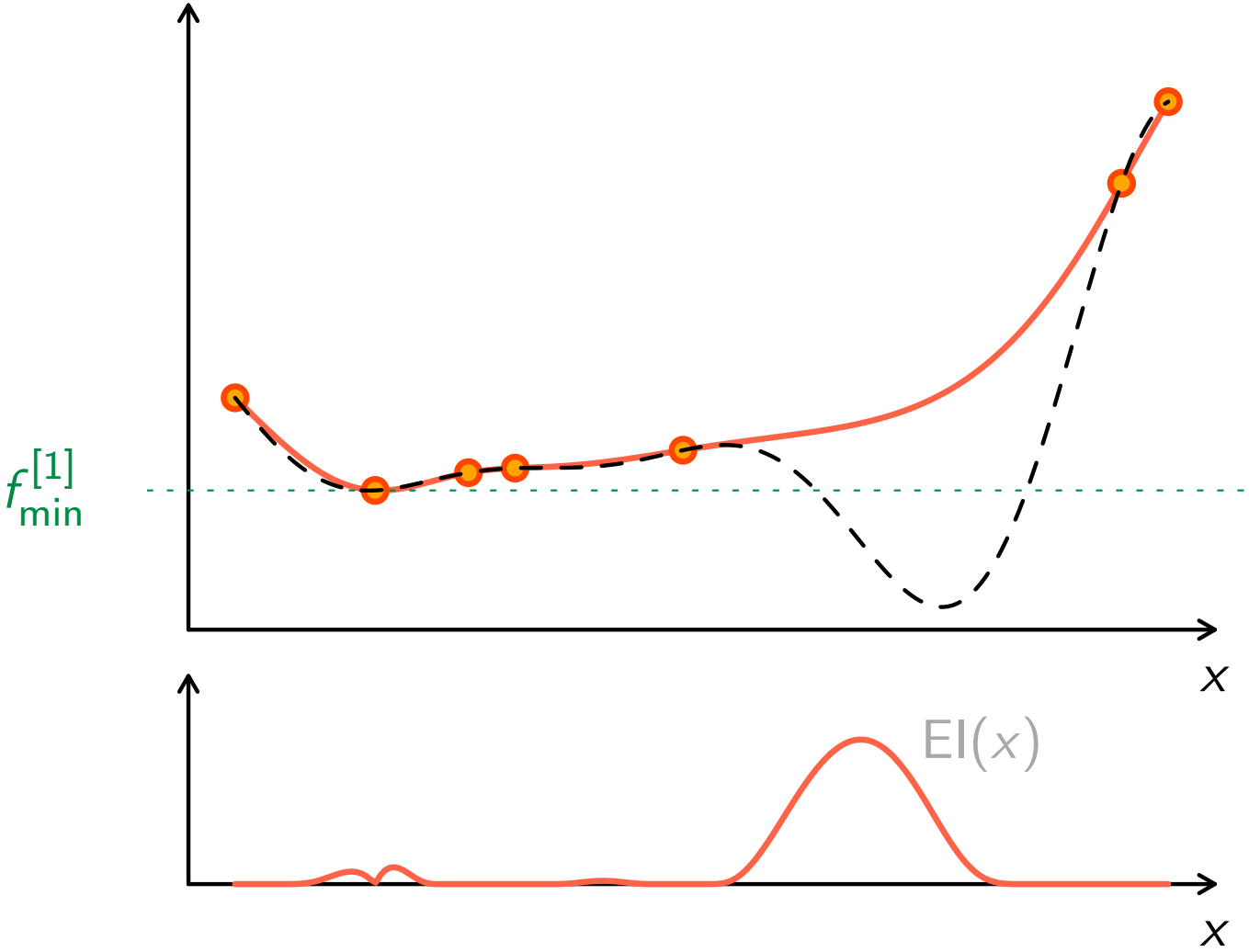

kriging pred.
$f_{\min}$
$x^\star$
x

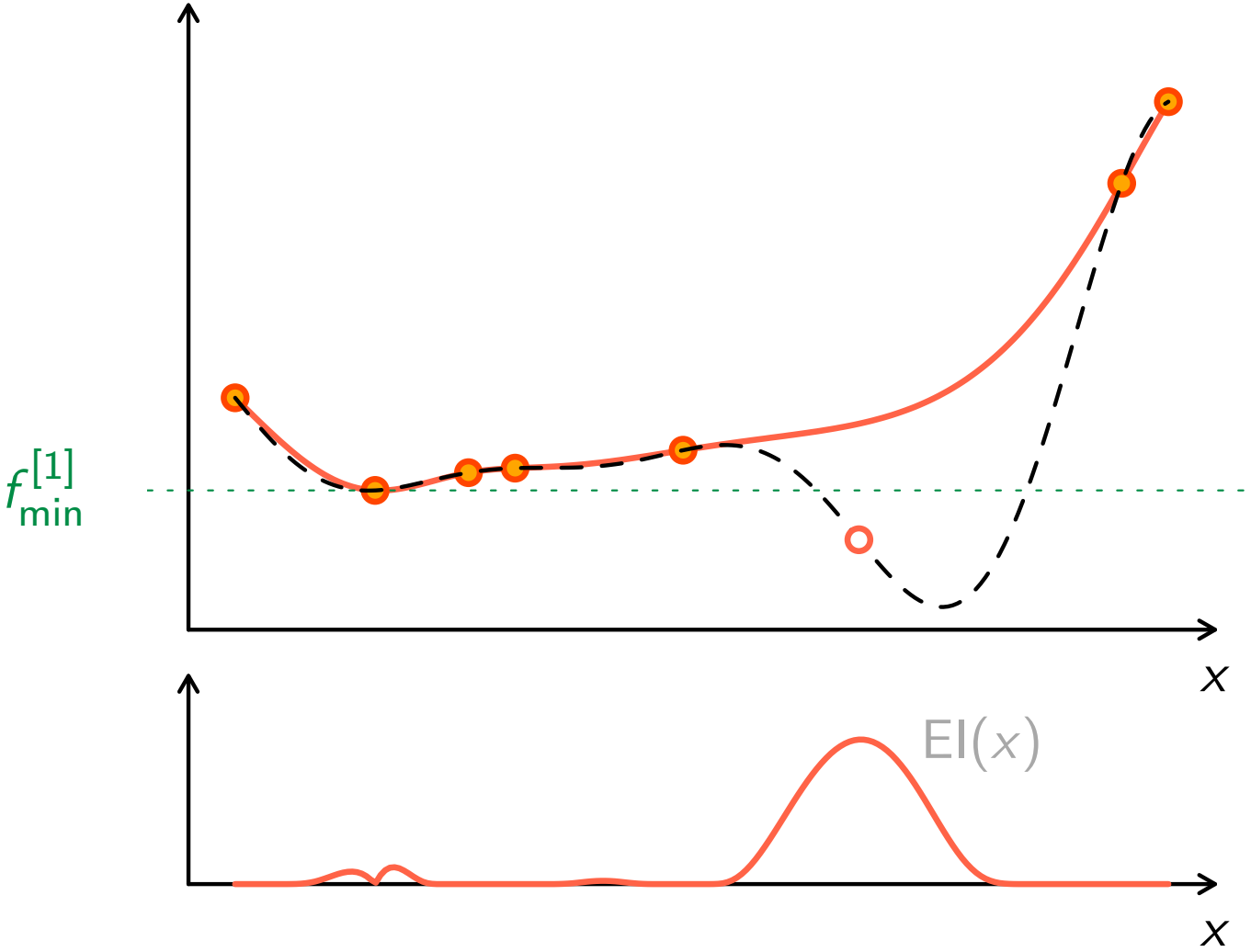
EGO: EI-based sequential optimization strategy

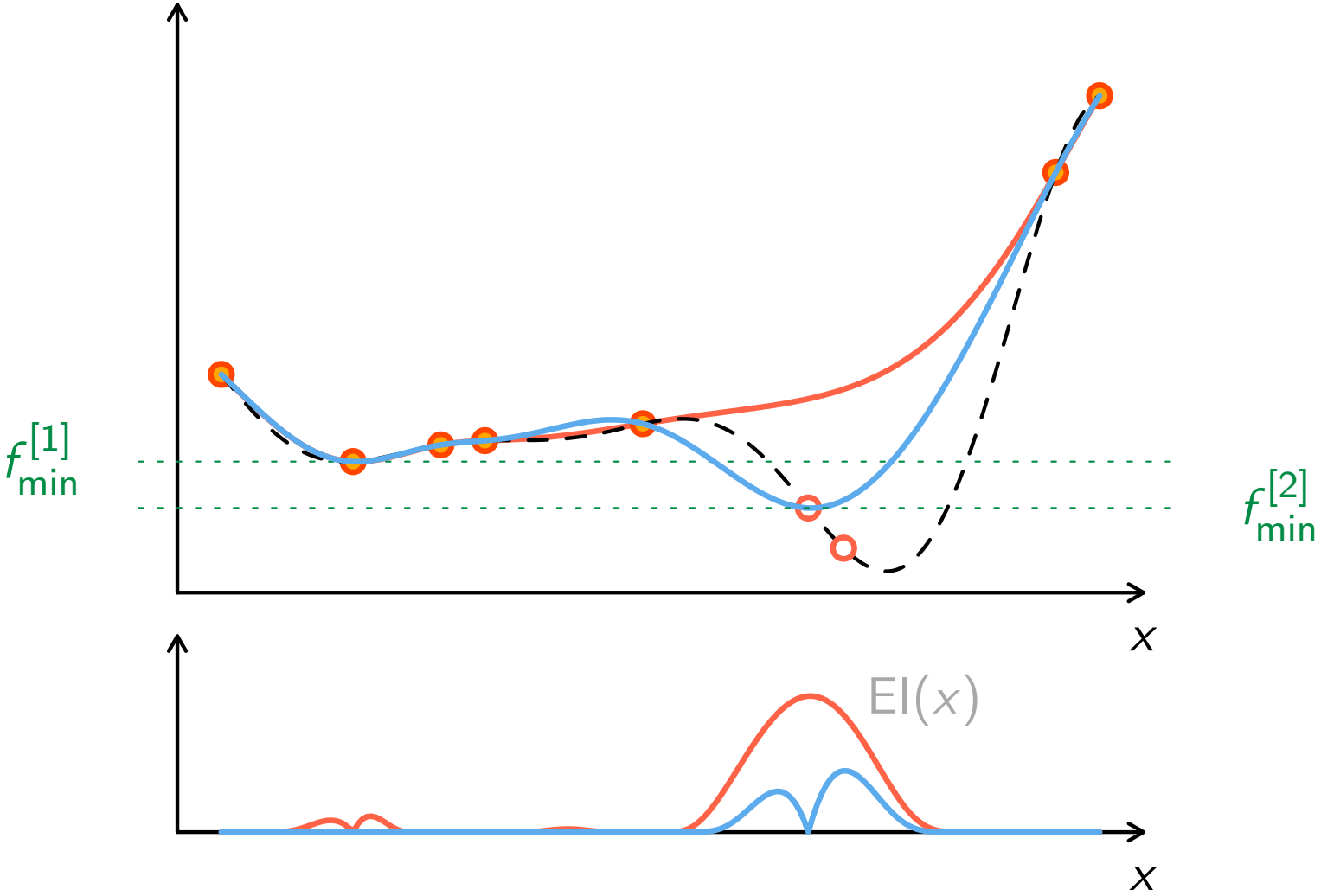
Start with an initial Kriging model. Then repeat until a stopping criterion is reached:

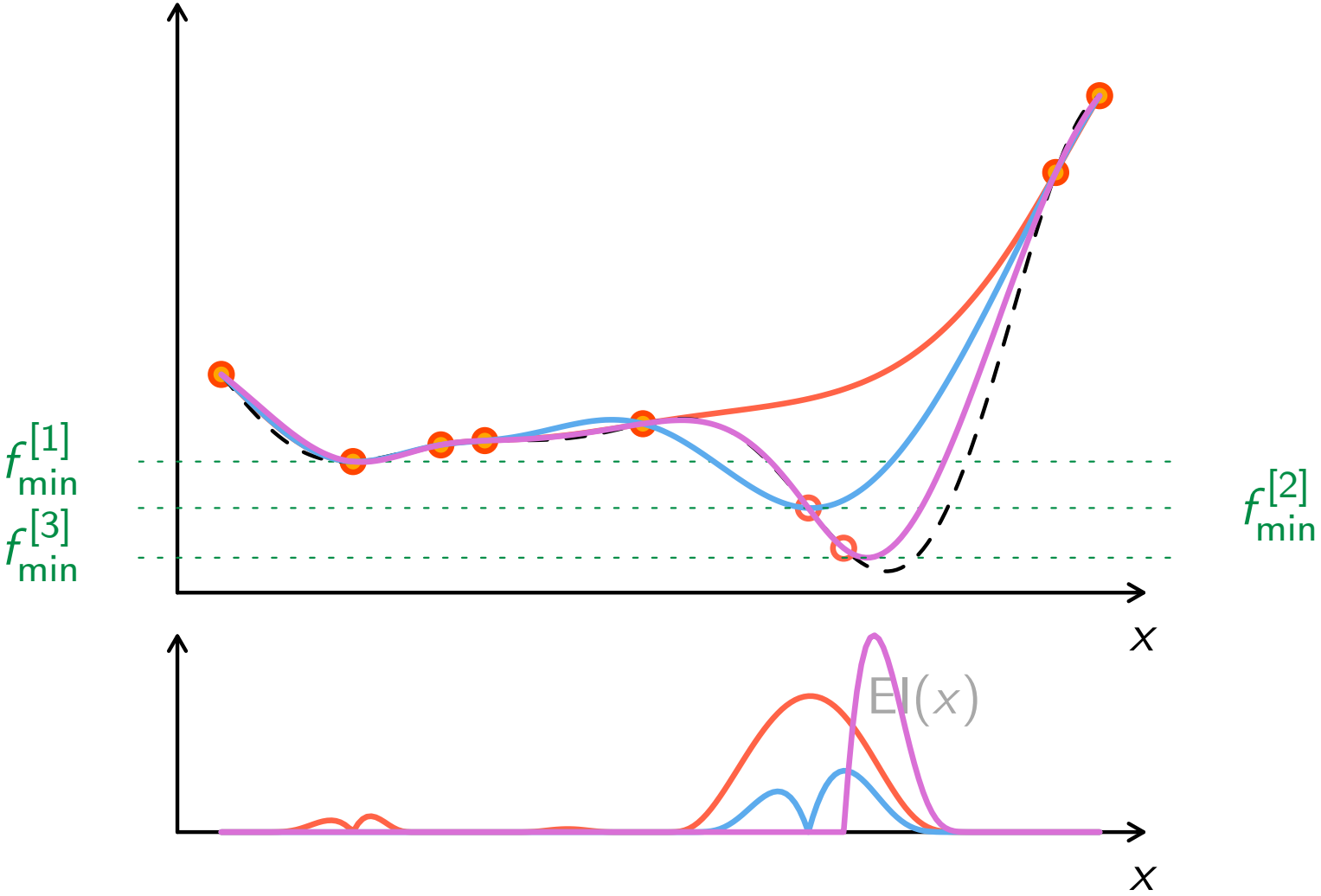
1. Find $\mathbf{x}^\star$ which maximizes $\mathrm{EI}(\mathbf{x})$
2. Evaluate the simulator at $\mathbf{x}^\star$
3. Update the learning set: $\mathbf{X} \leftarrow \mathbf{X} \cup \{\mathbf{x}^\star\}$, $\mathbf{y} \leftarrow \mathbf{y} \cup \{f_{\mathrm{sim}}(\mathbf{x}^\star)\}$
4. (Possibly) reestimate the Kriging model

$\longrightarrow$ "Efficient Global Optimization" (EGO) algorithm of Jones et al [JSW98]


$f_{\min}^{[1]}$
x
$\mathrm{EI}(x)$
x

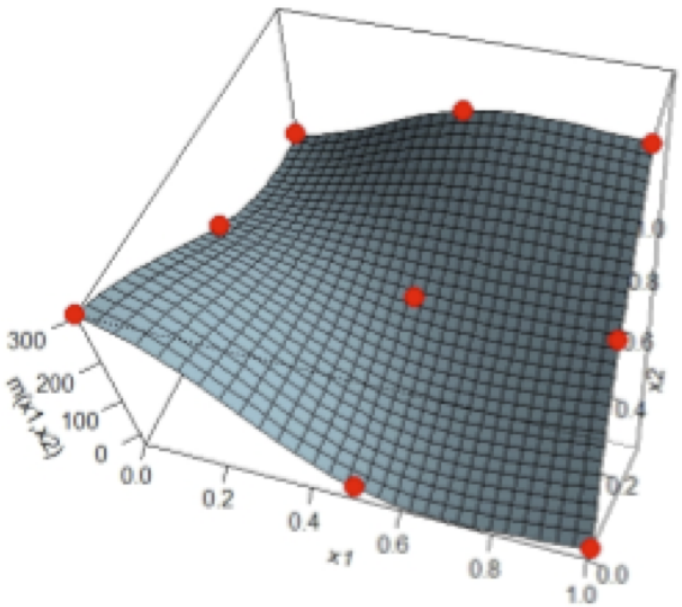

$f_{\min}^{[1]}$
x
$\mathrm{EI}(x)$
x

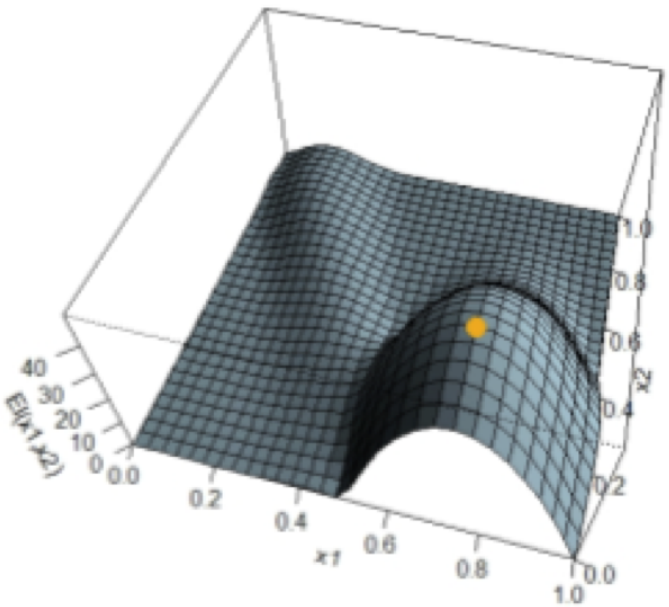

$f_{\min}^{[1]}$
$f_{\min}^{[2]}$
x
$\mathrm{EI}(x)$
x


$f_{\min}^{[1]}$
$f_{\min}^{[3]}$
$f_{\min}^{[2]}$
x
$\mathrm{EI}(x)$
x

A 2D example (Branin function)

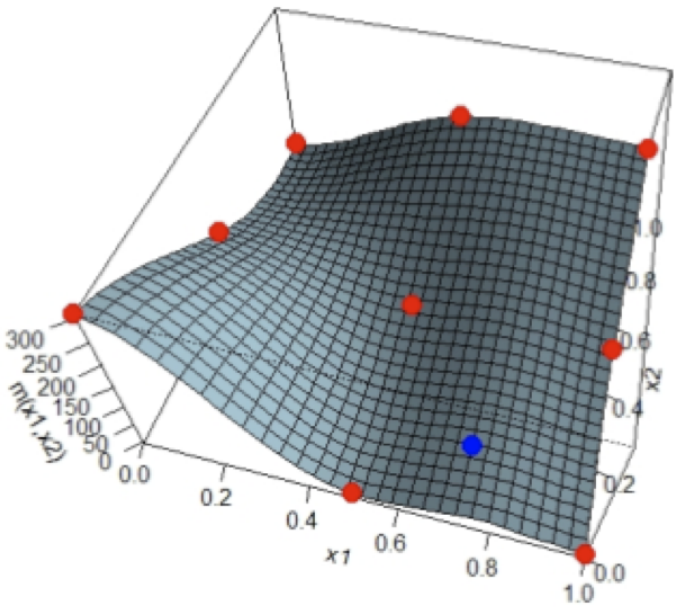
Left: Kriging model surface. Right: EI surface.

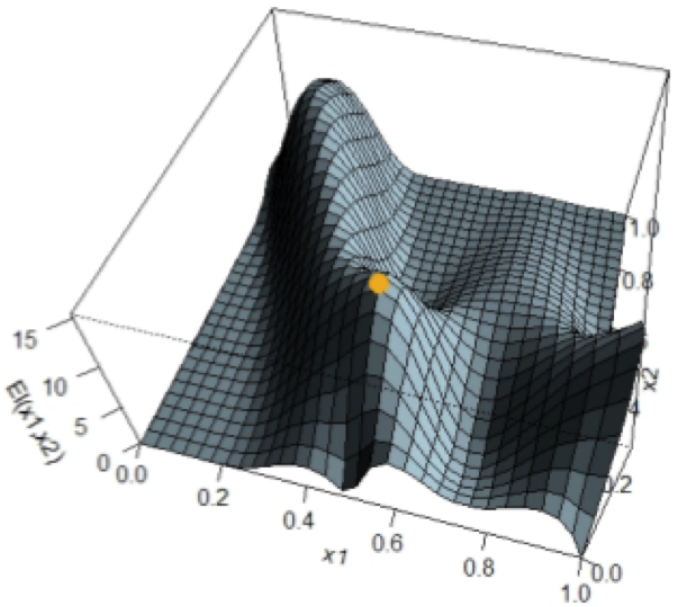




A 2D example (Branin function)

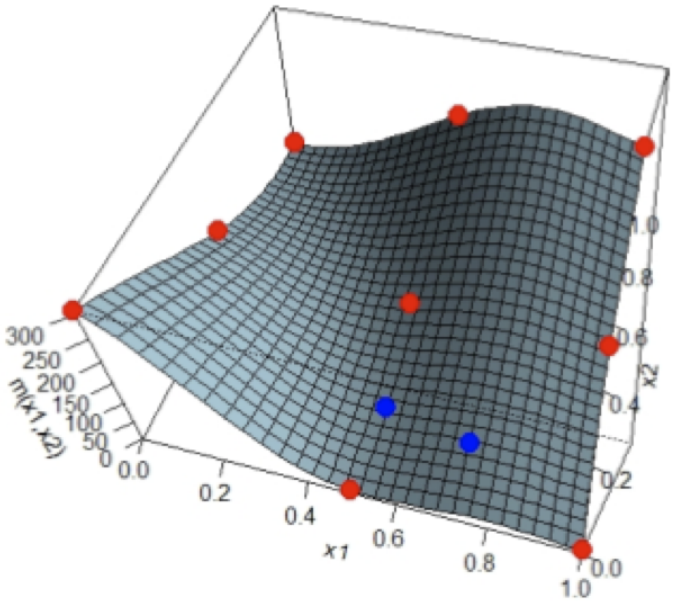
Left: Kriging model surface. Right: EI surface.

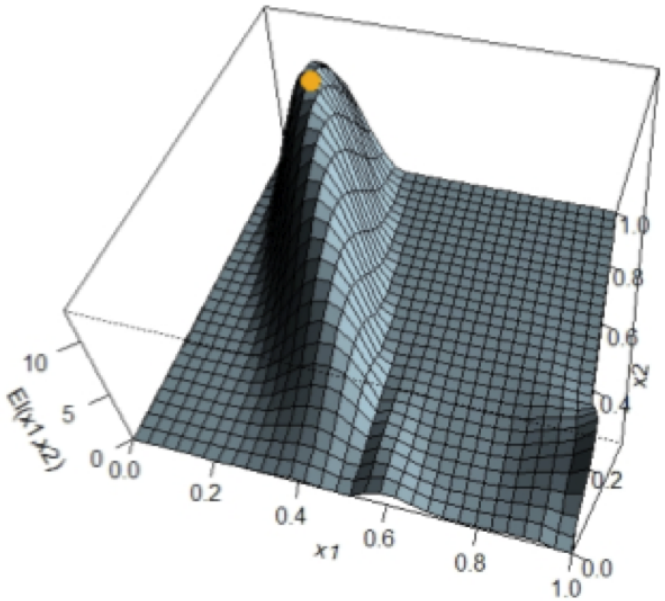




A 2D example (Branin function)

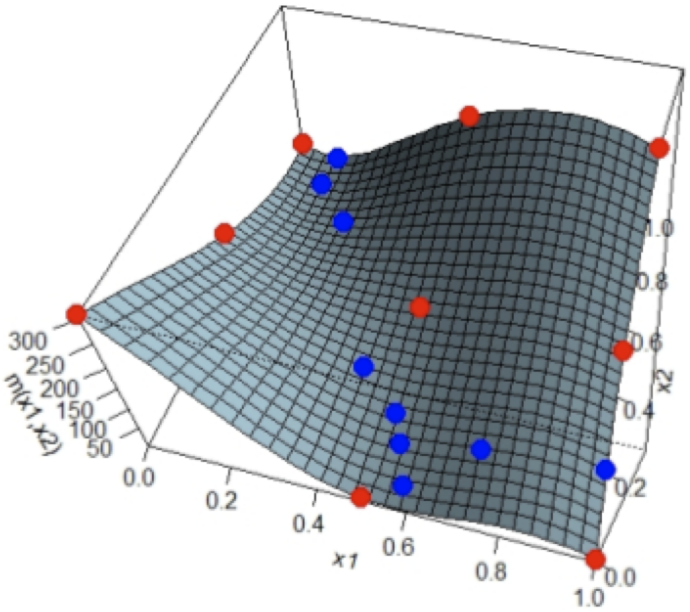
Left: Kriging model surface. Right: EI surface.





A 2D example (Branin function)

Left: Kriging model surface. Right: EI surface.



Adaptation to a parallel setting

Aim: To give a *batch* of points at each step

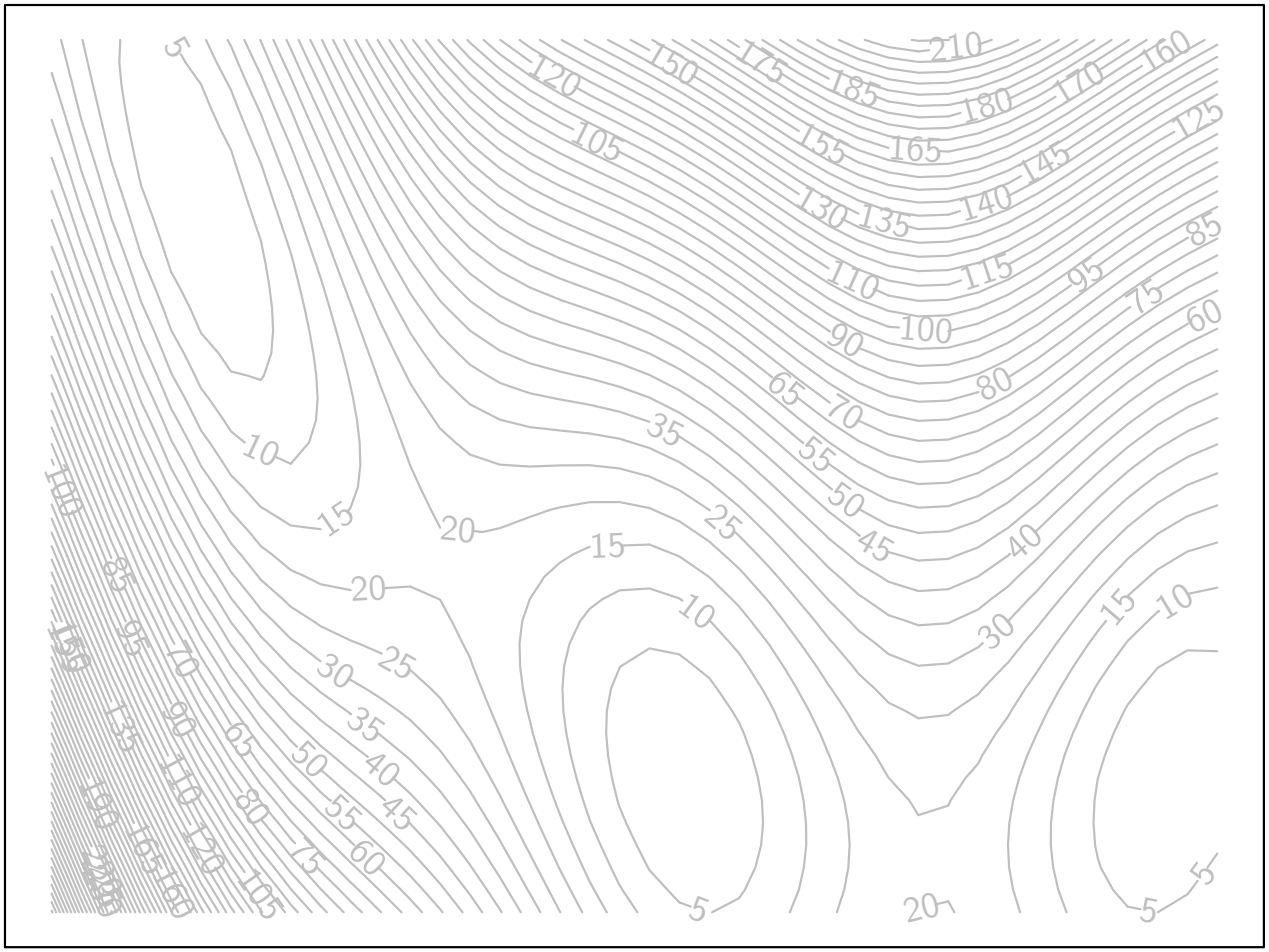
- q-EI: Generalizes EI for a batch of q points

  $\longrightarrow$ A batch gives an improvement if at least one is better

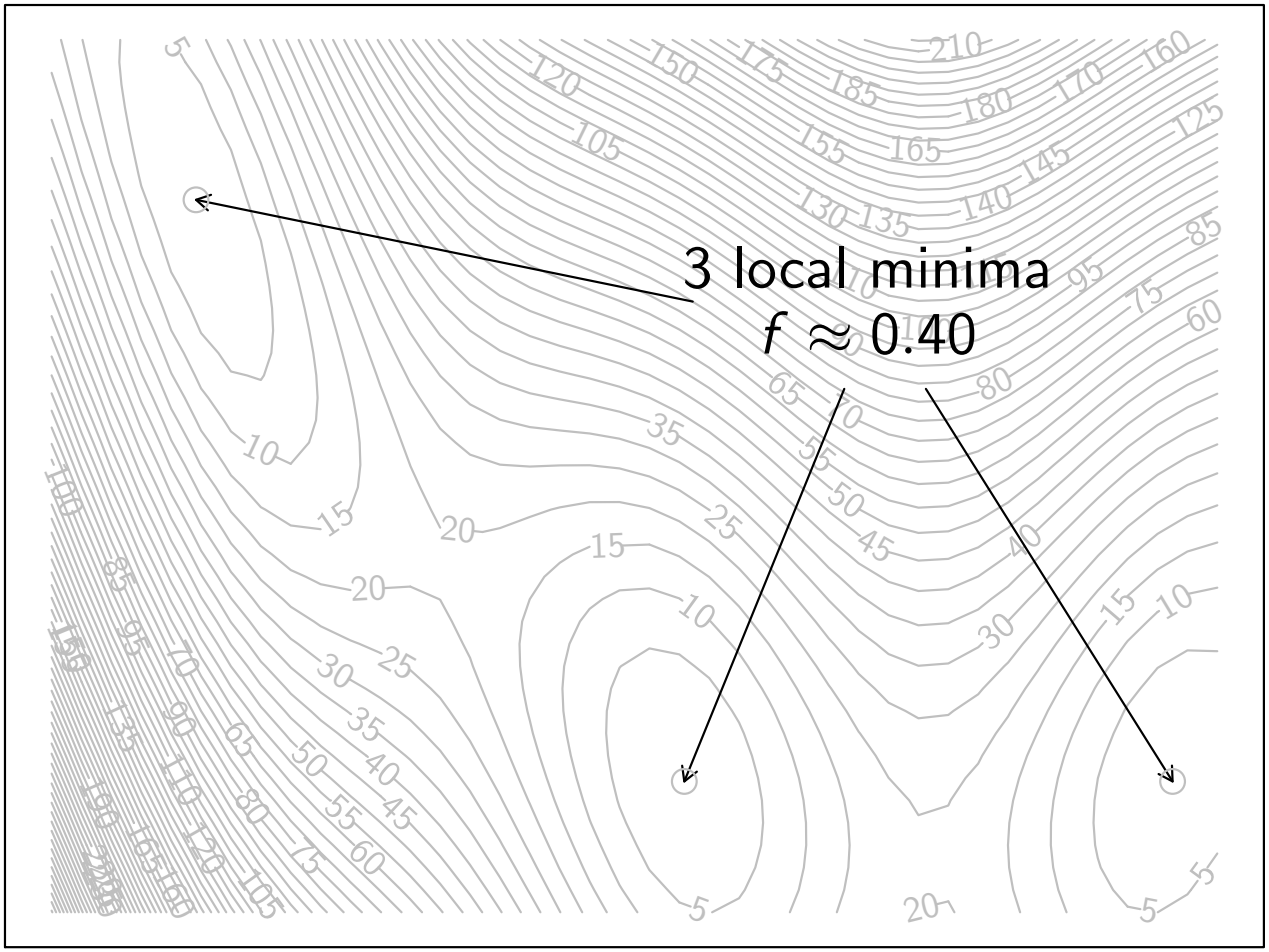
- Liar strategies: Apply q-times the 1-step EI strategy without evaluating the simulator

  $\longrightarrow$ Provide a 'reasonable' value of $f_{\mathrm{sim}}(\mathbf{x}^\star)$, typically the current minimun value

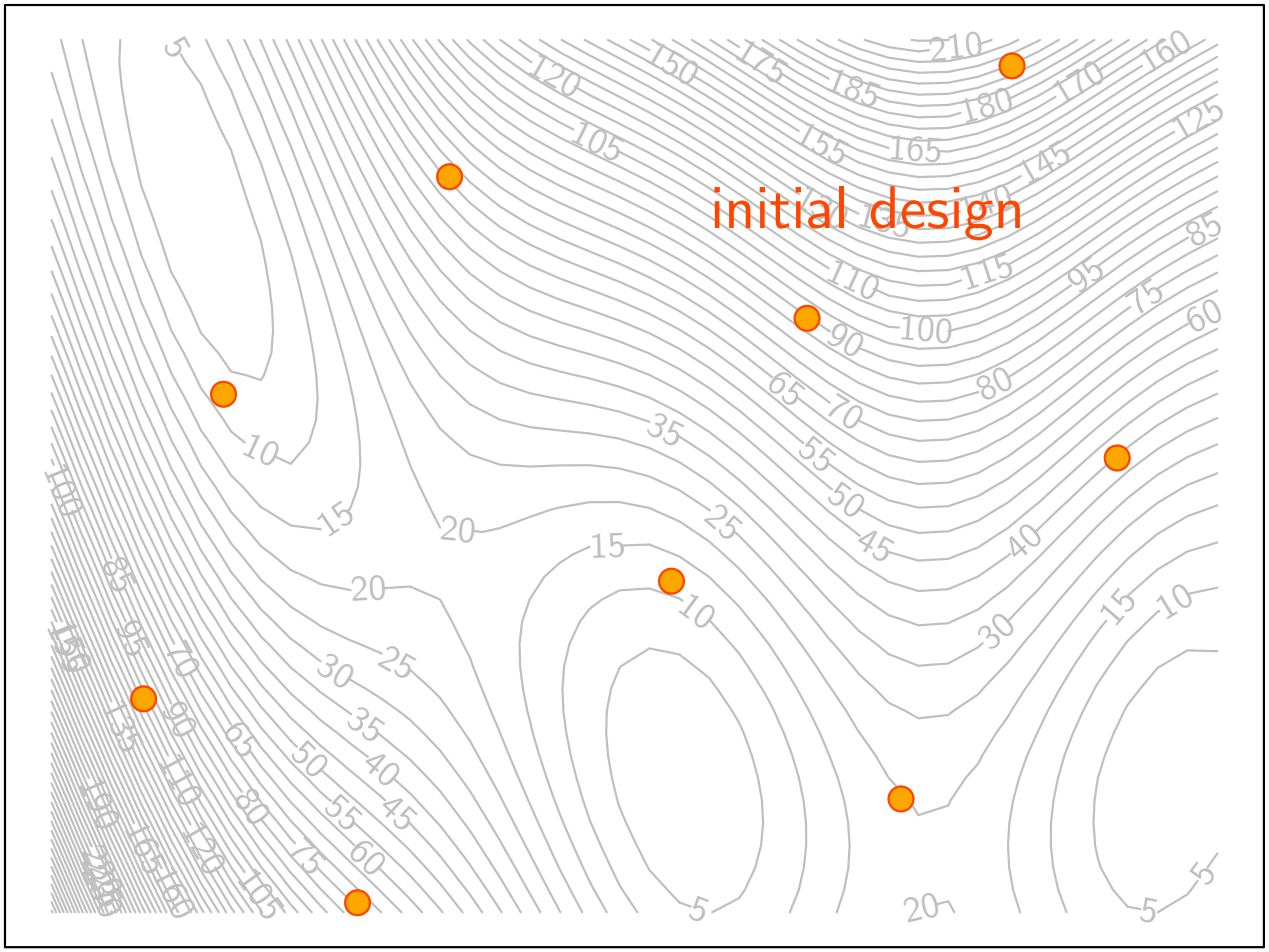
Toy example: Branin function



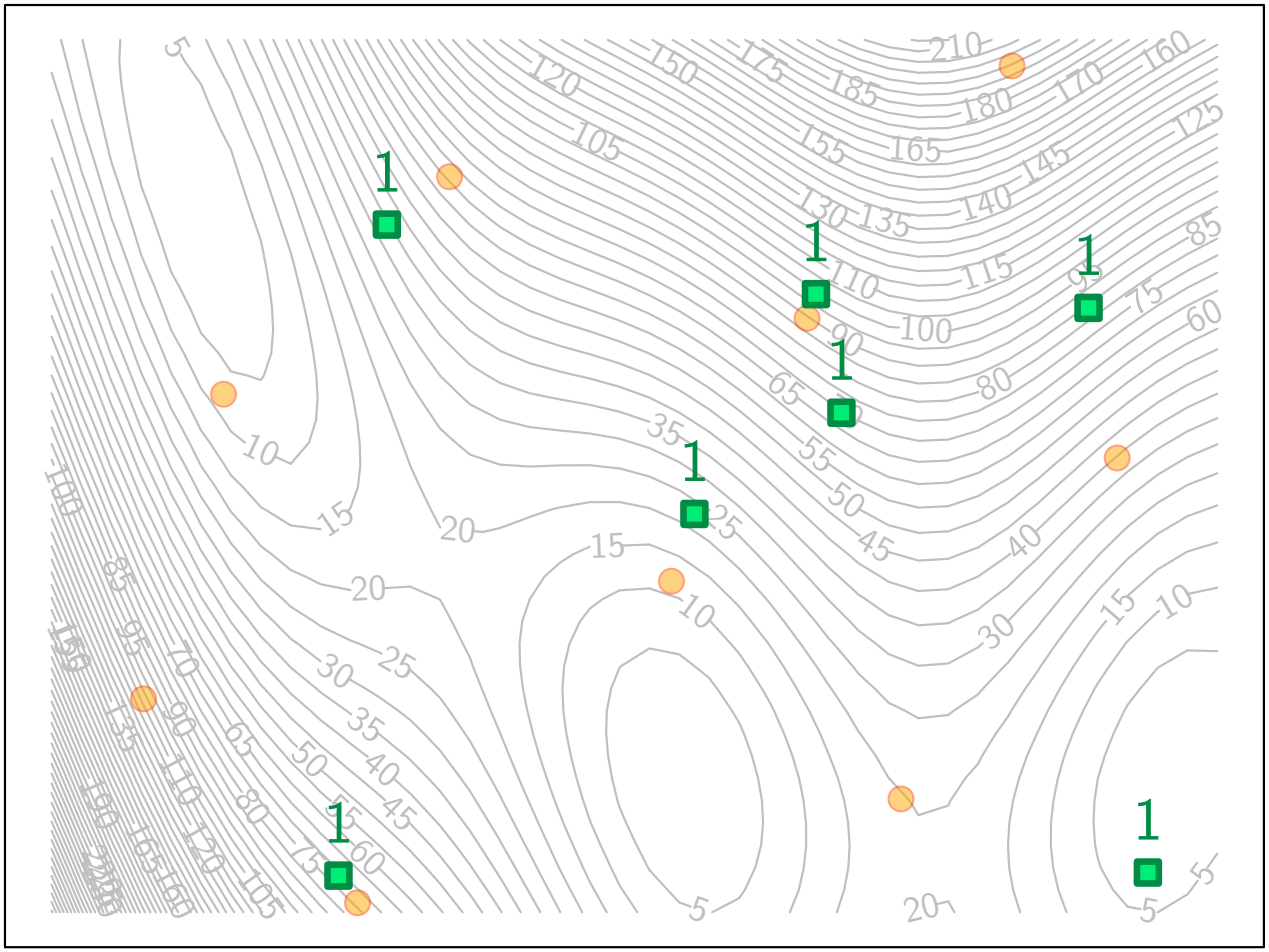
Toy example: Branin function



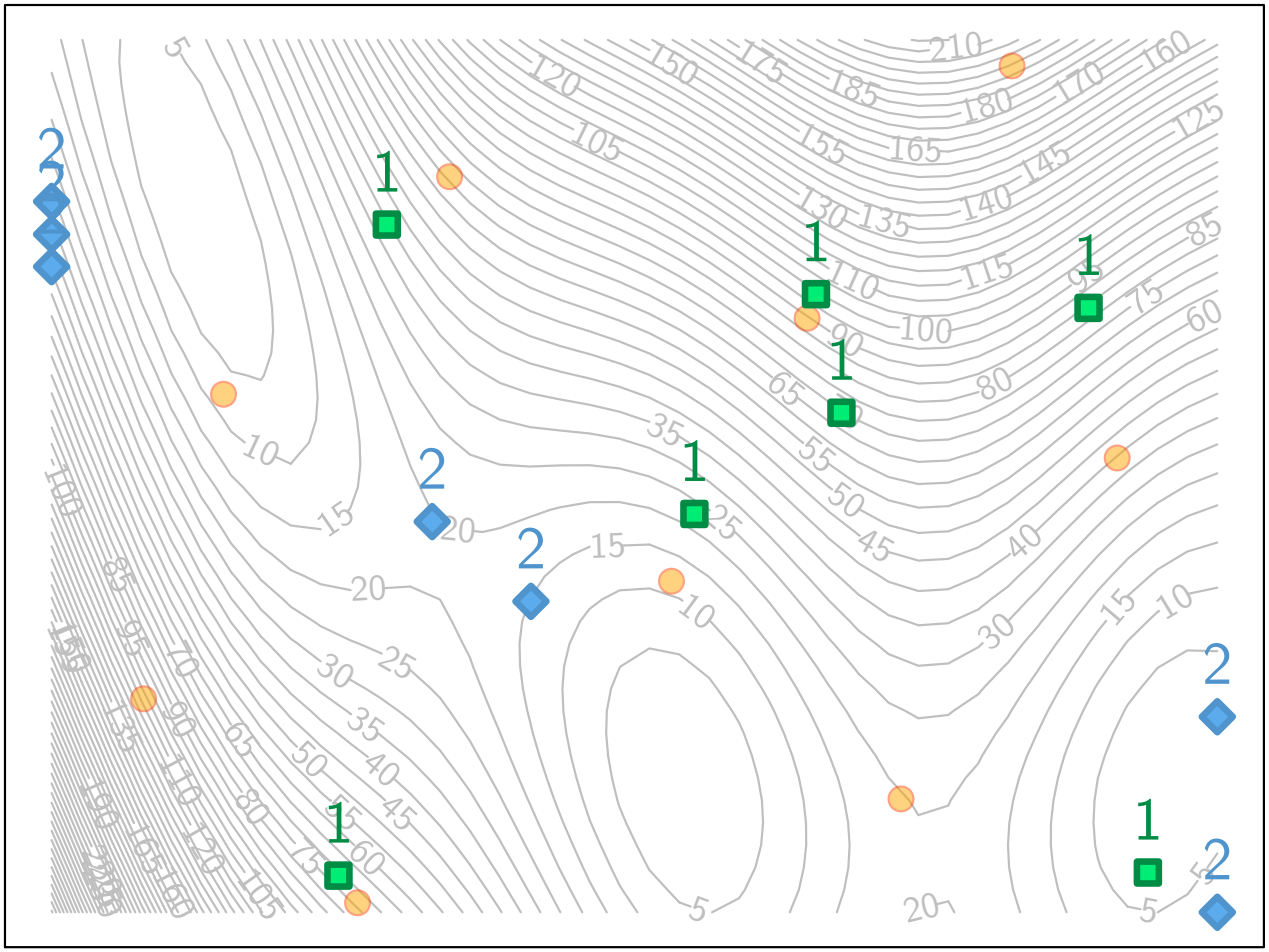
Toy example: Branin function



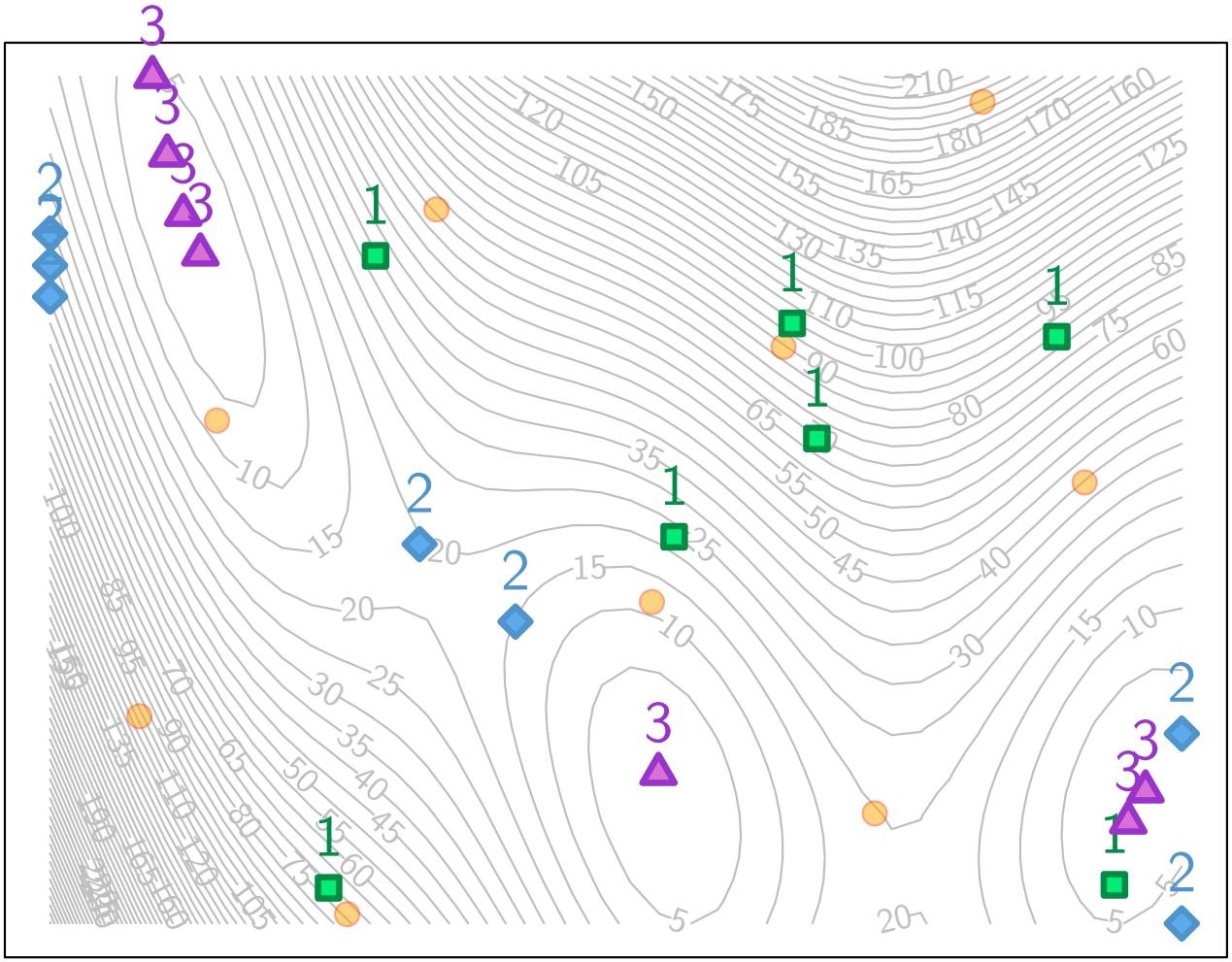
Toy example: Branin function

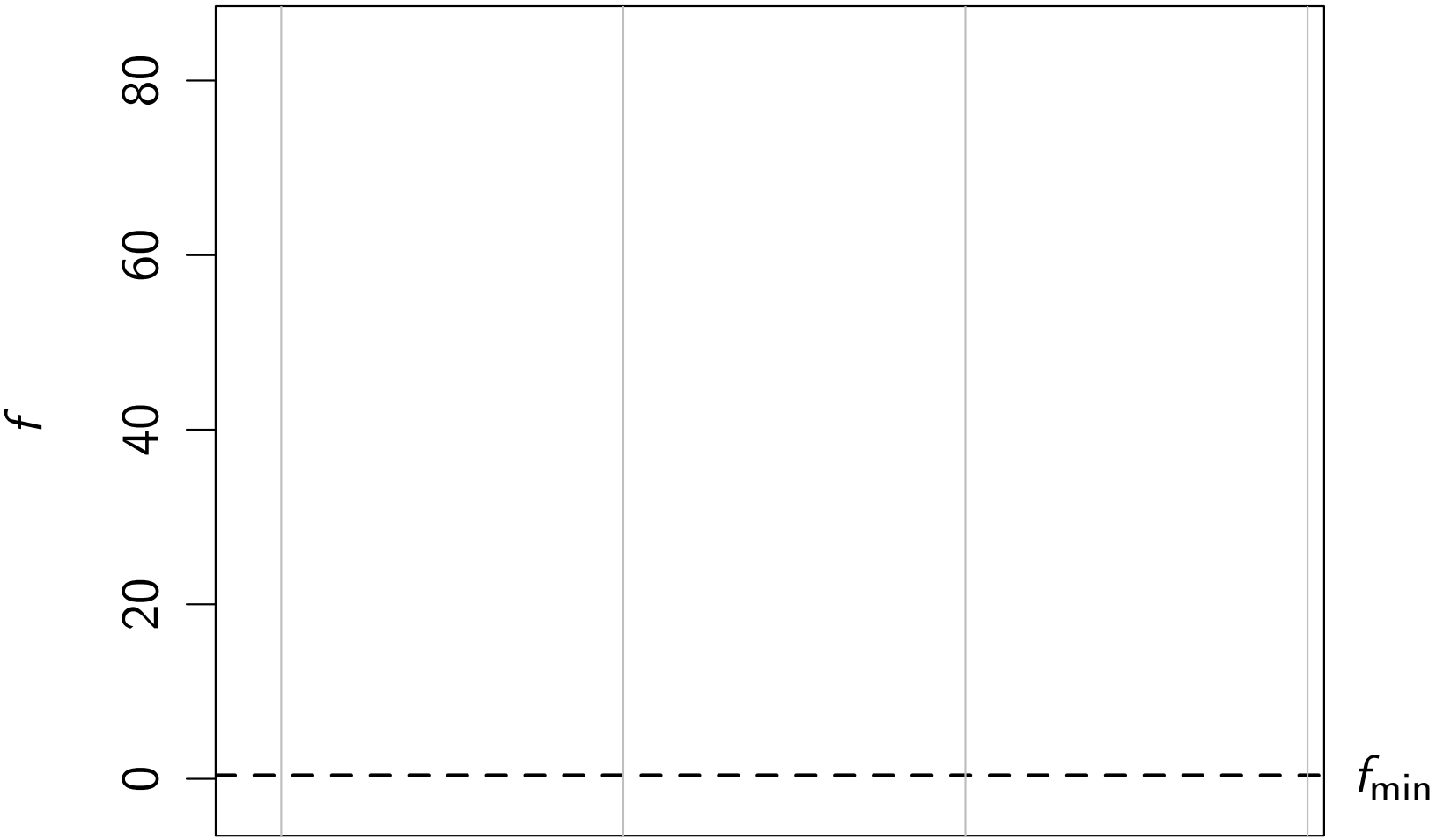


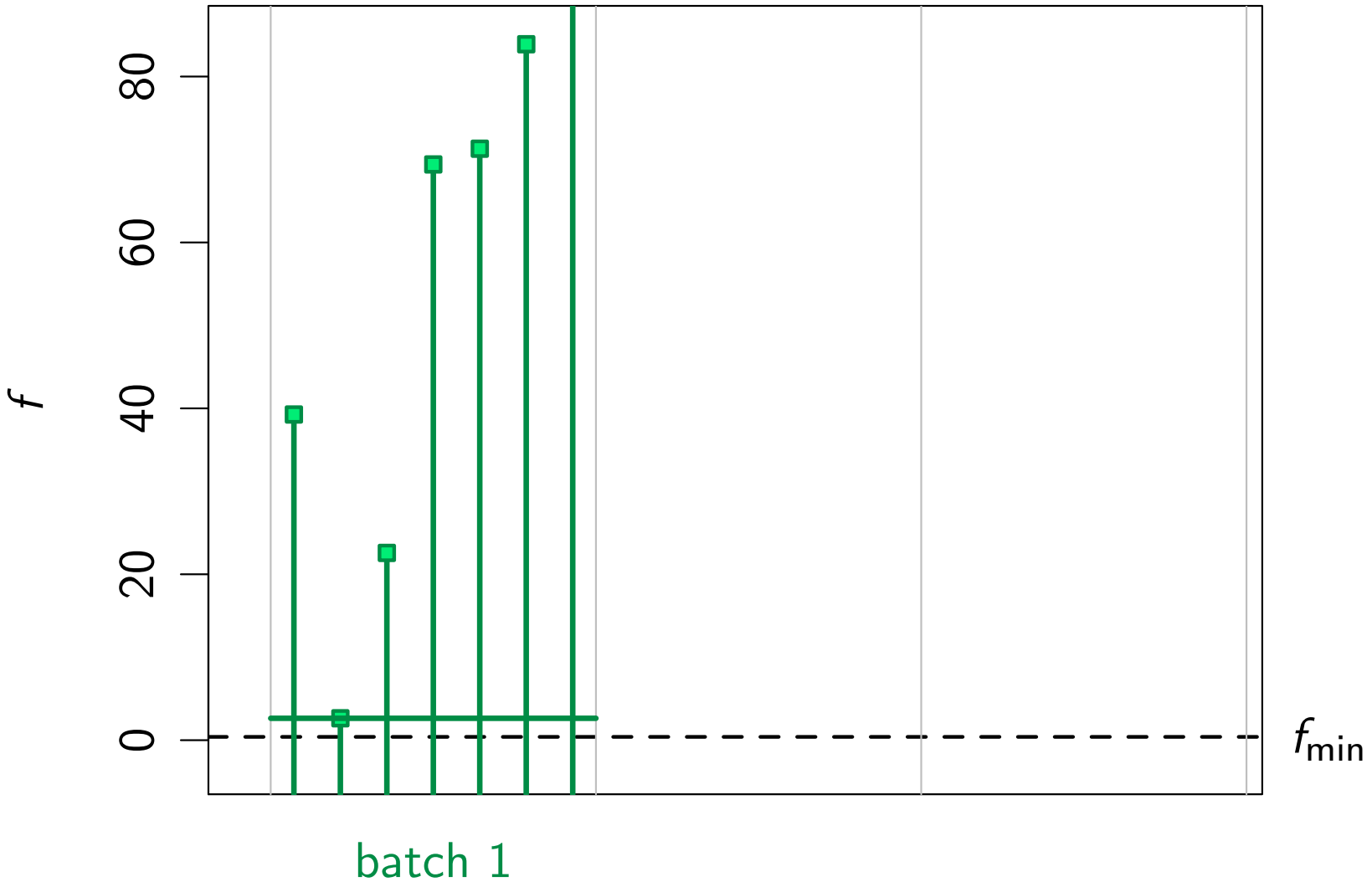
Toy example: Branin function

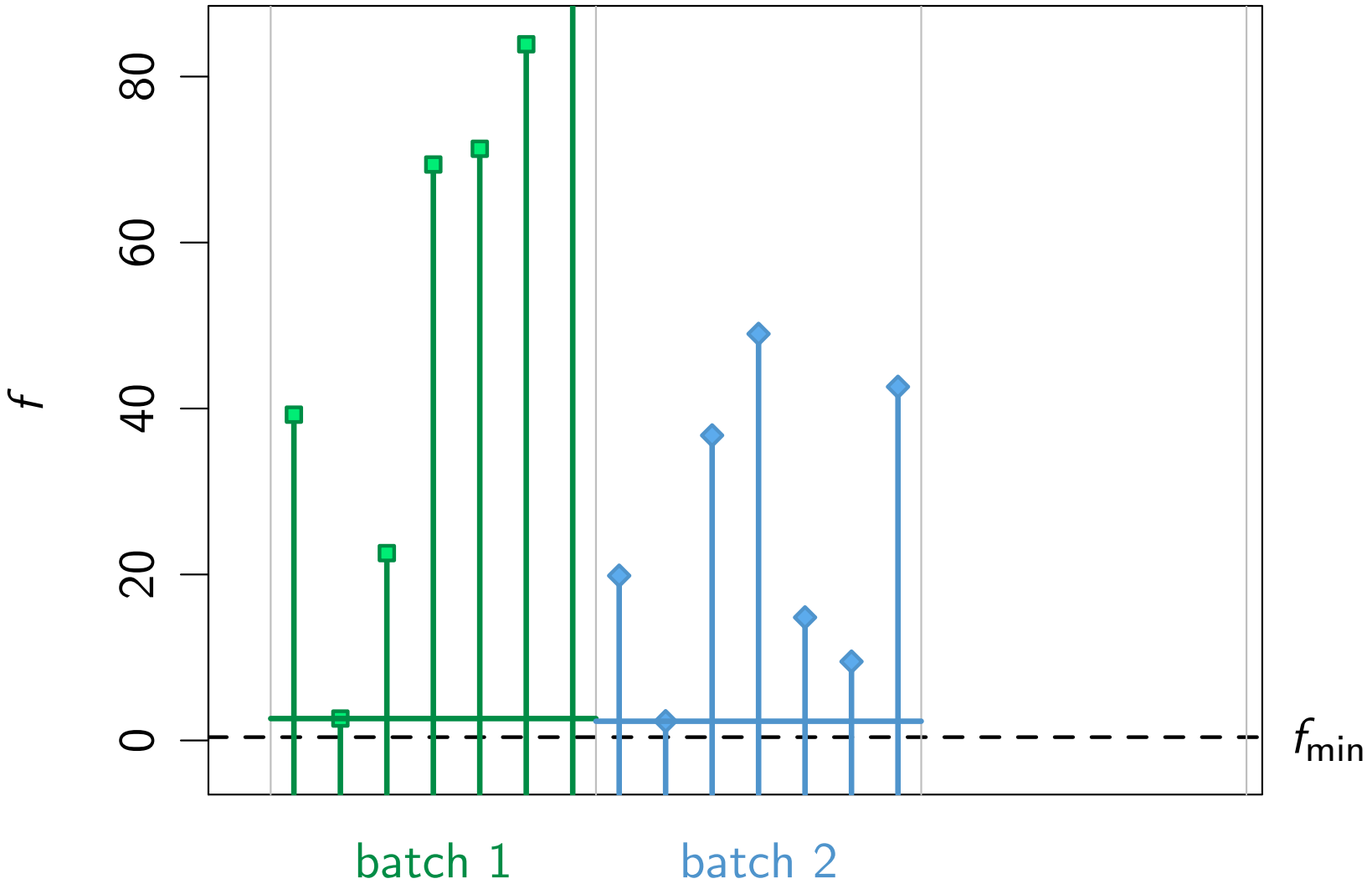


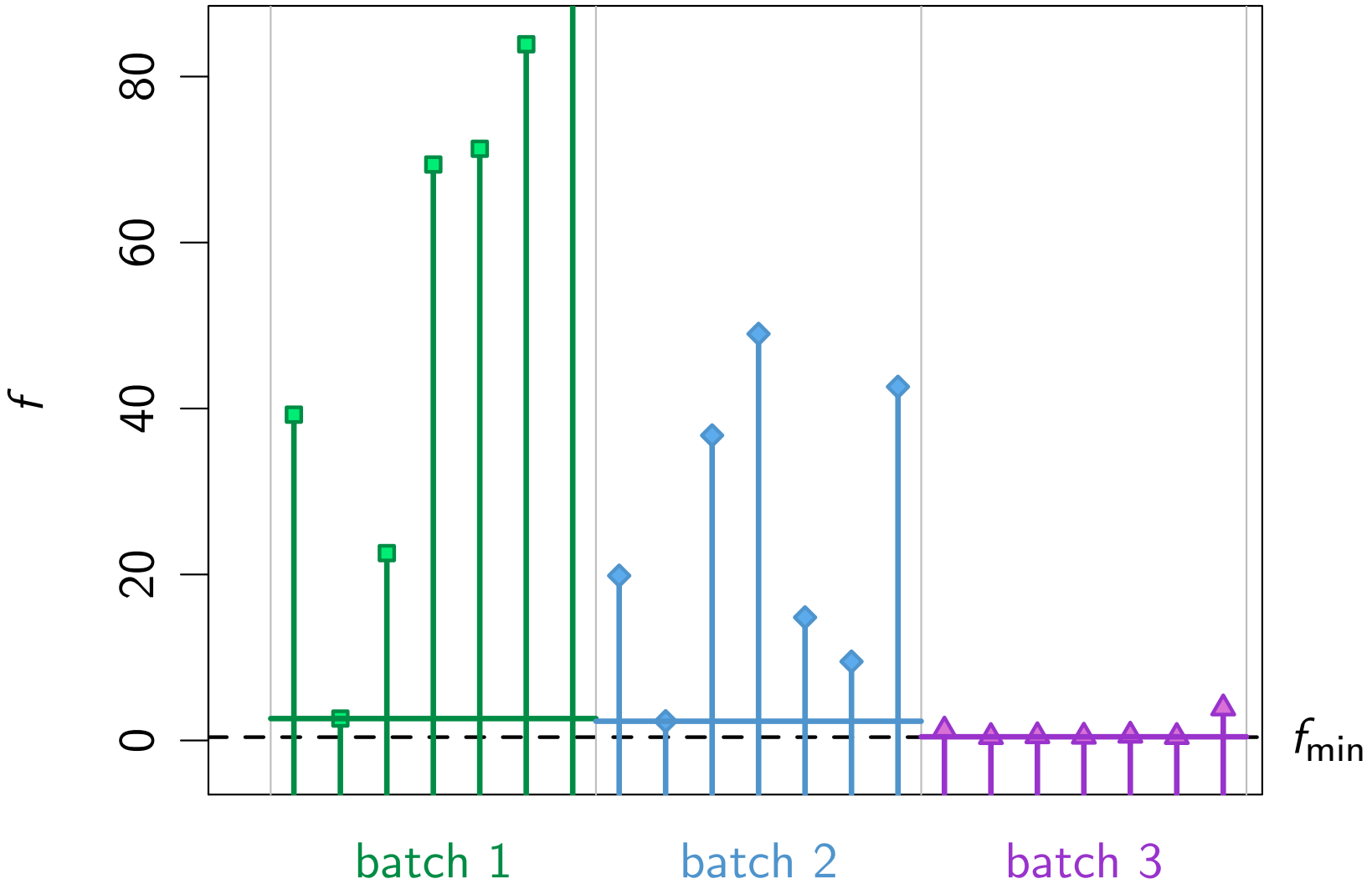
Toy example: Branin function




f
0
20
40
60
80
$f_{\min}$


f
0
20
40
60
80
$f_{\min}$
batch 1


f
0
20
40
60
80
$f_{\min}$
batch 1
batch 2


f
0
20
40
60
80
$f_{\min}$
batch 1
batch 2
batch 3

Example: Usage of ***DiceOptim*** *at IRSN*

promethee.irsn.org

Prométhée is a software workbench created by IRSN. It provides a Graphical User Interface for distributed automated parametric computation on cluster, workstations, desktop, ... It works with several simulators: *MCNP(X)*, *Moret*, *Apollo*, ...

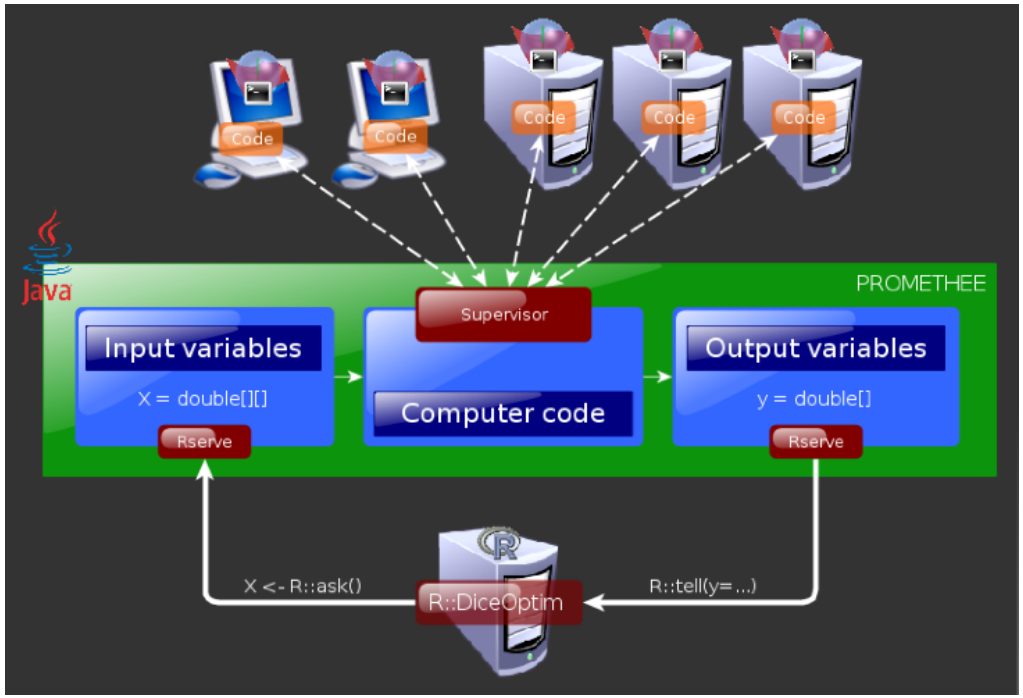
Prométhée is based on Java and RServe by S. Urbanek [Urb13].

R plays a crucial role in several tasks.

- Parameterization of input files.
- Metamodels, optimization, sensitivity analysis, ...
- Scripting language.
- Output management (web).

(Re)Dice packages are widely used.

Example: Usage of ***DiceOptim*** *at IRSN*



Screenshot of Richet et al [RGRD10]. The `ask` and `tell` mechanism is an original idea arising from package **sensitivity** [PIJ13].

DiceOptim features

1. Kriging-based optimization of deterministic simulators
   - EI algorithm (EGO)
   - Parallel EI algorithm: q-points EI, CL heuristics

DiceOptim features

1. Kriging-based optimization of deterministic simulators
   - EI algorithm (EGO)
   - Parallel EI algorithm: q-points EI, CL heuristics
2. Kriging-based optimization for noisy observations
   - Noisy EI-like criteria: Expected Quantile Improvement (EQI), Augmented EI (AEI), Approximate Knowledge Gradient (AKG)
   - Corresponding sequential optimization strategies

DiceOptim features

1. Kriging-based optimization of deterministic simulators
   - EI algorithm (EGO)
   - Parallel EI algorithm: q-points EI, CL heuristics

2. Kriging-based optimization for noisy observations
   - Noisy EI-like criteria: Expected Quantile Improvement (EQI), Augmented EI (AEI), Approximate Knowledge Gradient (AKG)
   - Corresponding sequential optimization strategies

$\longrightarrow$ For most criteria (EI, EQI, AKG), the analytical gradient is supplied, enhancing the algorithms efficiency

Part V

Playing with Kernels

Special kernels

DiceKriging allows the user to write her/his own kernel from a simple R function.

This function can be "inlined" using the **inline** package [SMS+13].

Example: kernel with an *invariance property*, e.g. symmetry. Specificity of computer experiments; usually not a concern in spatial stat. or geostatistics. Invariant kernels has been a theme of research in Dice and ReDice consortiums, see Ginsbourger et al. [GBRC12].

Special kernels

```
R> k <- function(x1, x2) {
+      0.5 * exp(-((x1 - x2) / 0.15)^2) +
+          0.5 * exp(-((1 - x1 - x2) / 0.15)^2)
+  }
R> kmUser <- kmData(y ~ 1, data = df, coef.trend = 0, kernel = k)
```

The defined kernel k has a 'magical' property

$$4\,k(x_1,\,x_2) = k^\star(x_1,\,x_2) + k^\star(s.x_1,\,x_2) + k^\star(x_1,\,s.x_2) + k^\star(s.x_1,\,s.x_2)$$

where s is the symmetry $x \mapsto 1 - x$ and $k^\star$ is a kernel. The sample paths become symmetrical.

Special kernels

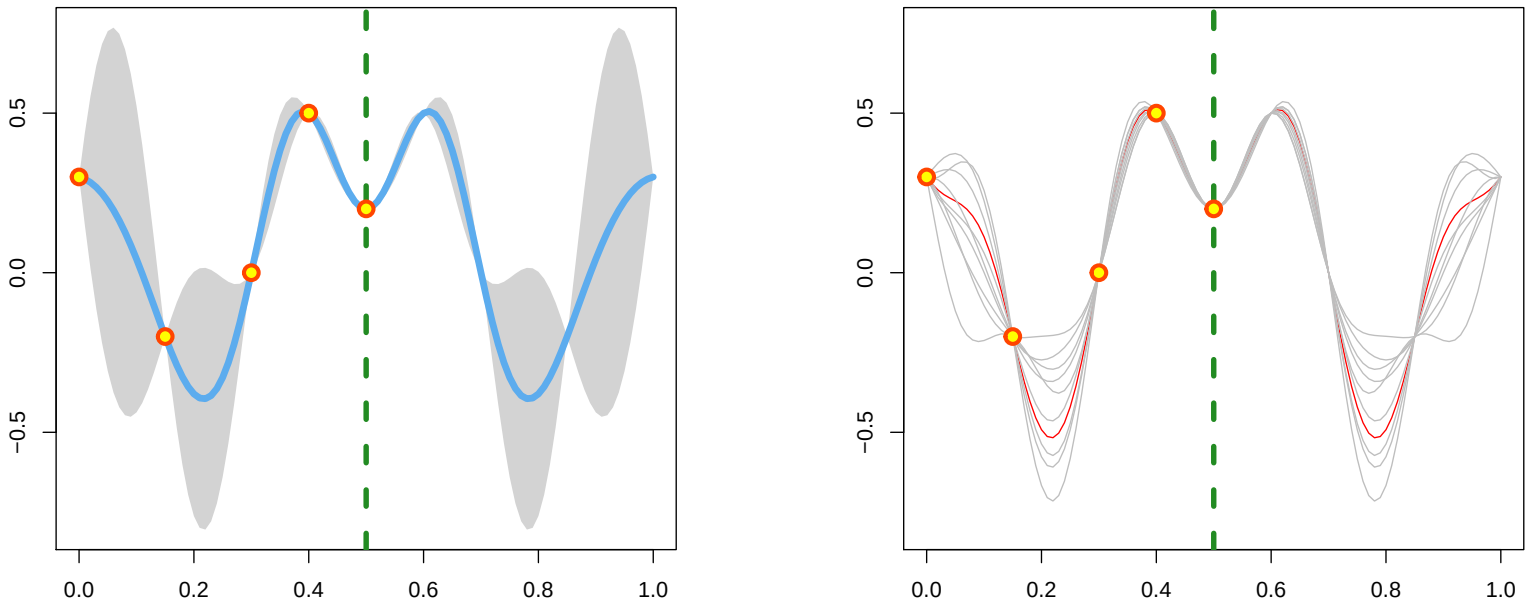


Figure: We can use `predict`, `simulate`, etc. symmetry automatically results. Left: prediction from 5 points. Right: conditional simulations

Special kernels

This generalizes to kernels invariant by a finite group $\mathcal{G}$ of transformations

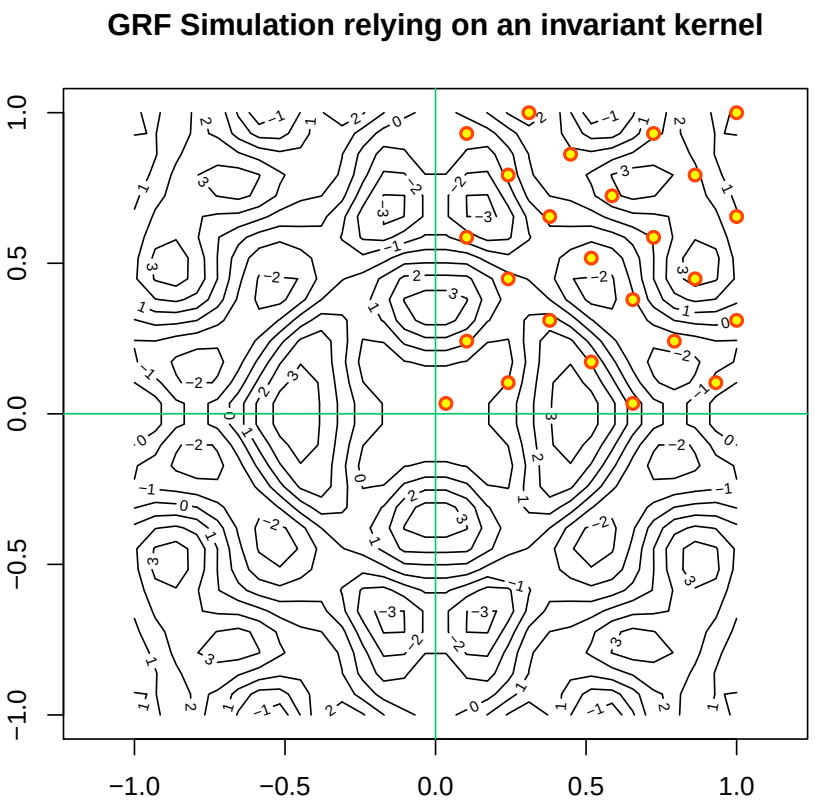
$$k(\mathbf{x}_1,\, \mathbf{x}_2) \propto \sum_{s_1,\, s_2 \in \mathcal{G}} k^\star(s_1.\mathbf{x}_2,\, s_2.\mathbf{x}_2)$$

the dimension d and the kernel $k^\star$ being arbitrary.

$\longrightarrow$ $\mathcal{G}$ was a group of 2 transformations id and s in previous example.

$\longrightarrow$ We can play with two axial symmetries in dimension $d = 2$. Then $\mathcal{G}$ has 4 elements.

Special kernels



Simulation from a kernel for dimension $d = 2$ with two axial symmetries.

Special kernels

User defined kernels have been made more performant and flexible in package **DKlab** (forthcoming).

- Parameter estimation by Maximum Likelihood.
- Optimization of performance through `.Call`.

Part VI

Practical Considerations

Package development

A few hints arising from our experience about R packages development within the consortiums(s): technical and non-technical considerations.

Take time to learn R

- Some key R objects must be perfectly known or understood:

  $\longrightarrow$ `data.frame` $\neq$ `matrix`, `factor`, ...
- A R function or *closure* is an amazingly powerful thing!

  $\longrightarrow$ but some practice is needed to make good use of it: *dots*, missing formals, scoping, ...
- Read *again and again* the manual Writing R Extensions.
- Take time to study the code of good existing packages.

  $\longrightarrow$ Packages written by R core members are of great help

Make your life easier

- Dramatic productivity gains can be reached by using RStudio for package development. This is especially true for new developers and PhD students.
- ... but some still love emacs/ESS and command lines `R CMD build`
- Packages such as **roxygen2**, **testthat** are of great value.
- Use a Version Control System.
- It is a real strength to co-work on a variety of platforms, text editors, graphical devices, ...

Package design

Kriging metamodels are not unlike statistical models...

- The data/formula interface is very flexible to create meta-models.

```
fit <- kmData(y ~ 1, data = myData,
              inputnames = c("Temp", "Press"))
```

- But must sometimes be completed.

  $\longrightarrow$ GP models have predictors *and inputs*. The order of the inputs is important.

- Implementing classical methods make a package easier to use: `summary`, `coef`, `predict`, `simulate`, ...

Methods

- Methods (S3, S4 or R5) enhance code reliability.
- Methods can allow users to extend our (Re)Dice packages *from outside*.

  $\longrightarrow$ write a new kernel class and a few methods as in **nlme**, among which `coef` and `'coef<-'`
- But writing methods requires a clear vision of the final code.

  $\longrightarrow$ using S3 temporarily can be a solution
- When you find duplicated code, consider writing a method.

Documentation

- Writing good documentation is difficult and time-consuming
  $\longrightarrow$ documenting S4 methods often generates headaches
- Write vignettes or reproducible research documents!
  $\longrightarrow$ Who reads the pdf version of a package's manual?

Compiled code

- Writing compiled code is necessary for some computational CE tasks: *kriging*, building complex designs, ...

  $\longrightarrow$ C, C++ or Fortran

- Using `.C` in a package is fairly simple and efficient.

  $\longrightarrow$ however, with much object duplication

- Very few people enjoy writing code with `.Call`.

  $\longrightarrow$ the use of macros can be intimidating

- Consider using RCpp! [EF11]

  $\longrightarrow$ there is certainly a good C/C++ programmer in an office next to your's.

Project management

- It is quite difficult for one person to write a package of CRAN (or higher) quality. This can be time-consuming.
- It is necessary to have feedback during the package development.
- Proving theorems and writing good R code is a real challenge for applied math PhD students!

Authorship

- Dice and even more ReDice consortiums are very well suited to support and encourage R package development

  $\longrightarrow$ at the present time, the CRAN policy may discourage some companies: maintainer and author(s) must be *persons*, which may be a problem.
- Urge academics/researchers on properly citing R CRAN packages (and their authors) as they do with research articles.

Thank you!

Merci à Kurt Hornik, Uwe Ligges et Brian Ripley, R core members

These slides were produced using *Sweave* by Friedrich Leisch.

Bibliography I

[CPG14] C. Chevalier, V. Picheny, and D. Ginsbourger, *Kriginv: An efficient and user-friendly implementation of batch sequential inversion strategies based on kriging*, Computational Statistics & Data Analysis **71** (2014), 1021–1034.

[DCI13] G. Damblin, M. Couplet, and B. Iooss, *Numerical studies of space filling designs: optimization of latin hypercube samples and subprojection properties*, Journal of Simulation **7** (2013), 276–289.

[DGR12] N. Durrande, D. Ginsbourger, and O. Roustant, *Additive Covariance kernels for high-dimensional Gaussian Process modeling*, Annales de la Faculté des Sciences de Toulouse **21** (2012), no. 3, 481–499.

[DH11] D. Dupuy and C. Helbert, *DiceEval: Construction and evaluation of metamodels*, 2011, R package version 1.1.

[DIMW14] G. Defaux, B. Iooss, V. Moutoussamy, and C. Walter, *mistral: Methods in Structural Reliability*, 2014, with contributions from N. Bousquet and P. Lemaitre.

Bibliography II

[DS13] C. Dutang and P. Savicky, *randtoolbox: Generating and testing random numbers*, 2013, R package version 1.13.

[Edd13] D. Eddelbuettel, *Seamless R and C++ integration with Rcpp*, Springer, New York, 2013, ISBN 978-1-4614-6867-7.

[EF11] D. Eddelbuettel and R. François, *Rcpp: Seamless R and C++ integration*, Journal of Statistical Software **40** (2011), no. 8, 1–18.

[FDR+14] J. Franco, D. Dupuy, O. Roustant, G. Damblin, and B. Iooss, *DiceDesign: Designs of Computer Experiments*, 2014, R package version 1.4.

[FLS06] K. Fang, R. Li, and A. Sudjianto, *Design and modeling for computer experiments*, Chapman & Hall/CRC, 2006.

[FMRJ13] J. Fruth, T. Muehlenstaedt, O. Roustant, and M. Jastrow, *fanovaGraph: Building Kriging models from FANOVA graphs*, 2013, R package version 1.4.4.

Bibliography III

[GBRC12] D. Ginsbourger, X. Bay, O. Roustant, and L. Carraro, *Argumentwise invariant kernels for the approximation of invariant functions*, Annales de la Faculté des Sciences de Toulouse **21** (2012), no. 3, 501–527.

[GLC10] D. Ginsbourger, R. Le Riche, and L. Carraro, *Computational intelligence in expensive optimization problems*, Studies in Evolutionary Learning and Optimization, ch. "Kriging is well-suited to Parallelize Optimization", pp. 131–162, Springer-Verlag, 2010.

[Gra12] L. Le Gratiet, *MuFiCokriging: Multi-Fidelity Cokriging models*, 2012, R package version 1.2.

[JSW98] D. R. Jones, M. Schonlau, and W. J. Welch, *Efficient global optimization of expensive black-box functions*, Journal of Global Optimization **13** (1998), 455–492.

Bibliography IV

[Kri51] D. G. Krige, *A statistical approach to some basic mine valuation problems on the witwatersrand*, Journal of the Chemical, Metallurgical and Mining Society of South Africa **52** (1951), no. 6, 119–139.

[Lei02] F. Leisch, *Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis*, Compstat 2002 — Proceedings in Computational Statistics (Wolfgang Härdle and Bernd Rönz, eds.), Physica Verlag, Heidelberg, 2002, ISBN 3-7908-1517-9, pp. 575–580.

[Mat63] G. Matheron, *Principles of geostatistics*, Economic Geology **58** (1963), 1246–1266.

[PBD+13] J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team, *nlme: Linear and Nonlinear Mixed Effects Models*, 2013, R package version 3.1-113.

Bibliography V

[PG14] V. Picheny and D. Ginsbourger, *Noisy kriging-based optimization method: a unified implementation within the DiceOptim package*, Computational Statistics & Data Analysis **71** (2014), 1035–1053.

[PGDP13] R. Paulo, G. Garcia-Donato, and J. Palomo, *SAVE: R package for the statistical analysis of complex computer models*, R package version 0.9.3.7.

[PIJ13] G. Pujol, B. Iooss, and A. Janon, *sensitivity: Sensitivity Analysis*, 2013, R package version 1.7.

[R C13] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013.

[RDC12] Y. Richet, Y. Deville, and C. Chevalier, *DiceView: Plot methods for computer experiments design and surrogate*, 2012, R package version 1.3-0.

Bibliography VI

[RGD12] O. Roustant, D. Ginsbourger, and Y. Deville, *DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization*, Journal of Statistical Software **51** (2012), no. 1, 1–55.

[RGRD10] Y. Richet, D. Ginsbourger, O. Roustant, and Y. Deville, *A grid computing environment for design and analysis of computer experiments*, UseR! conference, Gaithersburg, Maryland, USA, 2010.

[RW06] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, the MIT Press, `http://www.GaussianProcess.org/gpml`, 2006.

[SMS+13] O. Sklyar, D. Murdoch, M. Smith, D. Eddelbuettel, and R. Francois, *inline: Inline C, C++, Fortran function calls from R*, 2013, R package version 0.3.13.

Bibliography VII

[SWMW89] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statistical Science **4** (1989), no. 4, 409–435.

[SWN03] T. J. Santner, B. J. Williams, and W. Notz, *The design and analysis of computer experiments*, Springer-Verlag, New York, 2003.

[Urb13] S. Urbanek, *Rserve: Binary r server*, 2013, R package version 1.7-3.

[WDE11] H. Wickham, P. Danenberg, and M. Eugster, *roxygen2: In-source documentation for r*, 2011, R package version 2.2.2.

[Wic13] H. Wickham, *testthat: Testthat code. tools to make testing fun :)*, 2013, R package version 0.7.1.