Heteroskedastic Gaussian Processes for Simulation Experiments

Mickaël Binois (Inria Sophia Antipolis - Méditerranée)
mickael.binois@inria.fr

joint work with Matthias Chung (VT), Robert B. Gramacy (VT), Jiangeng Huang (VT), Mike Ludkovski (UCSB), Victoria Lyu (UCSB), Stefan Wild (Argonne), and Nathan Wycoff (VT)

UQSay seminar

March 21, 2019

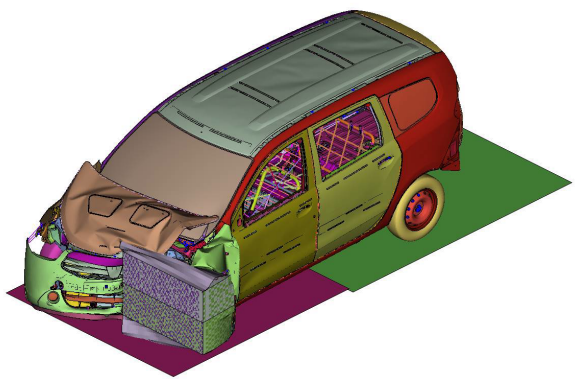
Problem description

Let us consider a time-consuming **black-box** simulator $f : \mathbb{R}^d \to \mathbb{R}$.

Aim: building a regression model of f given a set of observations $\mathbf{Y} = (y_1, \ldots, y_N)^\top$ at design locations $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$ when:

- observations are noisy, low signal to noise ratio
- noise variance is varying across $\mathbb{R}^d$

Examples of stochastic simulators:



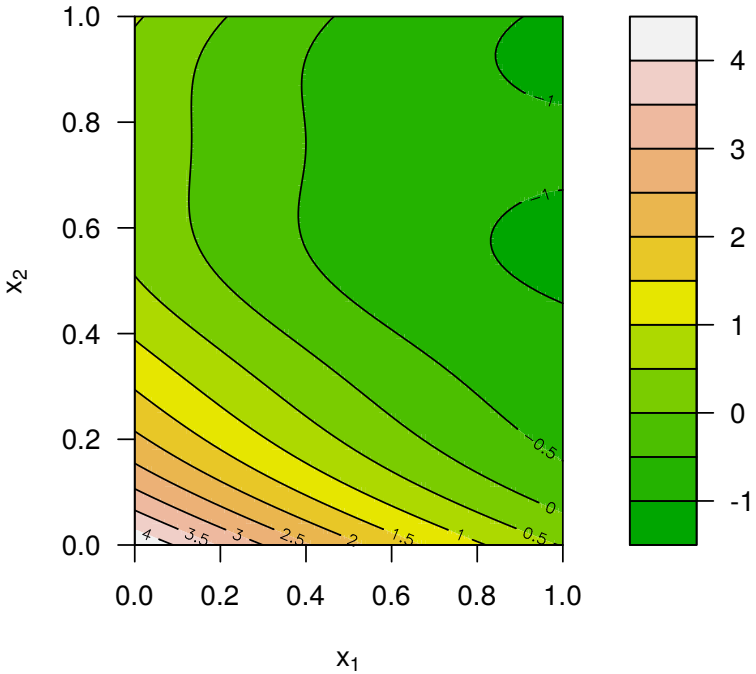
Car crash-worthiness

Cosmology

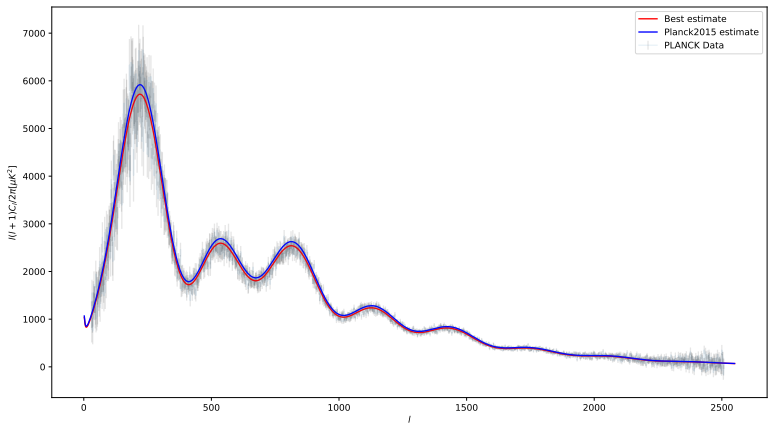
Many examples in physics, operations research, epidemiology, ML, ...

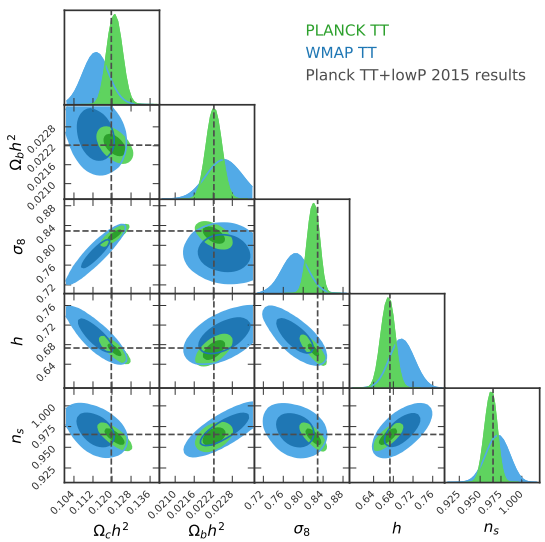
Applications

Optimization or safety



Calibration





Also: sensitivity analysis, dimension reduction,...

Outline

Outline

Gaussian processes (GPs)

GPs make popular surrogates because their predictions

- are rarely beaten in out-of-sample tests,
- have appropriate coverage (and can interpolate).

Definition (Gaussian vector)

A d-dimensional random vector Y is Gaussian iif $\forall \mathbf{a} \in \mathbb{R}^d$, $\mathbf{a}^\top Y$ is Gaussian.

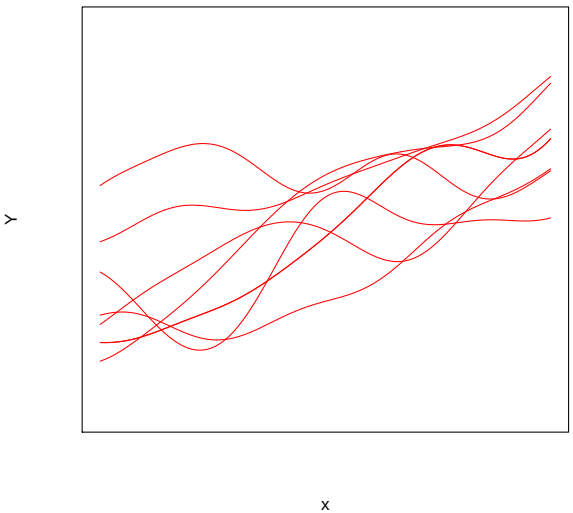
Definition (Gaussian process)

A random process Y indexed by D is said to be Gaussian iif $\forall \mathbf{x}_i \in D, \forall n \in \mathbb{N}$, $(Y(\mathbf{x}_1), \ldots, Y(\mathbf{x}_n))$ is a Gaussian vector.

GPs are fully characterized with their mean and covariance functions.

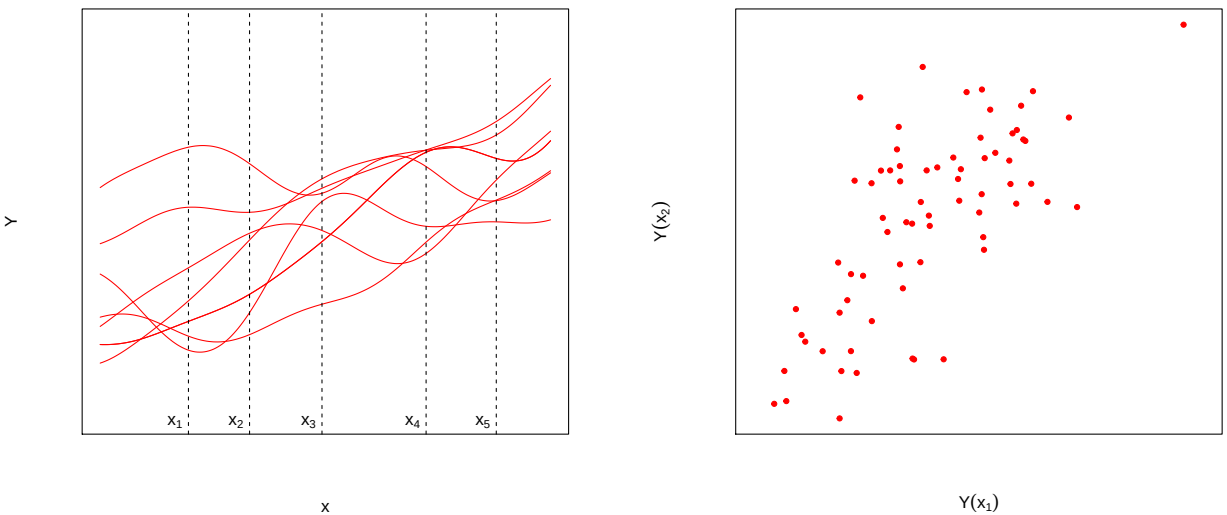
Gaussian processes (GPs)

Same with images:



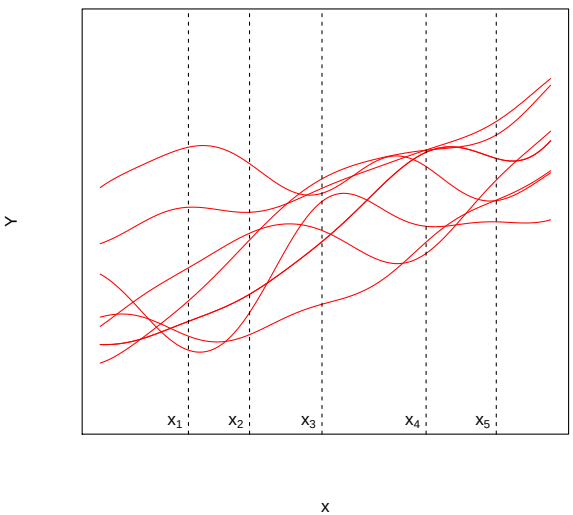
Gaussian processes (GPs)

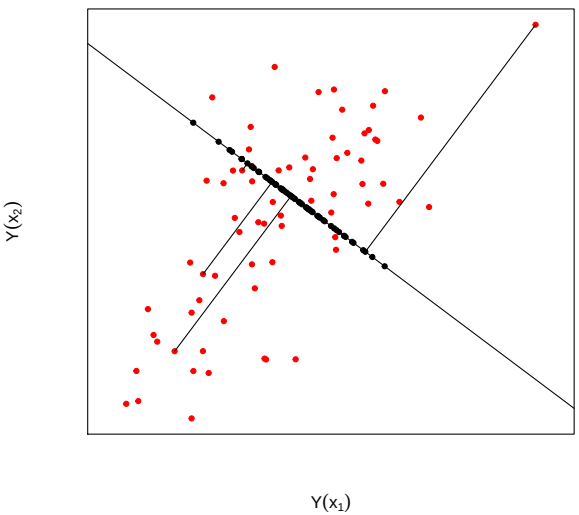
Same with images:



Gaussian processes (GPs)

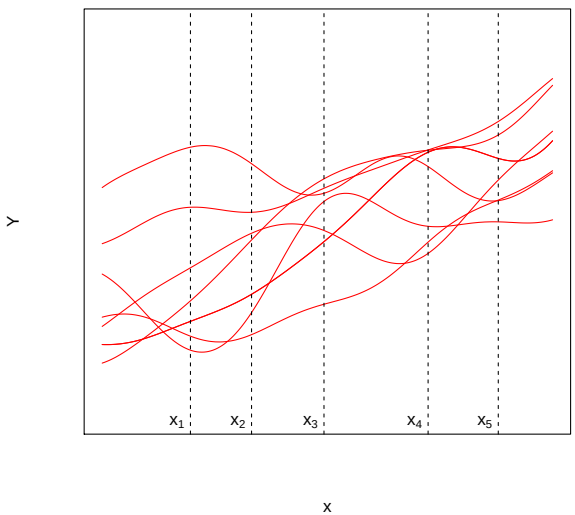
Same with images:

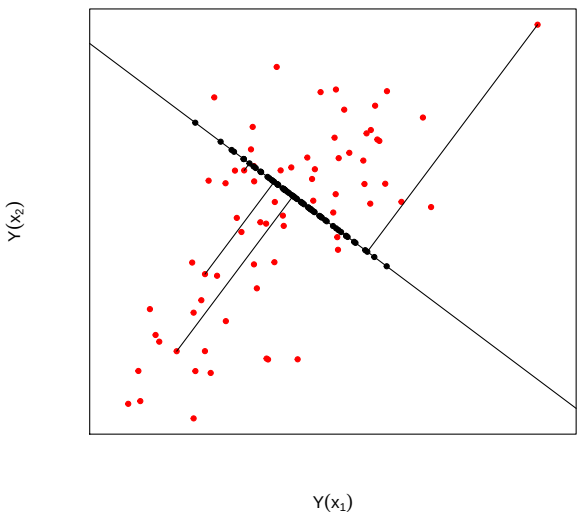


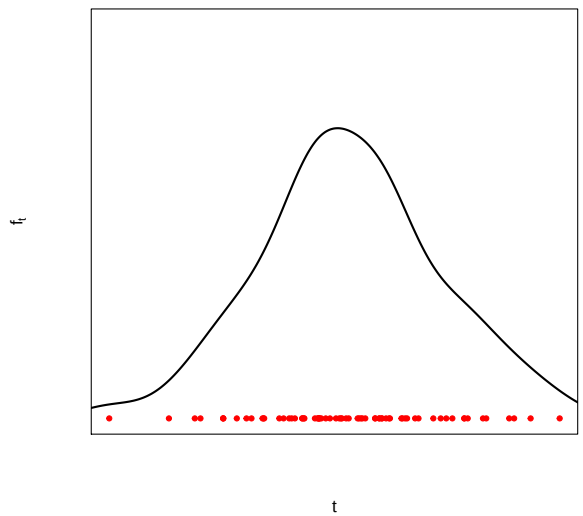


Gaussian processes (GPs)

Same with images:







Gaussian process regression

Observation model: $y(\mathbf{x}_i) = f(\mathbf{x}_i) + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, r(\mathbf{x}_i))$

For a zero mean GP with kernel k, MVN conditional identities give:

$$Y|\mathbf{Y} \sim \mathcal{GP}(\mu, \sigma^2) \text{ with}$$

$$\mu(\mathbf{x}) = \mathbb{E}(Y(\mathbf{x})|\mathbf{Y}) = \mathbf{k}(\mathbf{x})^\top (\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{Y},$$
$$\sigma^2(\mathbf{x}) = \mathbb{V}\mathrm{ar}(Y(\mathbf{x})|\mathbf{Y}) = k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x})^\top (\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{k}(\mathbf{x}) + r(\mathbf{x})$$

where $\mathbf{Y} = (y(\mathbf{x}_i))_{1\leq i\leq N}^\top$, $\mathbf{k}(\mathbf{x}) = (k(\mathbf{x},\mathbf{x}_i))_{1\leq i\leq N}^\top$,
$\mathbf{K}_N = (k(\mathbf{x}_i,\mathbf{x}_j))_{1\leq i,j\leq N}$, $\mathbf{\Sigma}_N = \mathrm{Diag}(r(\mathbf{x}_1),\ldots,r(\mathbf{x}_N))$

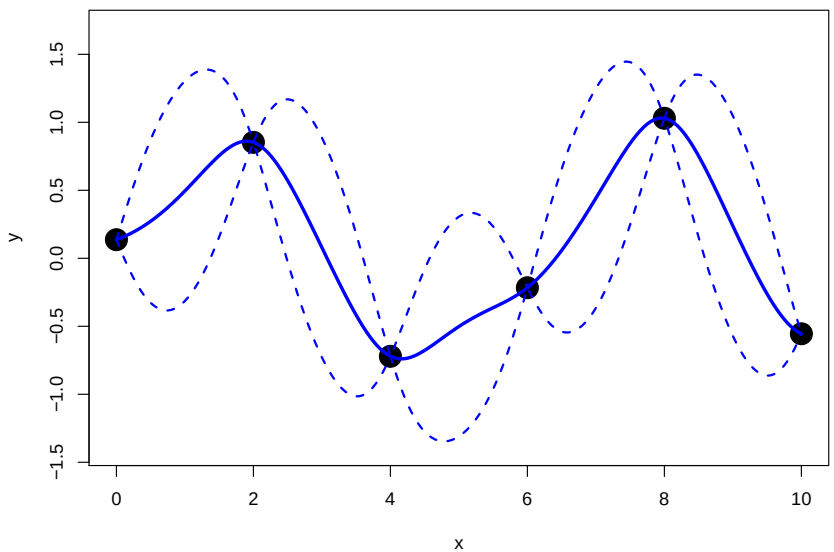
Remark: interest also in $P(y(\mathbf{x})|data)$, not only $P(f(\mathbf{x})|data)$

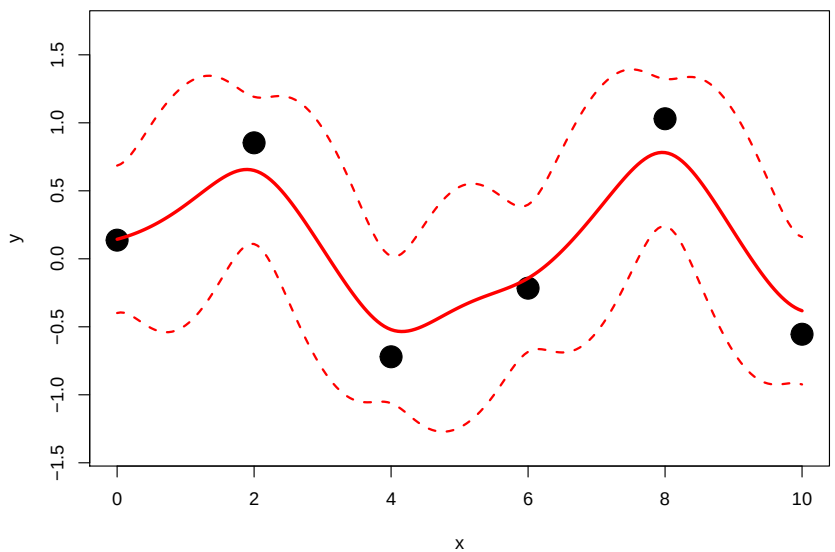
Gaussian process regression (2)

$$\boldsymbol{\Sigma}_N = \text{Diag}(\tau^2)$$

$$\mu(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top \mathbf{K}_N^{-1} \mathbf{Y},$$
$$\sigma^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^\top \mathbf{K}_N^{-1} \mathbf{k}(\mathbf{x})$$

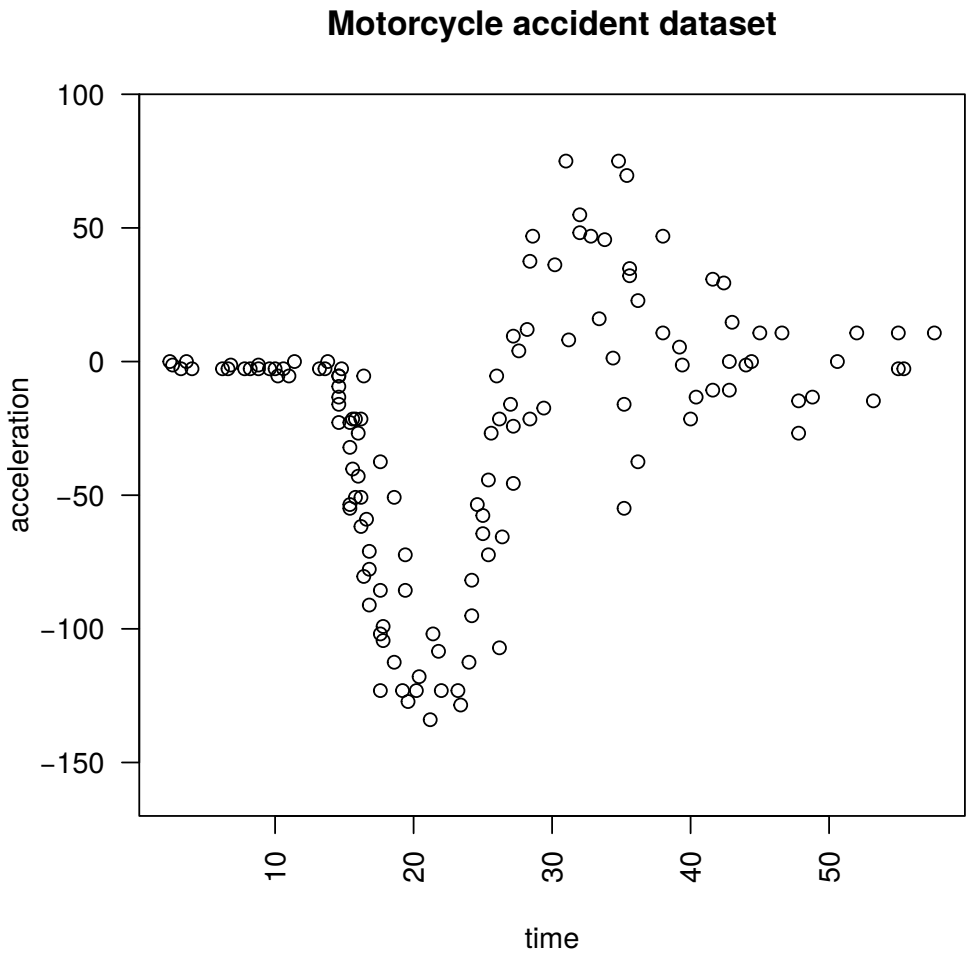
$$\mu(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (\mathbf{K}_N + \boldsymbol{\Sigma}_N)^{-1} \mathbf{Y},$$
$$\sigma^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) + \tau^2 - \mathbf{k}(\mathbf{x})^\top (\mathbf{K}_N + \boldsymbol{\Sigma}_N)^{-1} \mathbf{k}(\mathbf{x})$$





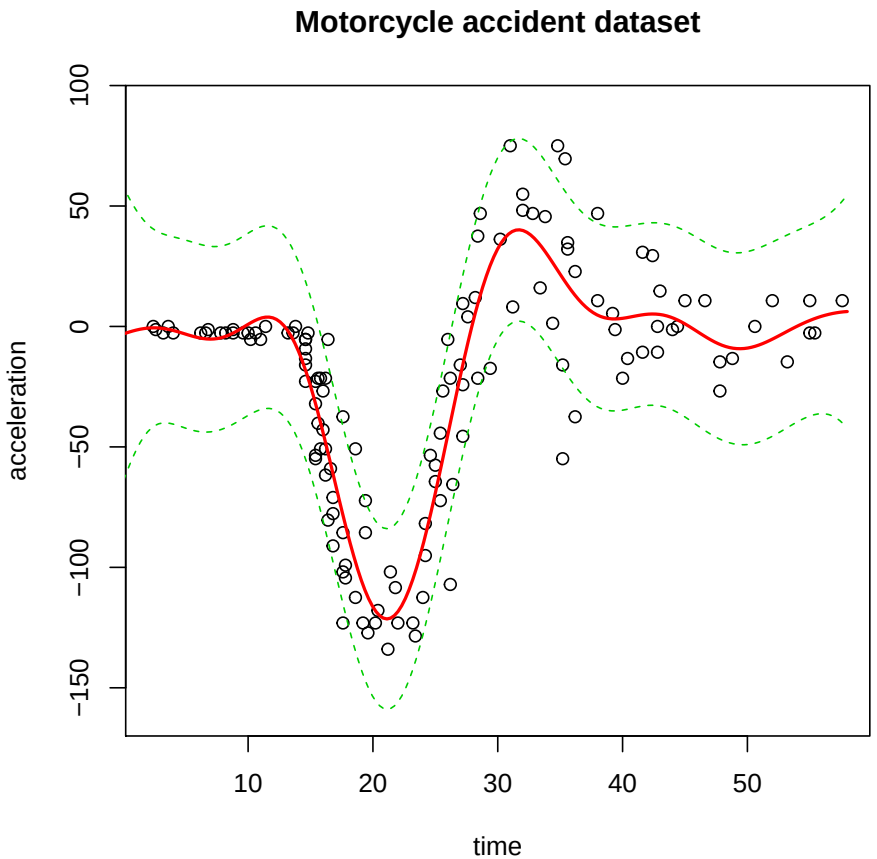
Motivating example for heteroskedasticity

Silverman (1985)'s motorcycle accident data



Motivating example for heteroskedasticity

Silverman (1985)'s motorcycle accident data



Gaussian process regression results with estimated constant noise:
$\rightarrow$ predictive mean is fine, but predictive variance is not.

Parameter estimation

Typically, the covariance kernel belongs to a parametric family (e.g., Gaussian or Matérn).

Estimation of the corresponding hyperparameters based either on:

- model error (i.e., cross validation, training/testing sets)
- variogram analysis
- **likelihood**

Likelihood, i.e., multivariate normal density:

$$L = \frac{1}{(2\pi)^{N/2}|\mathbf{K}_N + \mathbf{\Sigma}_N|^{1/2}} \exp\left(-\frac{1}{2}\mathbf{Y}^\top(\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{Y}\right).$$

This gives for the log-likelihood:

$$\log(L) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log(|\mathbf{K}_N + \mathbf{\Sigma}_N|) - \frac{1}{2}\mathbf{Y}^\top(\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{Y}$$

Parameter estimation (cont'd)

With stationary kernels, we use the following parameterization:

$$k(\mathbf{x}, \mathbf{x}') = \nu c(\mathbf{x} - \mathbf{x}'; \boldsymbol{\theta}) \text{ with } \nu \text{ the process variance}$$

Rewrite: $\mathbf{K}_N + \boldsymbol{\Sigma}_N = \nu(\mathbf{C}_N + \boldsymbol{\Lambda}_N)$, giving a plug-in estimator of ν:

$$\hat{\nu} = N^{-1}\mathbf{Y}^\top(\mathbf{C}_N + \boldsymbol{\Lambda}_N)^{-1}\mathbf{Y}$$

Concentrated log-likelihood:

$$\log(L) = -\frac{N}{2}\log(2\pi) - \frac{N}{2}\log\hat{\nu} - \frac{1}{2}\log|\mathbf{C}_N + \boldsymbol{\Lambda}_N| - N/2$$

Limitations: $\mathcal{O}(N^2)$ storage and $\mathcal{O}(N^3)$ computational complexity

But we need N to be large when the signal-to-noise ratio is low...
...and in addition, $r(\mathbf{x})$ is seldom known.

First option: using replicates

Now suppose that replication is present, with repeated design sites:

- provides a powerful tool to separate signal from noise.

Additional notations:

- $\bar{\mathbf{x}}_i$, $1 \leq i \leq n$ unique input locations, potentially $n \ll N$
- $y_i^{(j)}$ j^{th} out of $a_i \geq 1$ replicates collected at $\bar{\mathbf{x}}_i$
- $\bar{\mathbf{Y}} = (\bar{y}_1, \ldots, \bar{y}_n)^\top$ averages of replicates, $\bar{y}_i = \frac{1}{a_i} \sum_{j=1}^{a_i} y_i^{(j)}$
- $\boldsymbol{\Sigma}_n = \text{Diag}(r(\bar{\mathbf{x}}_1)/a_1, \ldots, r(\bar{\mathbf{x}}_n)/a_n)$
- $\widehat{\boldsymbol{\Sigma}}_n = \text{Diag}(\hat{\sigma}_1^2/a_1, \ldots, \hat{\sigma}_n^2/a_n)$, where $\hat{\sigma}_i^2 = \frac{1}{a_i - 1} \sum_{j=1}^{a_i} (y_i^{(j)} - \bar{y}_i)^2$

Generally, we use the n lower script for averaged quantities.

Stochastic kriging (SK) (Ankennman et al., 2010)

Show that the predictive equations using $\widehat{\mathbf{\Sigma}}_n$:

$$\mu_n(\mathbf{x}) = \mathbf{k}_n(\mathbf{x})^\top (\mathbf{K}_n + \widehat{\mathbf{\Sigma}}_n)^{-1} \bar{\mathbf{Y}}$$
$$\sigma_n^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}_n(\mathbf{x})^\top (\mathbf{K}_n + \widehat{\mathbf{\Sigma}}_n)^{-1} \mathbf{k}_n(\mathbf{x}) + r(\mathbf{x})$$

are asymptotically unbiased and MSE-optimal.

Corresponding log-likelihood is:

$$\log \bar{L} := -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{K}_n + \widehat{\mathbf{\Sigma}}_n| - \frac{1}{2} \bar{\mathbf{Y}}^\top (\mathbf{K}_n + \widehat{\mathbf{\Sigma}}_n)^{-1} \bar{\mathbf{Y}}$$

Pros:
- $\mathcal{O}(n^3)$ complexity, huge potential savings

Cons:
- requires a minimum amount of replication
- do not provide out-of-sample variance predictions

Second option: using latent variables

Idea: modeling *jointly* the (log-)variance by a second GP

- assumes smoothly varying noise across the input space
- introduces latent variables (log-variances)

But, the posterior on the joint process is intractable.

Existing approaches:

- full MCMC computation (Goldberg et al., 1998)
- hard-EM approximation (Kersting et al., 2007)
- hard-EM corrections for replicates (Boukouvalas et al., 2009)
- variational approximation (Lazaro-Gredilla et al., 2011)

They do not require replicates, but do not exploit them fully either.
And EM and MCMC brings a computational burden.
$\Rightarrow$ hybrid method with SK

Mapping large-N quantities to small-n ones

Similarly to SK, we exploit the structure coming from replication:

$$\mathbf{X} = (\bar{\mathbf{x}}_1, \dots, \bar{\mathbf{x}}_1, \dots \bar{\mathbf{x}}_n)^\top = \mathbf{U}\bar{\mathbf{X}}, \text{ with } \mathbf{U} = \mathit{Diag}(\mathbf{1}_{a_1,1}, \dots, \mathbf{1}_{a_n,1})$$

the $N \times n$ block matrix, where $\mathbf{1}_{k,l}$ the $k \times l$ matrix of ones.

It applies to all quantities, e.g., $\mathbf{K}_N = \mathbf{U}\mathbf{K}_n\mathbf{U}^\top$ and $\mathbf{\Sigma}_N = \mathbf{U}\mathbf{\Sigma}_n\mathbf{U}^\top$.

Motivates the use of familiar identities:

Woodbury and matrix determinant formulas

$$(\mathbf{D} + \mathbf{UBV})^{-1} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{U}(\mathbf{B}^{-1} + \mathbf{V}\mathbf{D}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{D}^{-1}$$
$$\det(\mathbf{D} + \mathbf{UBV}) = \det(\mathbf{B}^{-1} + \mathbf{V}\mathbf{D}^{-1}\mathbf{U}) \det(\mathbf{B}) \det(\mathbf{D})$$

with $\mathbf{D} \in \mathbb{R}^{N\times N}$ and $\mathbf{B} \in \mathbb{R}^{n\times n}$ are invertible, and $\mathbf{U}$, $\mathbf{V}^\top \in \mathbb{R}^{N\times n}$

Here: $\mathbf{D} = \mathbf{\Sigma}_N$, $\mathbf{B} = \mathbf{K}_N$, $\mathbf{V} = \mathbf{U}^\top$, $\mathbf{U}^\top\mathbf{U} = \mathrm{Diag}(a_1, \dots, a_n) = \mathbf{A}_n$

Large-N to little-n GP predictive equations

Recall the covariance parameterization: $\mathbf{K}_N + \mathbf{\Sigma}_N = \nu(\mathbf{C}_N + \mathbf{\Lambda}_N)$

The Woodbury identity directly gives :

$$\mathbf{k}_N(\mathbf{x})^\top(\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{Y} = \mathbf{c}_n(\mathbf{x})^\top(\mathbf{C}_n + \mathbf{\Lambda}_n\mathbf{A}_n^{-1})^{-1}\bar{\mathbf{Y}}$$
$$\mathbf{k}_N(\mathbf{x})^\top(\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{k}_N(\mathbf{x}) = \nu\mathbf{c}_n(\mathbf{x})^\top(\mathbf{C}_n + \mathbf{\Lambda}_n\mathbf{A}_n^{-1})^{-1}\mathbf{c}_n(\mathbf{x})$$

Reduced equations requires $\mathcal{O}(n^3)$ time instead of $\mathcal{O}(N^3)$.

They inherit all the properties of the full-N ones.

What about the likelihood?

Reduced concentrated log likelihood

Lemma

Let $\boldsymbol{\Upsilon}_n := \mathbf{C}_n + \mathbf{A}_n^{-1}\boldsymbol{\Lambda}_n$. *Then the little-n identity for the large-N expression for the concentrated log likelihood is*

$$\log L = cst. - \frac{N}{2}\log\hat{\nu}_N - \frac{1}{2}\sum_{i=1}^{n}\left[(a_i - 1)\log\lambda_i + \log a_i\right] - \frac{1}{2}\log|\boldsymbol{\Upsilon}_n|$$

where $\hat{\nu}_N := N^{-1}\left(\mathbf{Y}^\top\boldsymbol{\Lambda}_N^{-1}\mathbf{Y} - \bar{\mathbf{Y}}^\top\mathbf{A}_n\boldsymbol{\Lambda}_n^{-1}\bar{\mathbf{Y}} + \bar{\mathbf{Y}}^\top\boldsymbol{\Upsilon}_n^{-1}\bar{\mathbf{Y}}\right).$

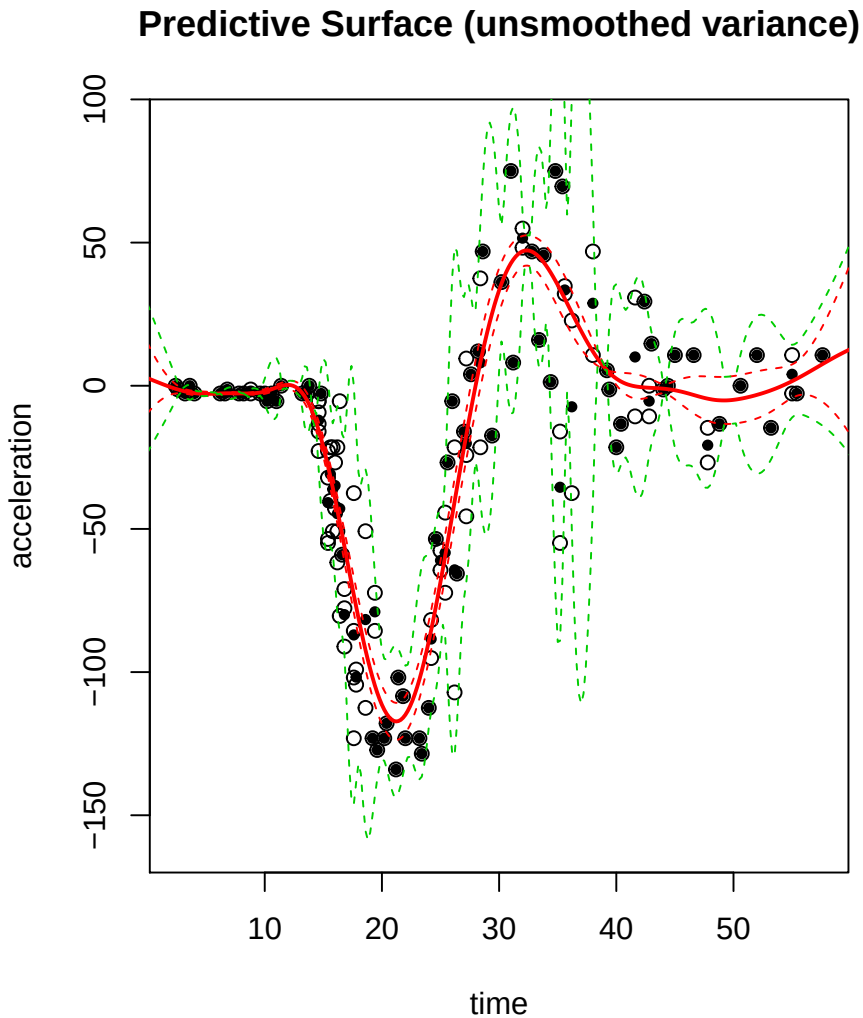
Still requires $\mathcal{O}(n^3)$ time and allow closed-form derivatives:

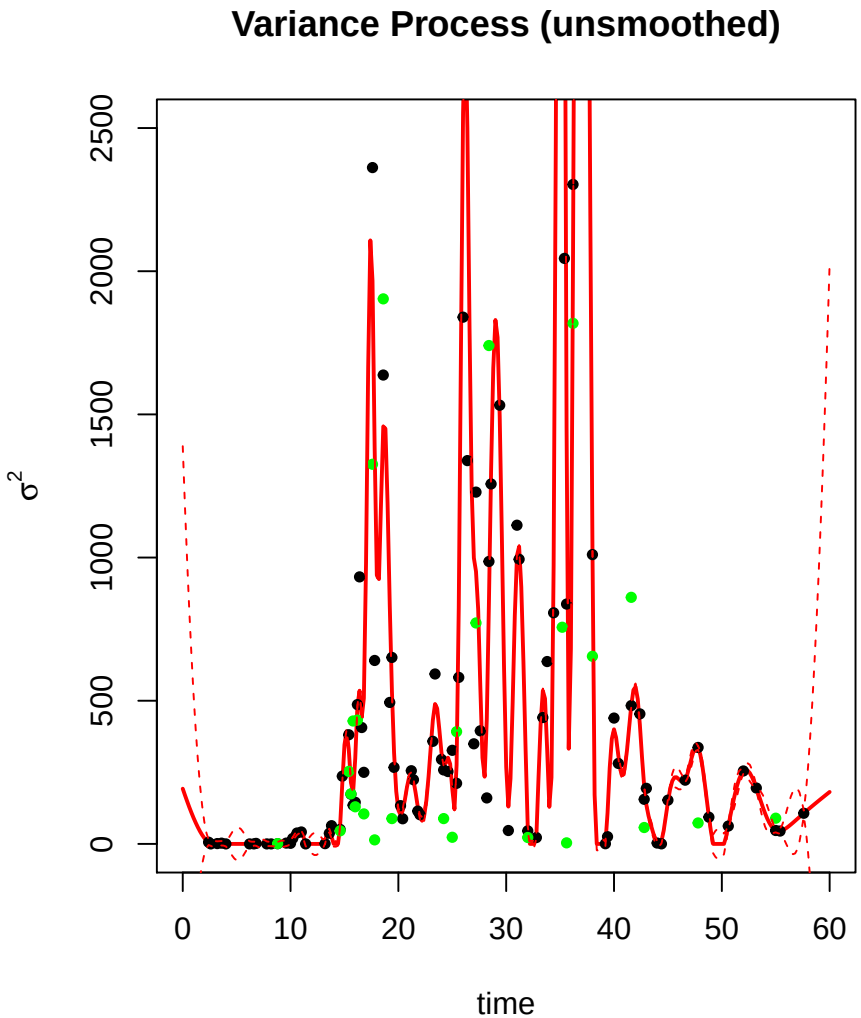
$$\frac{\partial\log L}{\partial\cdot} = \frac{N}{2}\frac{\partial\left(\mathbf{Y}^\top\boldsymbol{\Lambda}_N^{-1}\mathbf{Y} - \bar{\mathbf{Y}}\mathbf{A}_n\boldsymbol{\Lambda}_n^{-1}\bar{\mathbf{Y}} + n\hat{\nu}_n\right)}{\partial\cdot} \times (N\hat{\nu}_N)^{-1}$$
$$- \frac{1}{2}\sum_{i=1}^{n}\left[(a_i - 1)\frac{\partial\log\lambda_i}{\partial\cdot}\right] - \frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Upsilon}_n^{-1}\frac{\partial\boldsymbol{\Upsilon}_n}{\partial\cdot}\right)$$

Outline

Learning the latent

Direct optimization of $\mathbf{\Lambda}_n$?





$N = 133, n = 94$

$\hat{\nu}_n \mathbf{\Lambda}_n$: black points, $\widehat{\mathbf{\Sigma}}_n$: green points

Second GP

We borrow the machine learning idea of using a second GP on the noise variance:

$$\mathbf{\Lambda}_n = \mathbf{C}_{(g)}(\mathbf{C}_{(g)} + g\mathbf{A}_n^{-1})^{-1}\mathbf{\Delta}_n$$

(smoothing of latent $\mathbf{\Delta}_n = \text{Diag}(\delta_1, \ldots, \delta_n)$ values)

Additional hyperparameters: lengthscale ϕ, nugget g

Remains tractable:

$$\frac{\partial \log L}{\partial \mathbf{\Delta}_n} = \frac{\partial \mathbf{\Lambda}_n}{\partial \mathbf{\Delta}_n}\frac{\partial \log L}{\partial \mathbf{\Lambda}_n} = \mathbf{C}_{(g)}(\mathbf{C}_{(g)} + g\mathbf{A}_n^{-1})^{-1}\frac{\partial \log L}{\partial \mathbf{\Lambda}_n}, \quad \text{where}$$

$$\frac{\partial \log L}{\partial \lambda_i} = \frac{N}{2} \times \frac{\frac{a_i s_i^2}{\lambda_i^2} + \frac{\bar{\mathbf{Y}}^\top \mathbf{\Upsilon}_n^{-1}\mathbf{\Upsilon}_n^{-1}\bar{\mathbf{Y}}}{a_i}}{\hat{\nu}_N} - \frac{a_i - 1}{2\lambda_i} - \frac{1}{2a_i}(\mathbf{\Upsilon}_n)^{-1}_{i,i}$$

Optional: replacing $\mathbf{\Lambda}_n$ with $\exp(\mathbf{\Lambda}_n)$ to ensure positivity

Joint optimization of the full set of hyperparameters: $\{\boldsymbol{\theta}, \phi, \boldsymbol{\Delta}_n, g\}$

Objective function is the *concentrated* joint log-likelihood:

$$\log \tilde{L} = -\frac{N}{2}\log \hat{\nu}_N - \frac{1}{2}\sum_{i=1}^{n}\left[(a_i - 1)\log \lambda_i + \log a_i\right] - \frac{1}{2}\log|\boldsymbol{\Upsilon}_n|$$
$$-\frac{n}{2}\log \hat{\nu}_{(g)} - \frac{1}{2}\log|\mathbf{C}_{(g)} + g\mathbf{A}_n^{-1}| + \text{cst.}$$

with closed-form derivatives once a form for $\mathbf{C}_n$ and $\mathbf{C}_{(g)}$ is chosen.

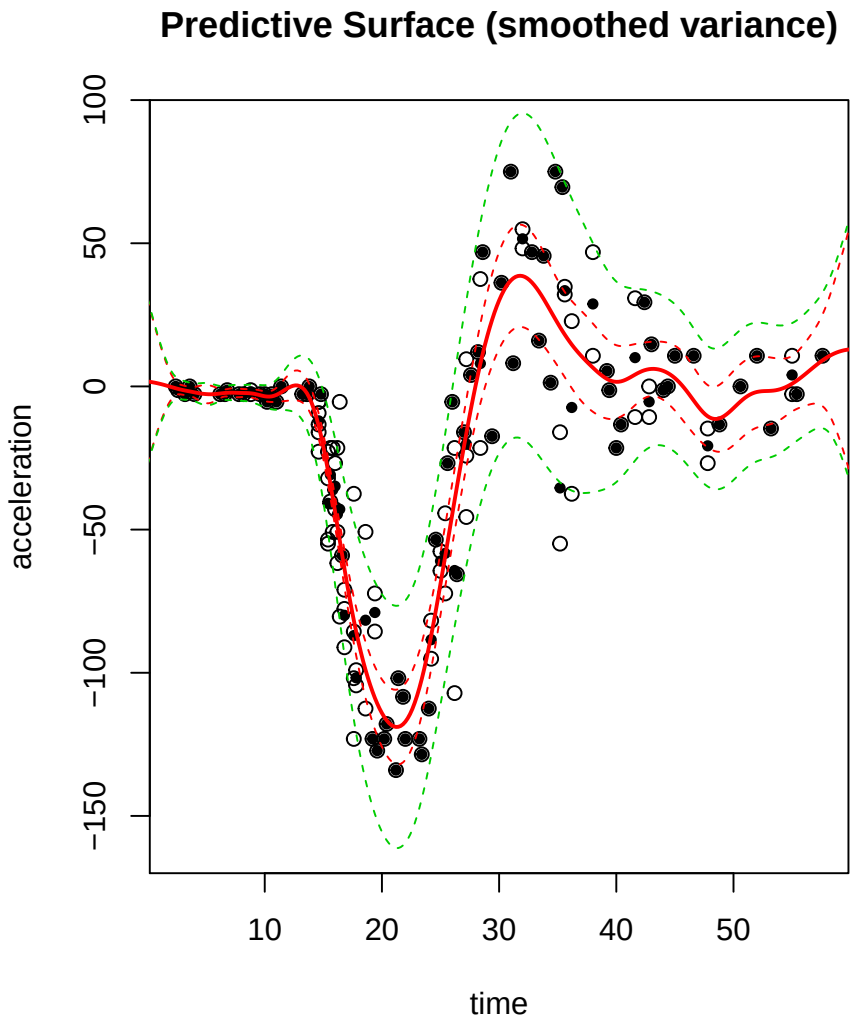
Allows testing for heteroskedasticity.

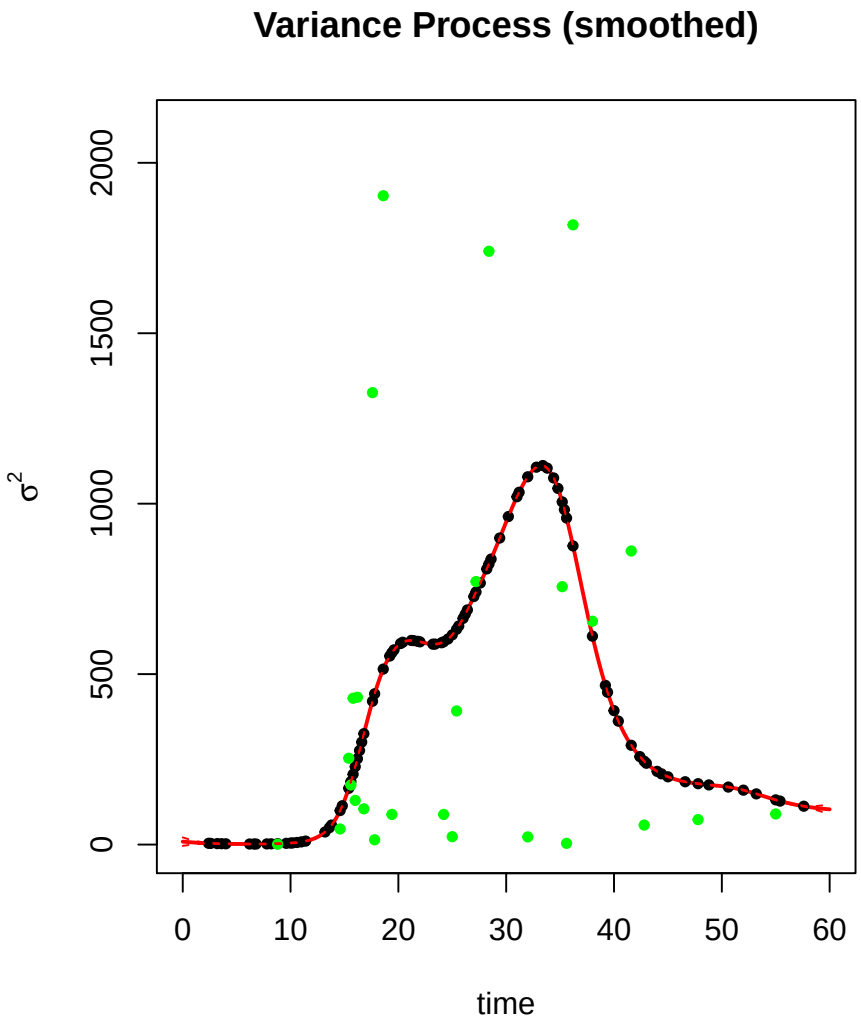
Interestingly, smoothing is for naught: $\tilde{L}$ is maximized when $g = 0$.

But, this ad-hoc smoothing is a useful device in three ways:

- connects SK to Goldberg's latent representation
- eases the optimization with an annealing effect
- yields a smooth solution if optimization is stopped prematurely

Example 1: motorcycle data





$\hat{\nu}_n \mathbf{\Lambda}_n$: black points, $\widehat{\mathbf{\Sigma}}_n$: green points

Example 2: Assemble to order (Hong et al., 2006; Xie et al., 2012)

Inventory management problem:

- 5 products are produced, requiring some of 8 different items
- sold products bring profit, storing items have a cost
- orders come randomly over a time period
- random replenishment of items
- variables are the target stock of each item ($[0, 20]^8$)
- output is the profit per unit time

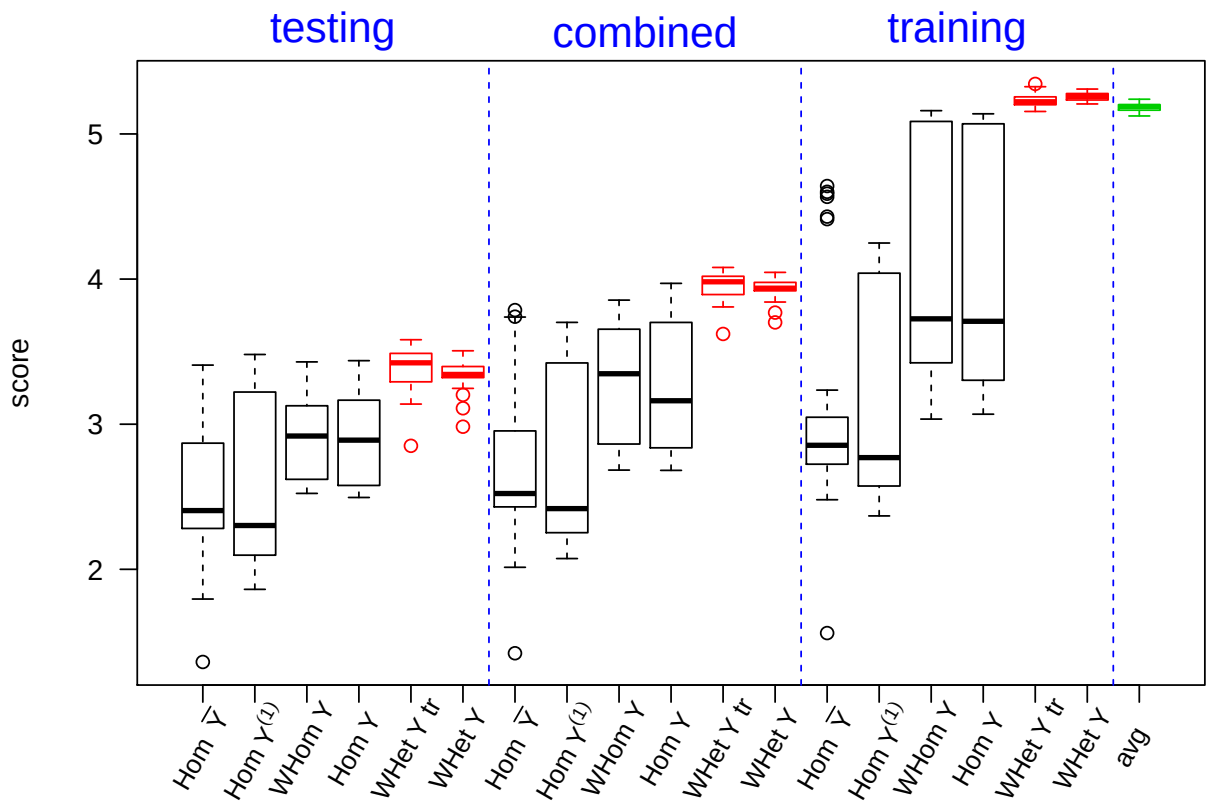
Experiment:

- full data is a Latin Hypercube sample of size 2000, with 10 replicates each
- training data is 1000 designs, with uniformly sampled number of replicates
- testing data is the other half, and remaining replicates

Example 2: Assemble to order (Hong et al., 2006; Xie et al., 2012)

Results focusing on the mean accuracy relative to predicted variance based on a proper scoring rule (Gneiting et al., 2007):

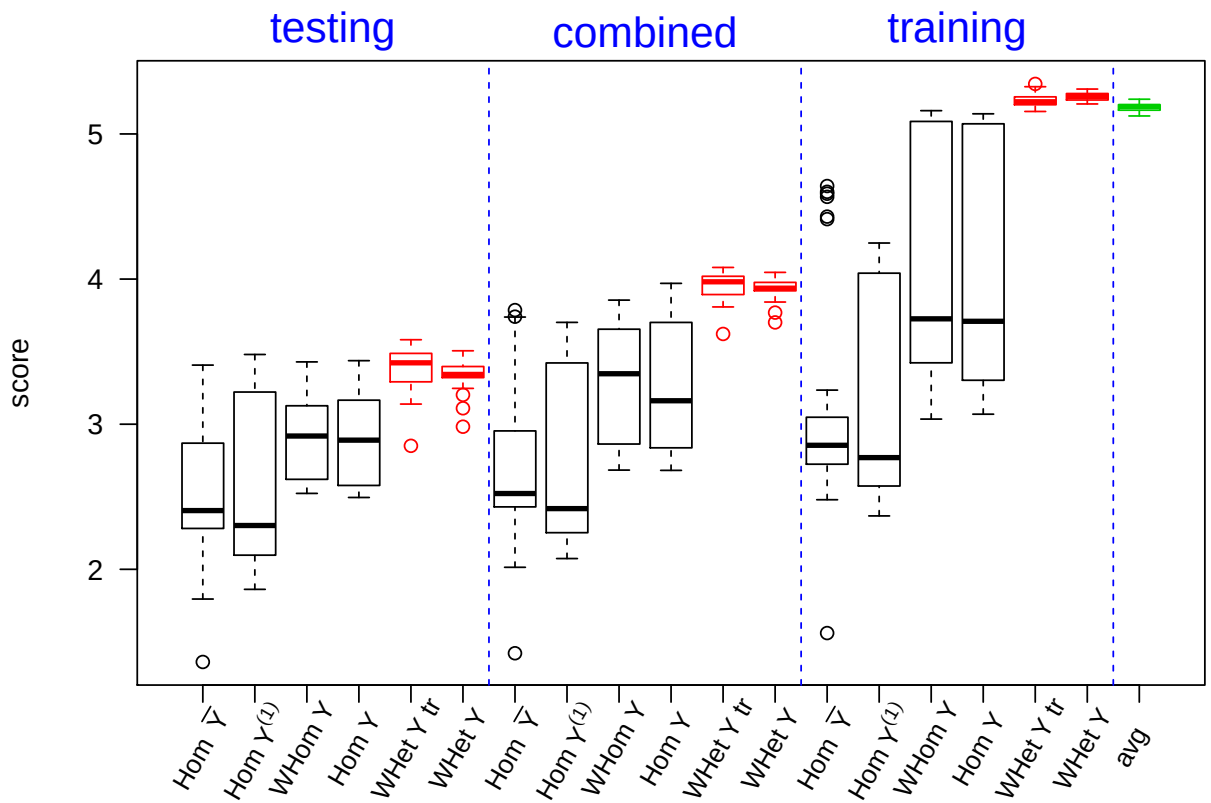
$$S(P, y) = -\left(\frac{y - \mu_P}{\sigma_P}\right)^2 - log(\sigma_P^2)$$

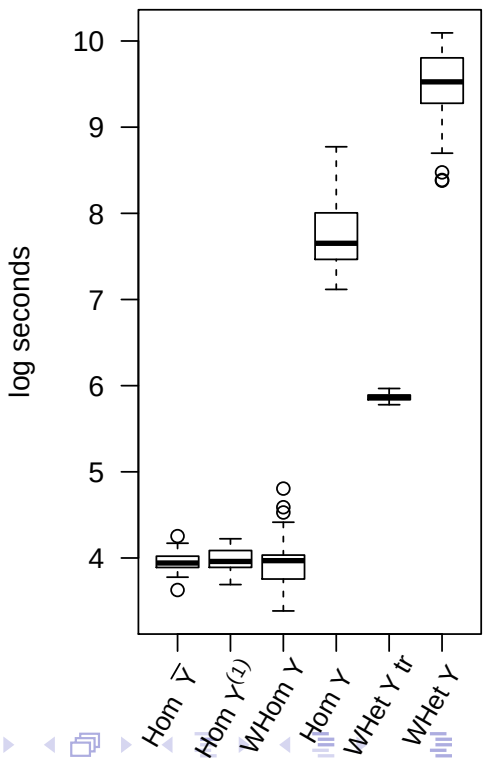


Example 2: Assemble to order (Hong et al., 2006; Xie et al., 2012)

Results focusing on the mean accuracy relative to predicted variance based on a proper scoring rule (Gneiting et al., 2007):

$$S(P, y) = -\left(\frac{y - \mu_P}{\sigma_P}\right)^2 - log(\sigma_P^2)$$





Example 3: Epidemic management (Hu et al., 2015)

Study disease outbreak dynamics based on stochastic compartmental modeling:

- Susceptible, Infected, Recovered (SIR) counts
- The continuous time state (S_t, I_t, R_t) is a Markov chain, with transition $S + I \to 2I$ and $I \to R$
- considered output is the total number of newly infected:

$$f(\mathbf{x}) := \mathbb{E}[S_0 - \lim_{T\to\infty} S_T | (S_0, I_0, R_0) = \mathbf{x}] = \gamma \mathbb{E}\big[\int_0^\infty I_t \, dt | \mathbf{x}\big]$$

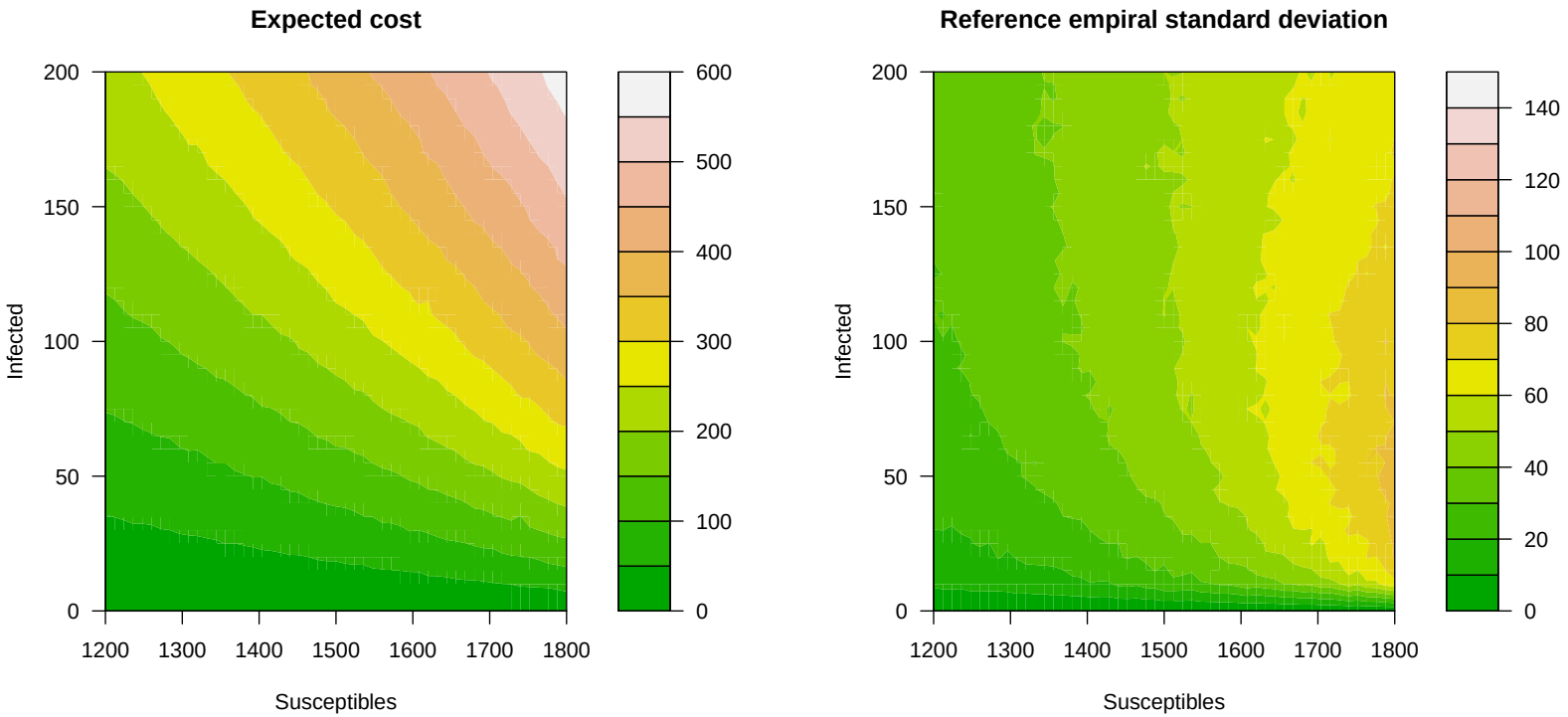
  estimated by Monte Carlo

Experiments:

- total population $M = 2000$
- testing set is 2000 designs on the grid, 100 replicates
- training set is 1000 designs, 500 with 5 replicates, 250 with 10, 150 with 50, 100 with 100

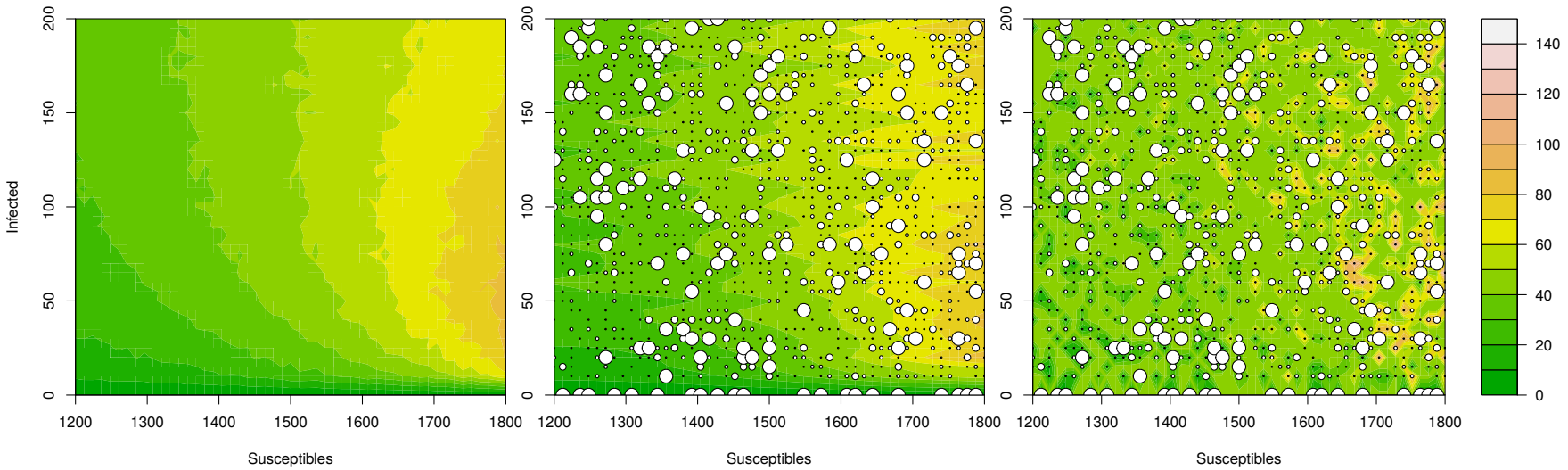
Example 3: Epidemic management (Hu et al., 2015)

Reference mean and noise surfaces



Example 3: Epidemic management (Hu et al., 2015)

Comparison of standard deviation estimations



(a) Reference set (b) Joint estimation (c) Empirical

Dot size indicates number of replicates

Outline

Generalities on experimental design

General design methods:

- space-filling designs (e.g., Latin hypercubes (LHS), maximin-LHS, minimax-LHS, maxPro-LHS),
- orthogonal arrays,
- sparse-grids,
- hybrids, ...

Model based design criteria:

- integrated mean square prediction error,
- maximum prediction error,
- entropy,
- Fisher information, ...

They can be optimized for design points all at once or sequentially.

Sequential design procedure

Sequential design framework

1. Construct initial space-filling design
2. While stopping criterion not met:
   1. Train GP model
   2. Enrich design with selected criterion

Preferred approach when hyperparameters are unknown.

Specifically for replicates, they are added

- by batches of fixed size (e.g., Boukouvalas et al., 2014)
- in a separate phase (e.g., Ankenman et al., 2010; Liu et al., 2010)

We aim for something more flexible, adding replicates on the fly, without specifying a batch size.

Integrated Mean Squared Prediction Error (IMSPE)

Here, we use IMSPE as a criterion:

$$IMSPE(\mathbf{X}) = \int_{\mathbf{x}\in\mathbf{D}} \sigma_N^2(\mathbf{x})d\mathbf{x}$$
$$= \int_{\mathbf{x}\in\mathbf{D}} k(\mathbf{x},\mathbf{x}) - \mathbf{k}_n(\mathbf{x})^\top(\mathbf{K}_n + \widehat{\mathbf{\Sigma}}_n)^{-1}\mathbf{k}_n(\mathbf{x})$$

Commonly approximated as a sum over a set of points.
Closed form expression e.g., for separable Matérn, Gaussian kernels.

Sequential version, adding $\mathbf{x}_{N+1}$:

$$IMSPE(\mathbf{X}, \mathbf{x}_{N+1}) = \int_{\mathbf{x}\in\mathbf{D}} \sigma_{N+1}^2(\mathbf{x})d\mathbf{x}$$

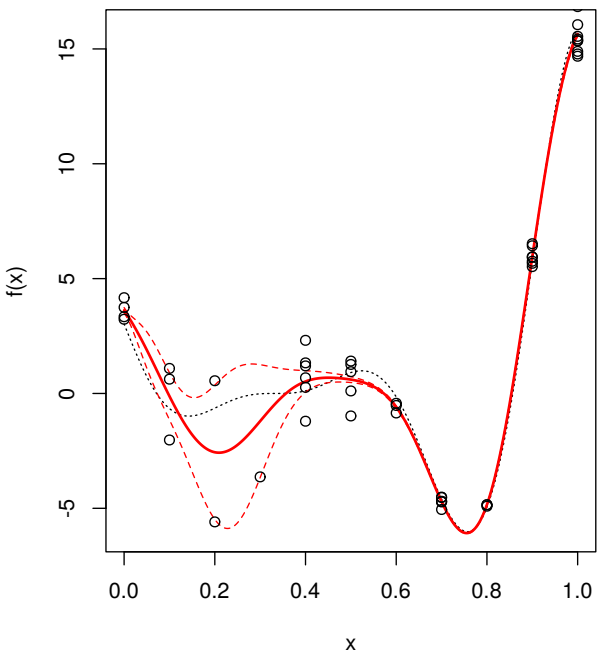
$$= \int_{\mathbf{x}\in\mathbf{D}} \sigma_N^2(\mathbf{x}) - \mathbf{k}_{\text{old},N}(\mathbf{x},\mathbf{x}_{N+1})^\top(\mathbf{K}_M + \mathbf{\Sigma}_{N+1})^{-1}\mathbf{k}_{\text{old},N}(\mathbf{x},\mathbf{x}_{N+1})d\mathbf{x}$$

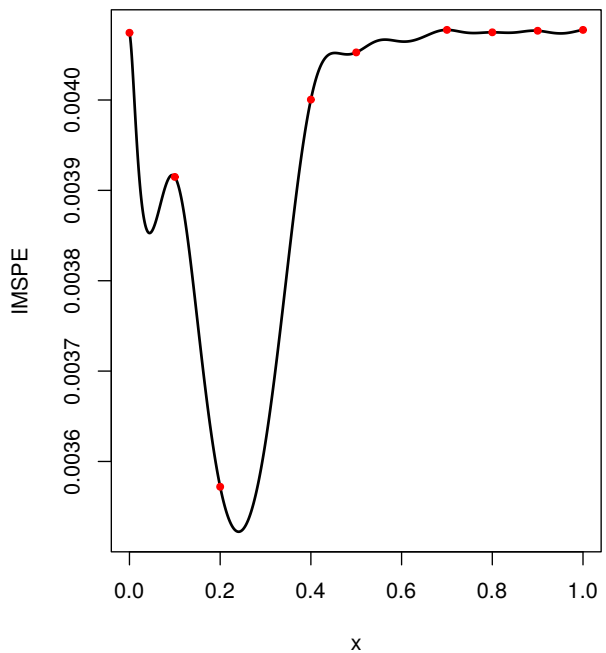
Why replicating?

In sequential design, there are a few cases where replication occurs:

1) The optimum is at an existing design
2) Optimization effect: local optimum not better than existing design

Example:





3) Rounding effect (or discrete search space)
4) Computational efficiency: faster update, discrete search

Sequential IMSPE

A) When adding a new $\tilde{\mathbf{x}}$:

$$I_{N+1}(\tilde{\mathbf{x}}) = I_N - \left(\sigma_n^2(\tilde{\mathbf{x}})\mathbf{g}(\tilde{\mathbf{x}})^\top \mathbf{W}_n \mathbf{g}(\tilde{\mathbf{x}}) + 2\mathbf{w}(\tilde{\mathbf{x}})^\top \mathbf{g}(\tilde{\mathbf{x}}) + \sigma_n^2(\tilde{\mathbf{x}})^{-1} w(\tilde{\mathbf{x}}, \tilde{\mathbf{x}})\right)$$

with $w(\mathbf{x}_i, \mathbf{x}_j) = \int_{\mathbf{x}\in D} k(\mathbf{x}_i, \mathbf{x}) k(\mathbf{x}_j, \mathbf{x})\, d\mathbf{x}$, $1 \leq i, j \leq n$ and
$\mathbf{g}(\tilde{\mathbf{x}}) = -\sigma_n^2(\tilde{\mathbf{x}})^{-1} \mathbf{K}_n^{-1} \mathbf{k}_n(\tilde{\mathbf{x}})$

Closed form expressions and **derivatives** are available (e.g., for separable Matérn, Gaussian kernels).

B) When replicating at $\mathbf{x}_k$, $1 \leq k \leq n$, we show that:

$$I_{N+1}(\bar{\mathbf{x}}_k) = I_N - \mathrm{tr}(\mathbf{B}_k \mathbf{W}_n)$$

with $\mathbf{B}_k = \frac{\left(\mathbf{K}_n^{-1}\right)_{.,k} \left(\mathbf{K}_n^{-1}\right)_{k,.}}{a_k(a_k+1)/r(\bar{\mathbf{x}}_k) - (\mathbf{K}_n)^{-1}_{k,k}}$, a rank-one matrix

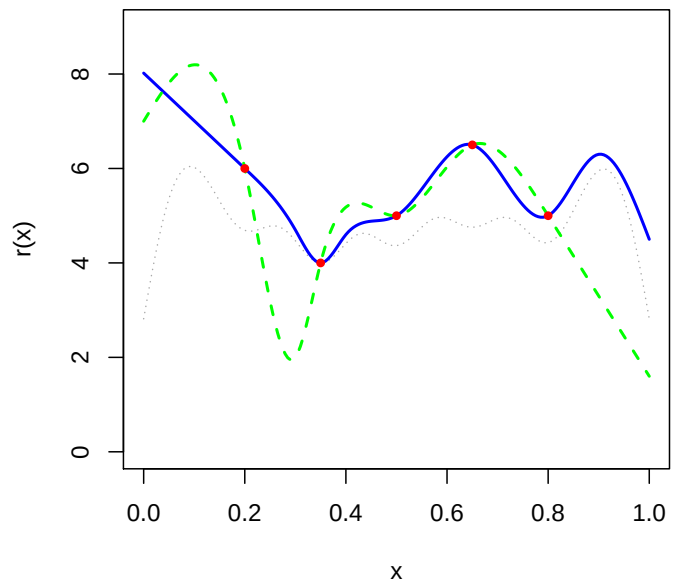
Replication: IMPSE optimal with heteroskedasticity

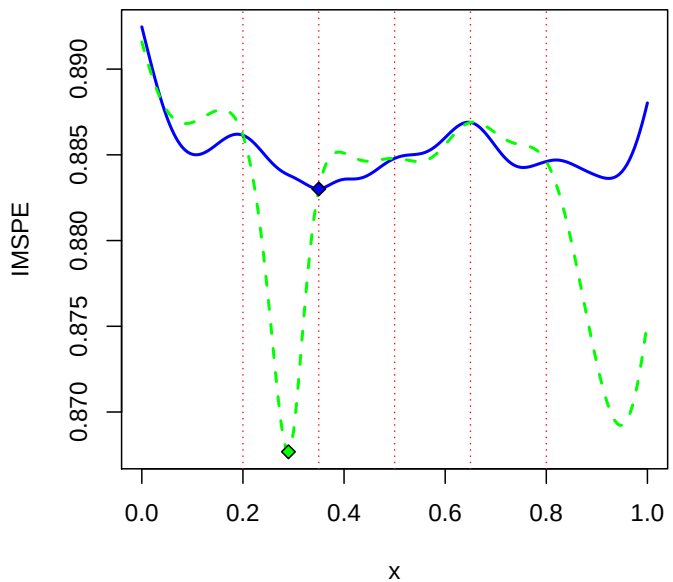
Proposition

Given unique design locations $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_n$, *replicating is optimal if* $\forall \tilde{\mathbf{x}} \in D$

$$r(\tilde{\mathbf{x}}) \geq \frac{\mathbf{k}(\tilde{\mathbf{x}})^\top \mathbf{K}_n^{-1} \mathbf{W}_n \mathbf{K}_n^{-1} \mathbf{k}(\tilde{\mathbf{x}}) - 2\mathbf{w}(\tilde{\mathbf{x}})^\top \mathbf{K}_n^{-1} \mathbf{k}(\tilde{\mathbf{x}}) + w(\tilde{\mathbf{x}}, \tilde{\mathbf{x}})}{\mathrm{tr}(\mathbf{B}_{k^*} \mathbf{W}_n)} - \check{\sigma}_n^2(\tilde{\mathbf{x}})$$

where $k^* \in \arg\min_{1 \leq k \leq n} I_{N+1}(\bar{\mathbf{x}}_k)$.



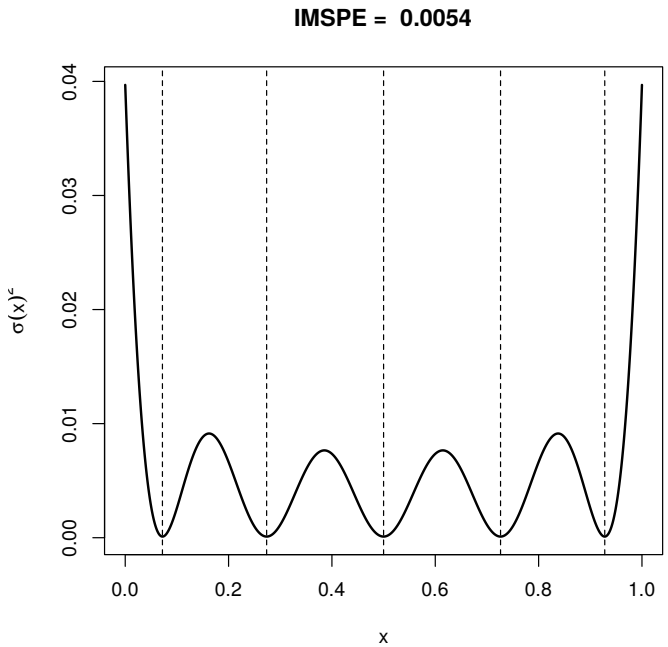


Also: discretization or optimization effects

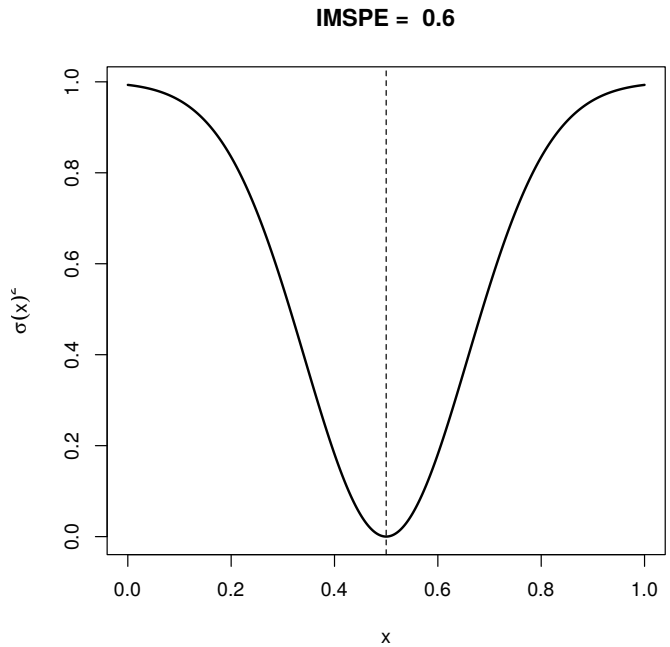
Myopic IMSPE

Like most criteria, IMSPE ignores the limited budget.

Example (fixed hyperparameters)



Non-sequential

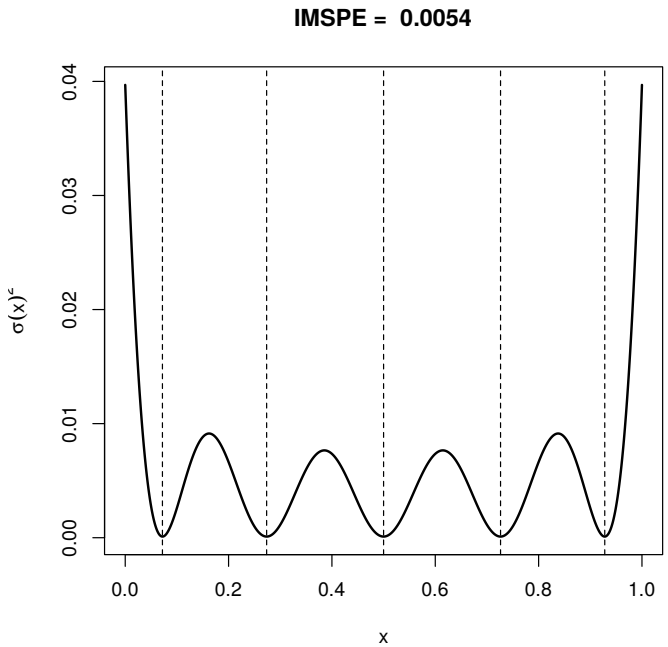


Sequential

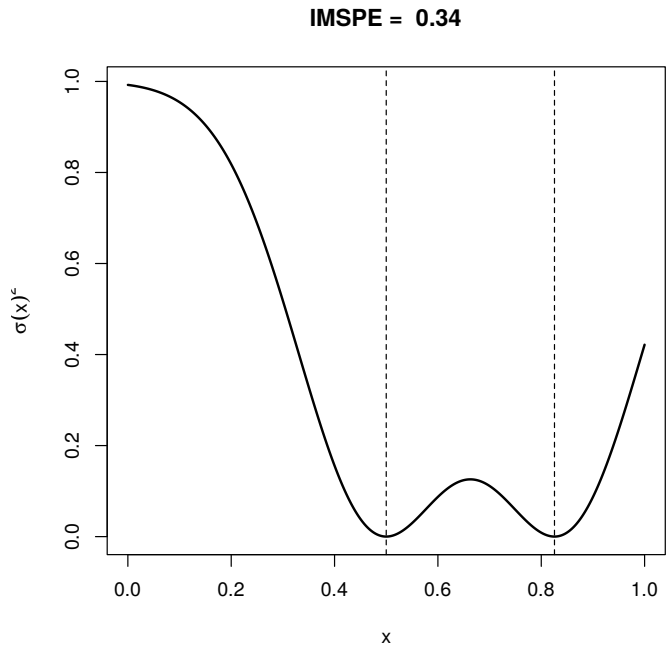
Myopic IMSPE

Like most criteria, IMSPE ignores the limited budget.

Example (fixed hyperparameters)



Non-sequential

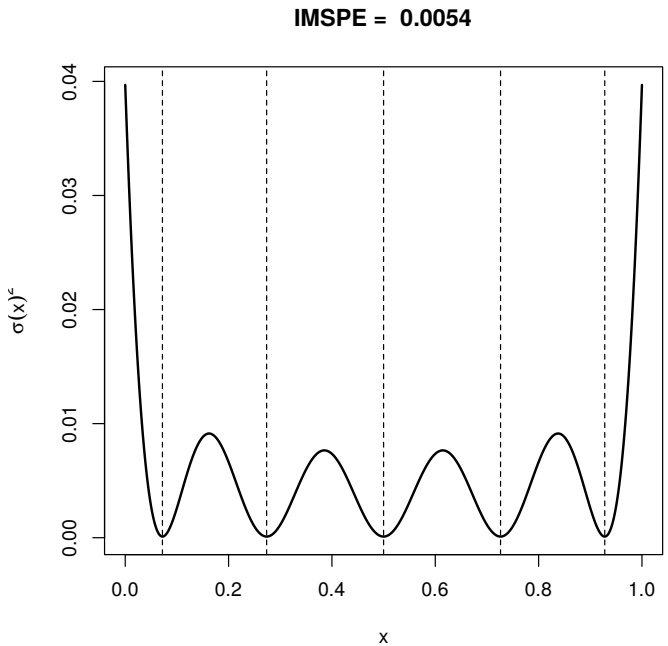


Sequential

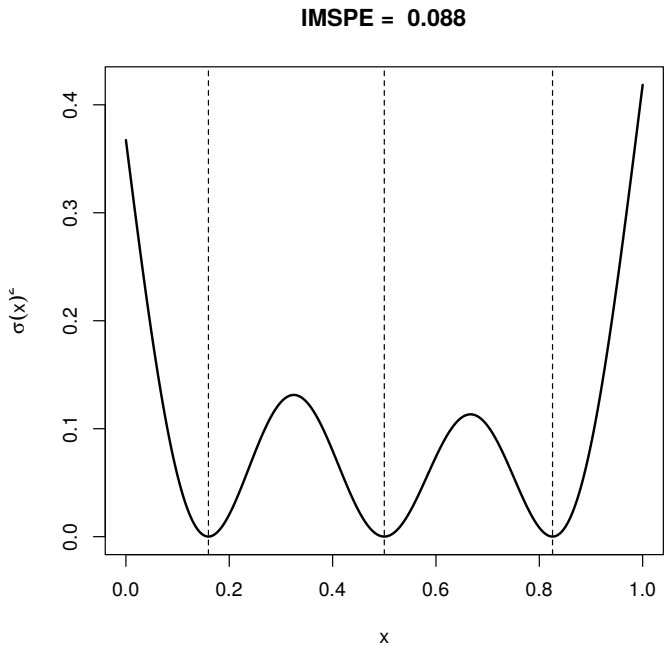
Myopic IMSPE

Like most criteria, IMSPE ignores the limited budget.

Example (fixed hyperparameters)



Non-sequential

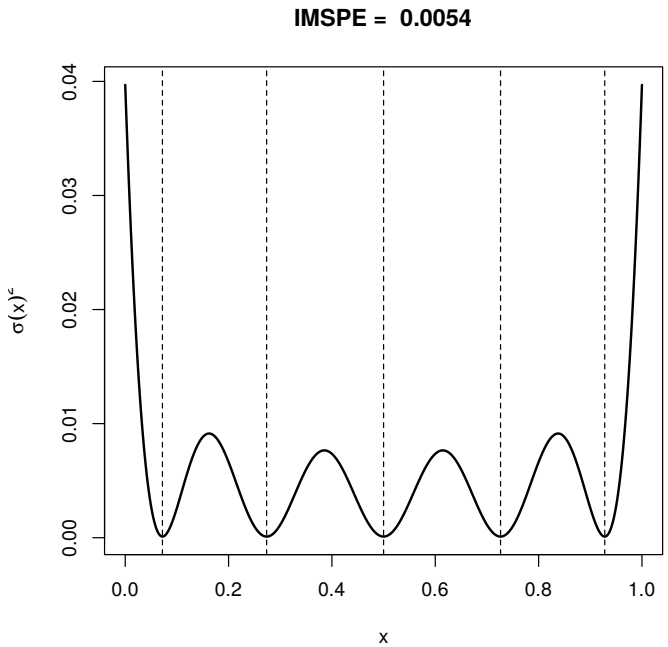


Sequential

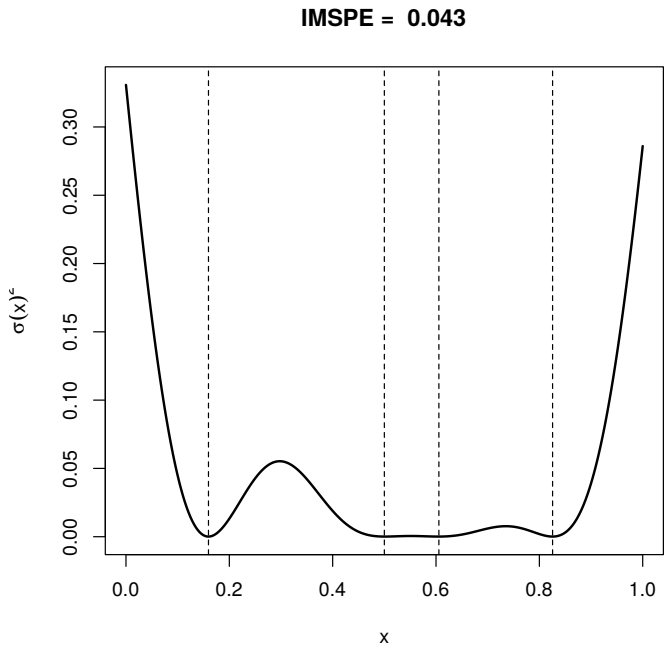
Myopic IMSPE

Like most criteria, IMSPE ignores the limited budget.

Example (fixed hyperparameters)



Non-sequential

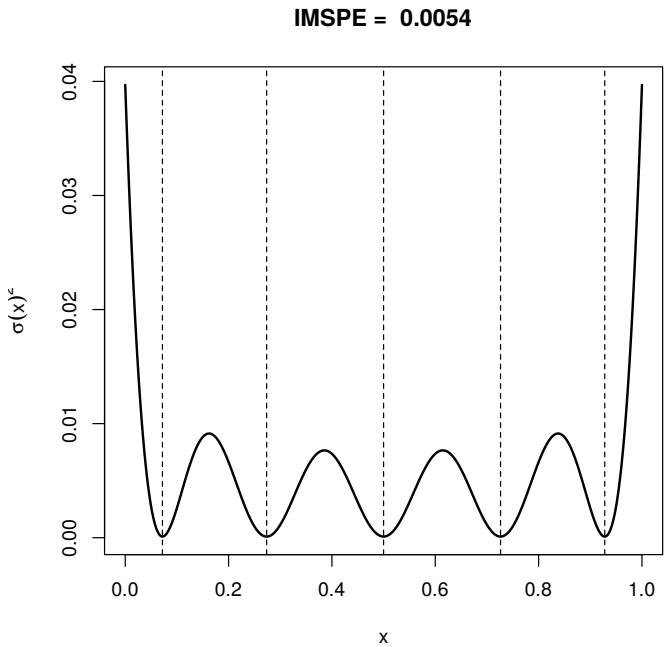


Sequential

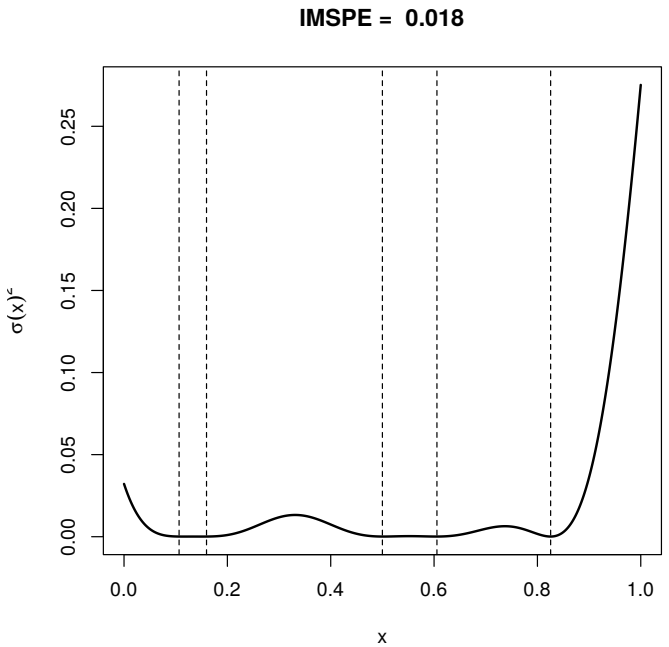
Myopic IMSPE

Like most criteria, IMSPE ignores the limited budget.

Example (fixed hyperparameters)



Non-sequential

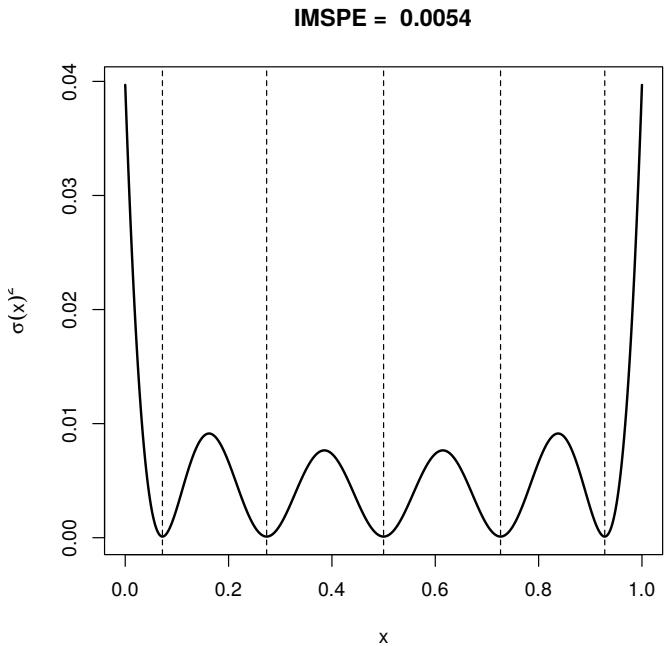


Sequential

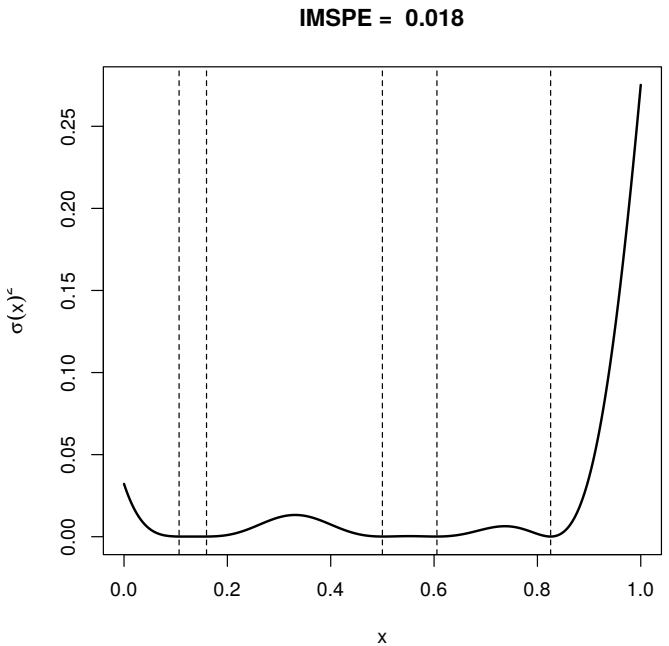
Myopic IMSPE

Like most criteria, IMSPE ignores the limited budget.

Example (fixed hyperparameters)



Non-sequential



Sequential

If monotone submodularity holds, with fixed hyperparameters, the gap between the two methods is bounded.

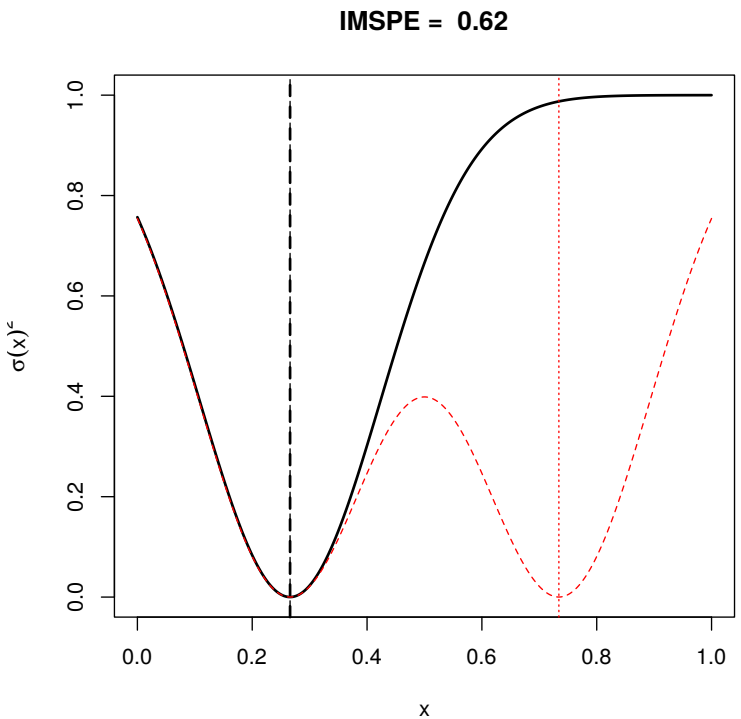
Looking ahead principle

Aim: taking into account the future steps of optimization:

$$\text{find } \mathbf{x}_{N+1} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+h})$$

where $\mathbf{x}^*_{N+1+i} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+i})$, $1 \leq i \leq h$.

Example: looking one step ahead ($h = 1$)



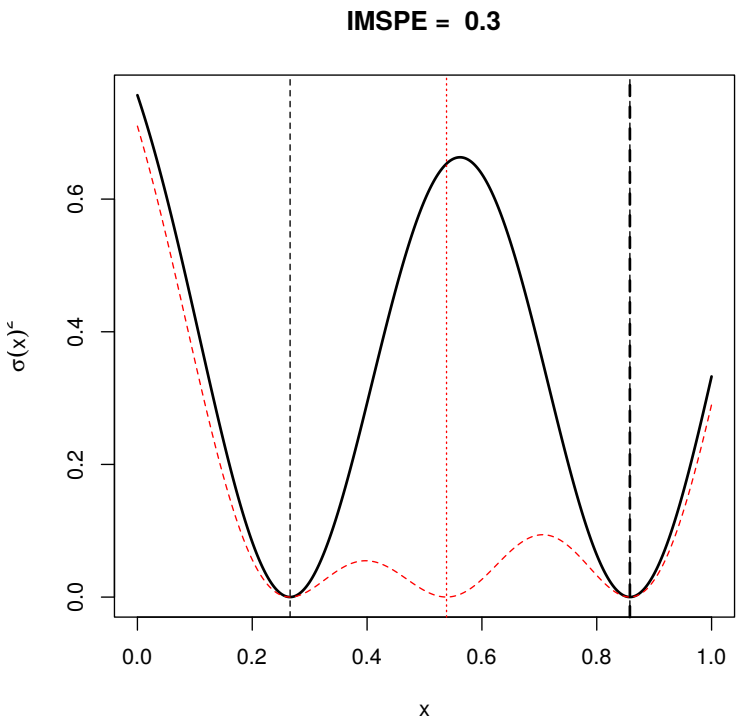
Looking ahead principle

Aim: taking into account the future steps of optimization:

$$\text{find } \mathbf{x}_{N+1} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+h})$$

where $\mathbf{x}^*_{N+1+i} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+i})$, $1 \leq i \leq h$.

Example: looking one step ahead ($h = 1$)



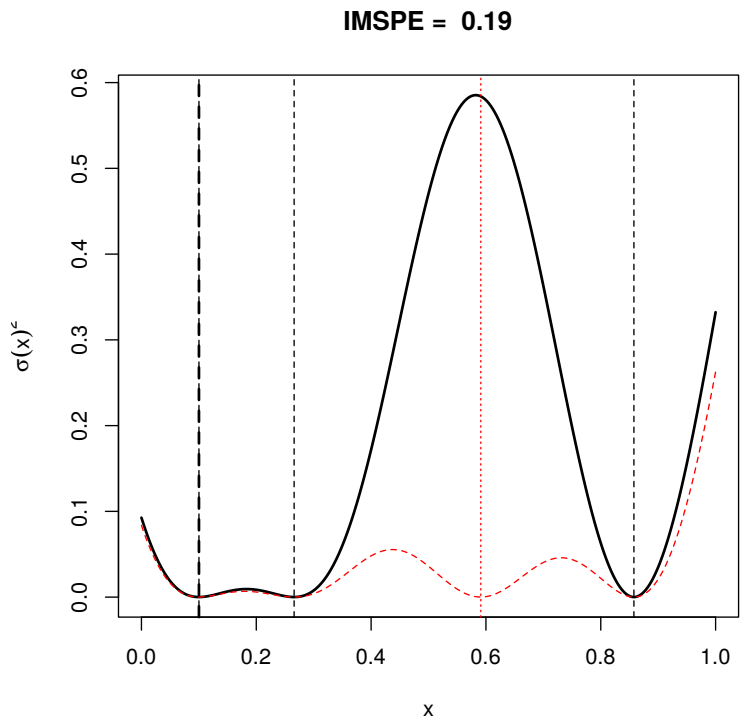
Looking ahead principle

Aim: taking into account the future steps of optimization:

$$\text{find } \mathbf{x}_{N+1} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+h})$$

where $\mathbf{x}^*_{N+1+i} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+i})$, $1 \leq i \leq h$.

Example: looking one step ahead ($h = 1$)



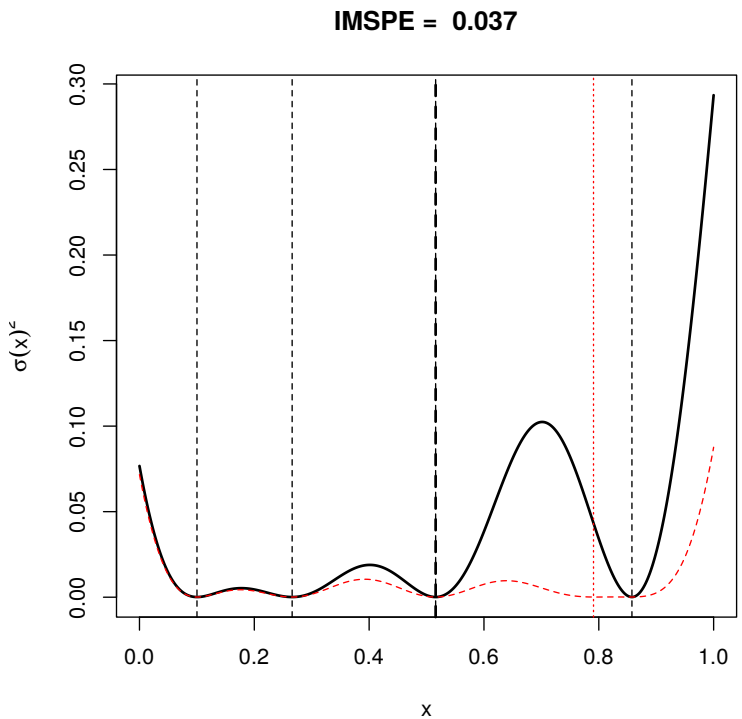
Looking ahead principle

Aim: taking into account the future steps of optimization:

$$\text{find } \mathbf{x}_{N+1} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+h})$$

where $\mathbf{x}^*_{N+1+i} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+i})$, $1 \leq i \leq h$.

Example: looking one step ahead ($h = 1$)



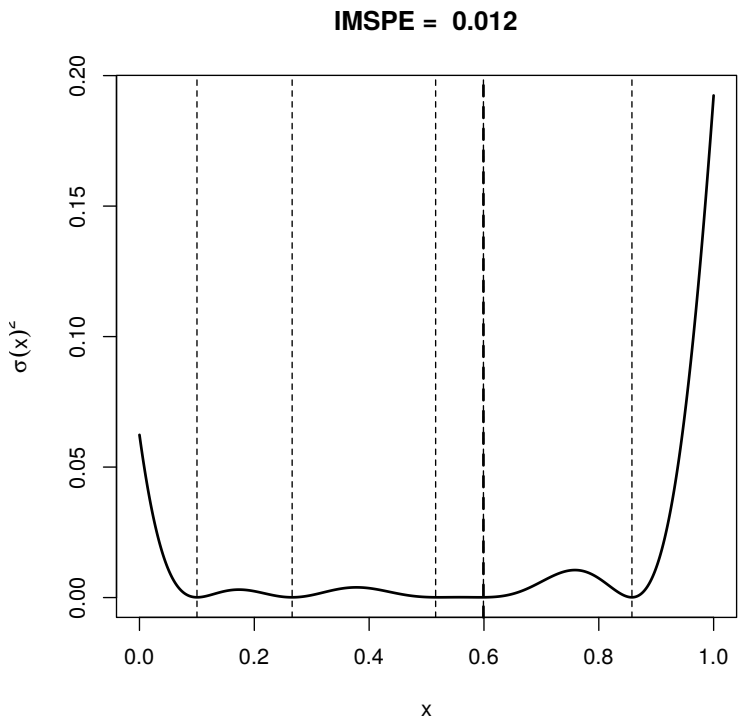
Looking ahead principle

Aim: taking into account the future steps of optimization:

$$\text{find } \mathbf{x}_{N+1} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+h})$$

where $\mathbf{x}^*_{N+1+i} \in \arg\min IMPSE(\mathbf{X}, \mathbf{x}_{N+1}, \mathbf{x}^*_{N+2}, \ldots, \mathbf{x}^*_{N+1+i})$, $1 \leq i \leq h$.

Example: looking one step ahead ($h = 1$)



But, as the horizon increases, it becomes quickly slow.

Looking ahead principle for replication

Recall that n should remain moderate. In this context:

- sampling a new design impacts both the remaining budget of new designs, and the computational time of future iterations
- while looking for the best design to replicate is fast

So how to encourage replication in the process?

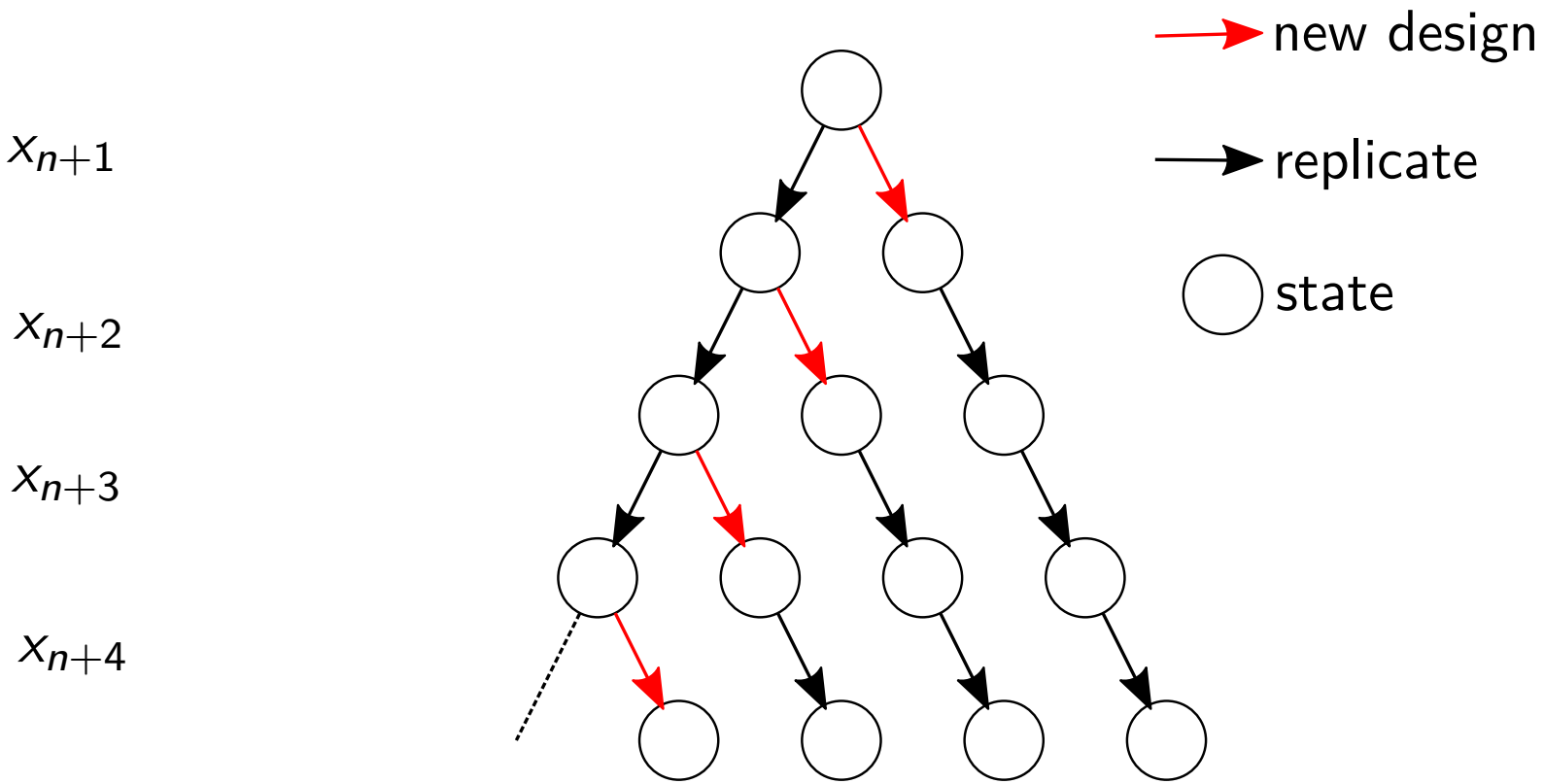
Clearly, searching for the best sequence of new/replicated designs is out of reach.

$\Rightarrow$ We thus simplify the problem to consider the following decision at each step:

- add a new design now (and replicate later)
- replicate now (add a new design later)

Looking ahead principle to encourage replication (2)

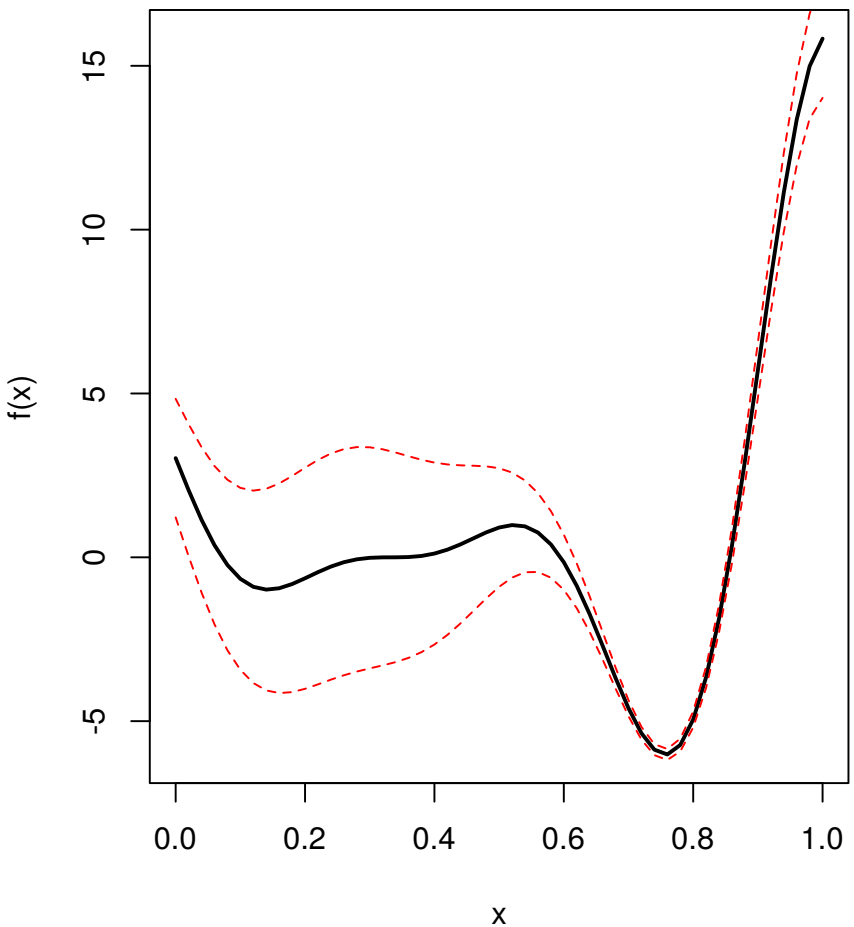
Graphical view of the rollout procedure, horizon $h = 3$:



Note: increasing the number of steps-ahead has a flattening effect, since **when** the new design is added has less and less effect.

Illustration: 1-dimensional function

True functions



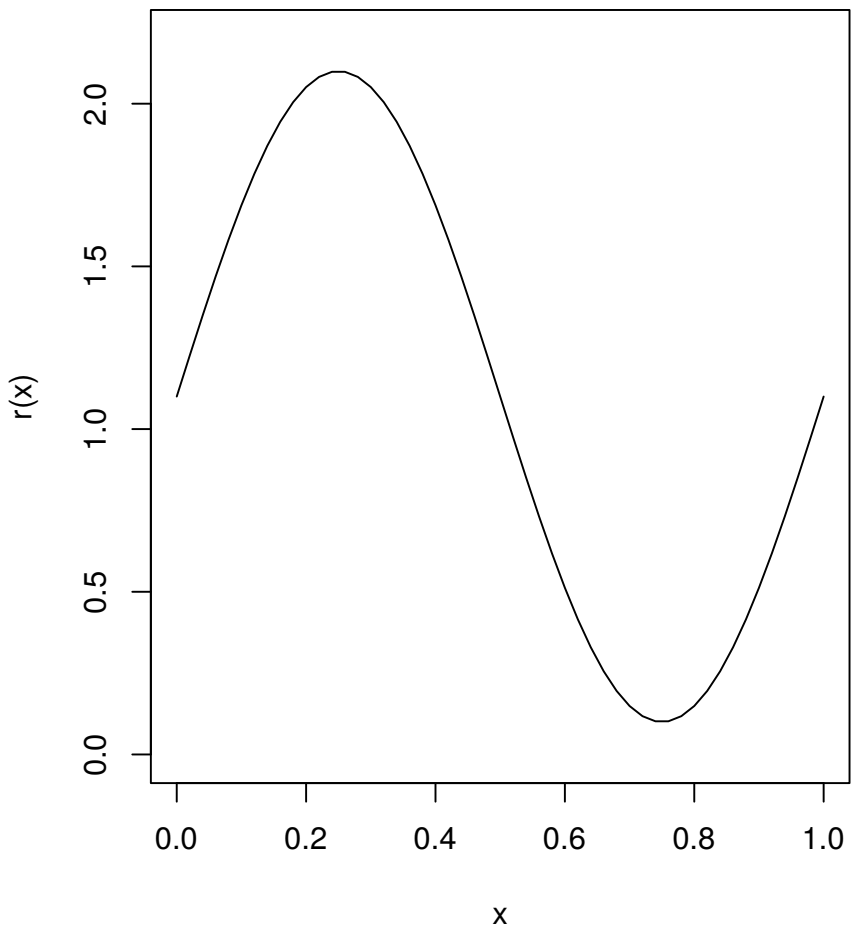


Illustration: 1-dimensional function

Initial design: 21 equi-spaced points with 5 replicates

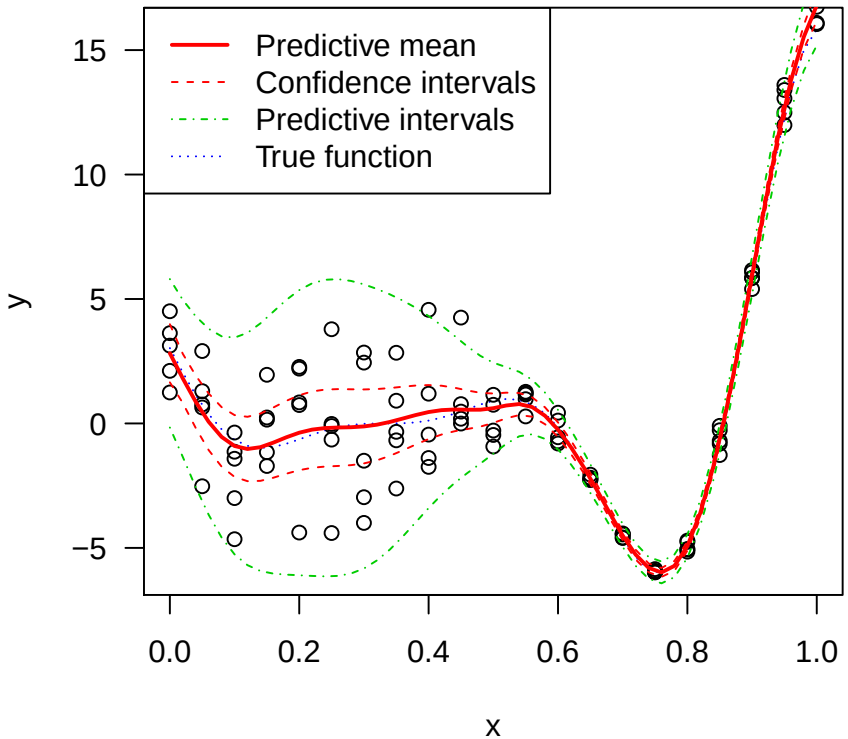
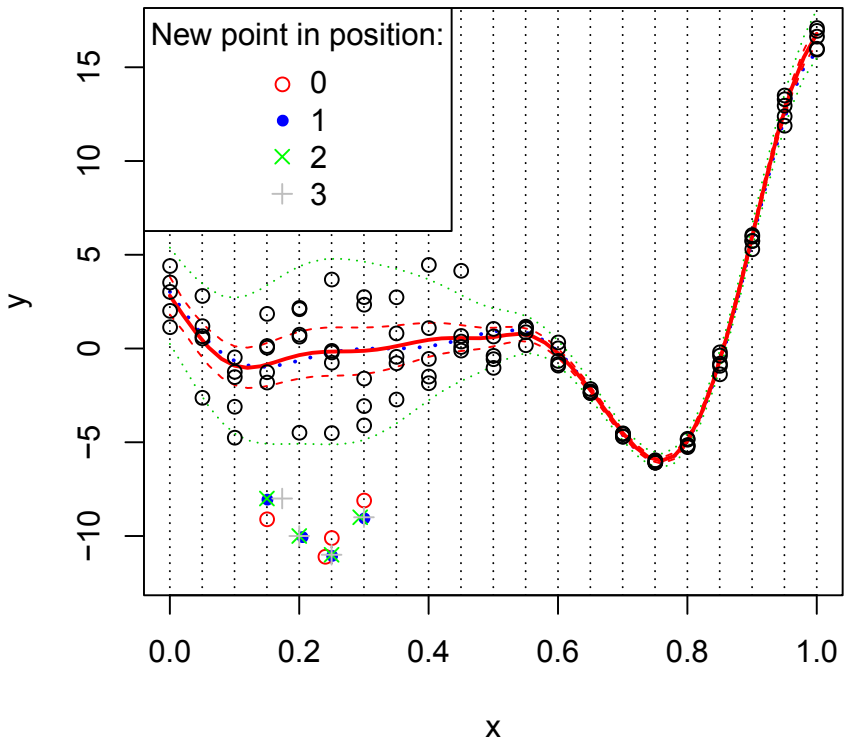
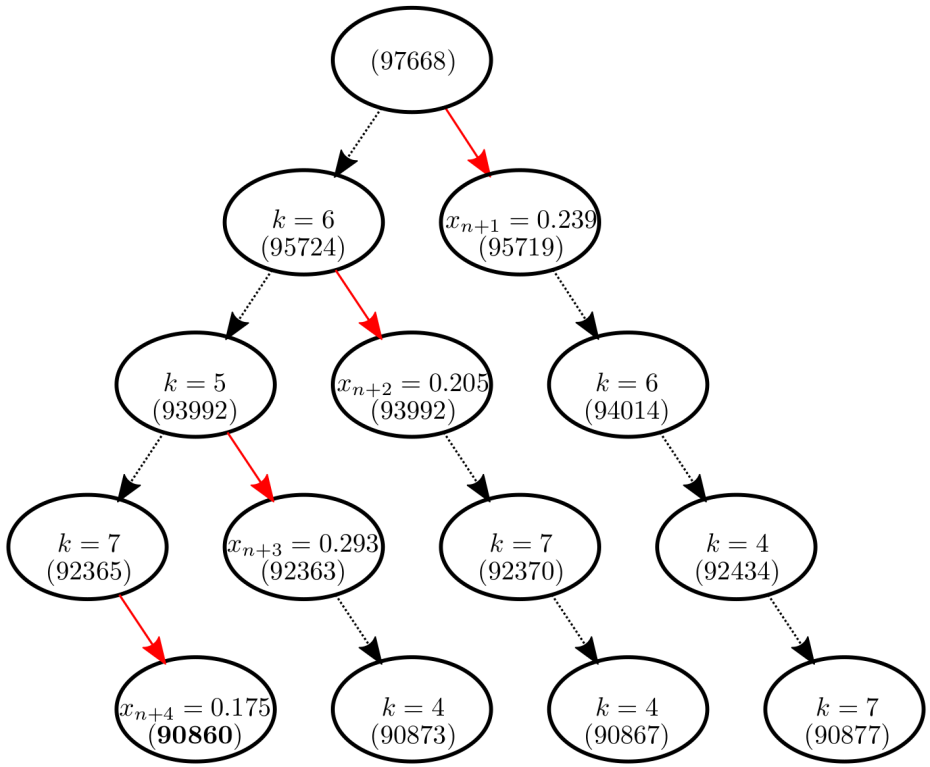


Illustration: 1-dimensional function

Now consider the next decision, with $h = 3$:





Selecting the horizon

How can we choose the horizon, h, in real-time?

We have simple on-line adjustments which tune the horizon in order to:

- **Target** a ratio n/N, reducing the GP modeling cost

$$h_{N+1} \leftarrow \begin{cases} h_N + 1 & \text{if } n/N > \rho \text{ \& a new } \bar{x}_{n+1} \text{ is chosen} \\ \max\{h_N - 1, -1\} & \text{if } n/N < \rho \text{ \& a replicate is chosen} \\ h_N & \text{otherwise.} \end{cases}$$

- **Adapt** to minimize IMSPE regardless of computational cost

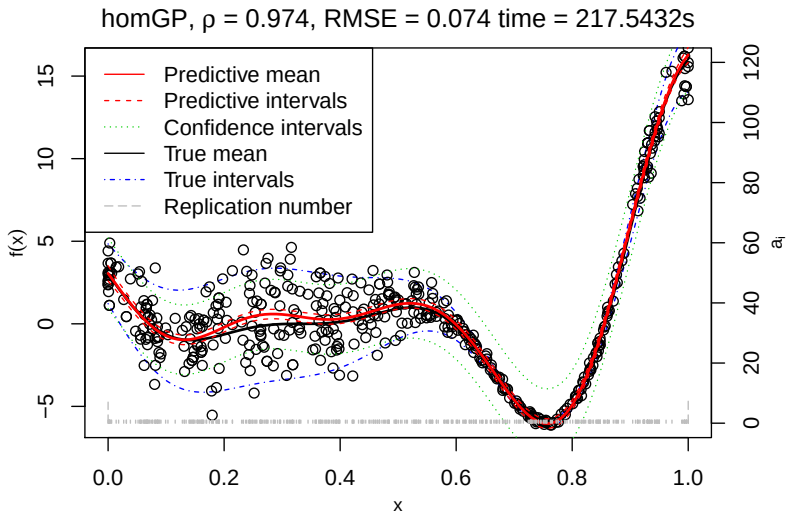
$$h_{N+1} \sim \mathrm{Unif}\{a'_1, \ldots, a'_n\} \quad \text{with} \quad a'_i := \max(0, a^*_i - a_i)$$

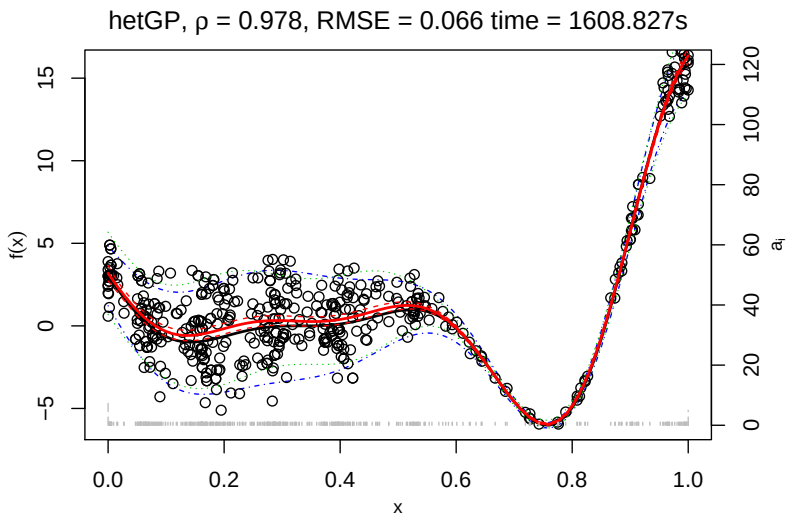
with $a^*_i \propto r(\bar{x}_i)K_i = (\mathbf{K}_n^{-1}\mathbf{W}_n\mathbf{K}_n^{-1})_{i,i}$ the static optimal replicate balancing from the SK literature.

Example 1: 1-dimensional function

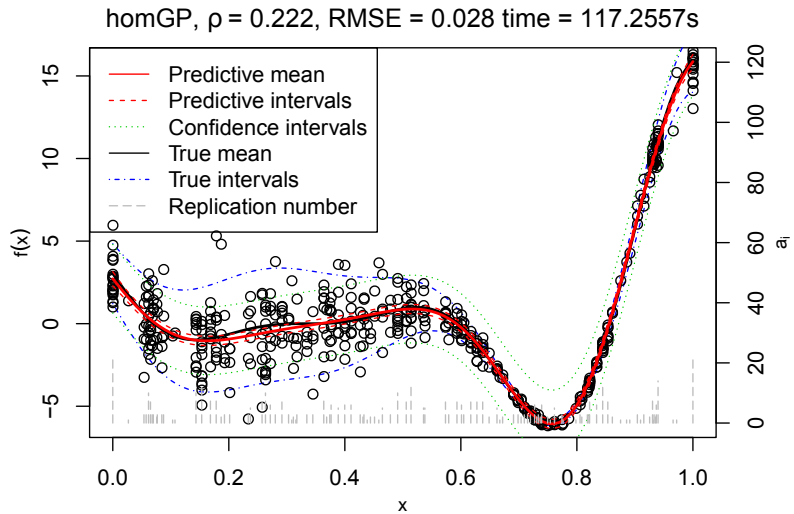
Setup: $1d$ test case, 20 initial points, 480 infill points.

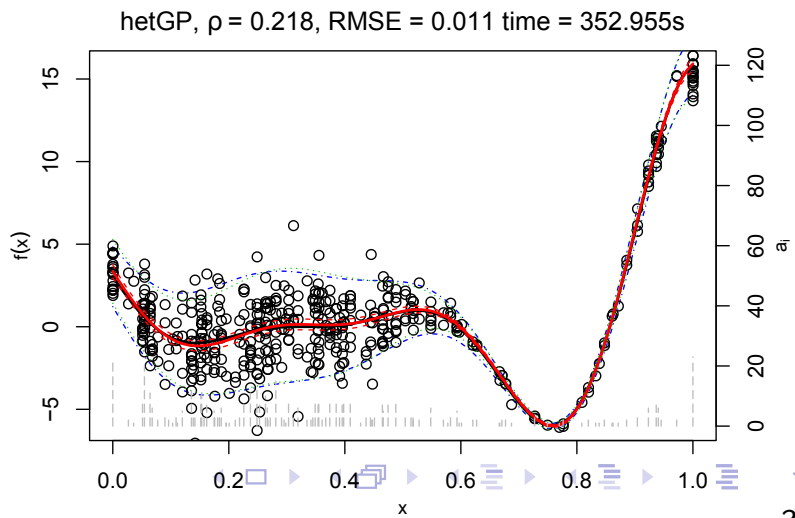
No look-ahead





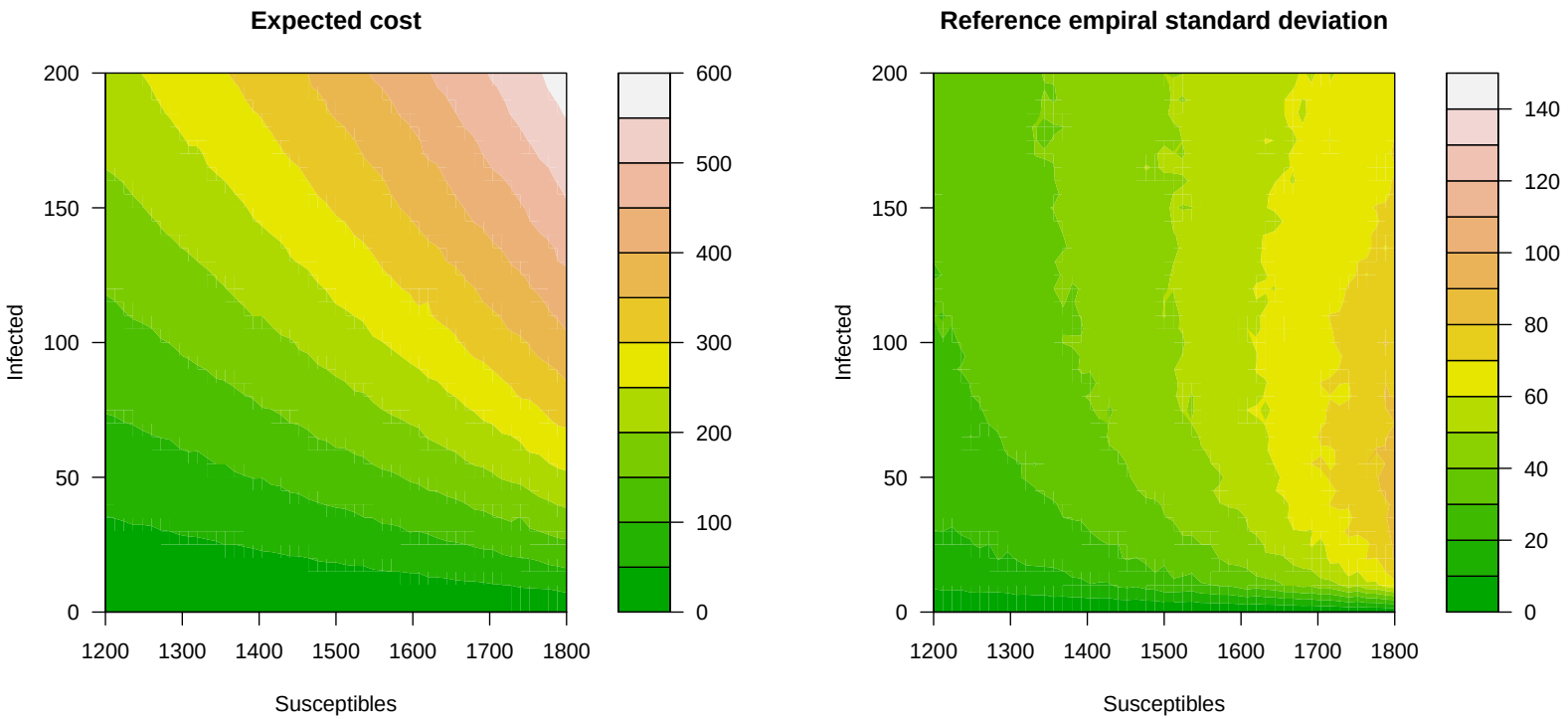
Look-ahead with adapt scheme





Example 2: Epidemic management (Hu et al., 2015)

Reference mean and noise surfaces

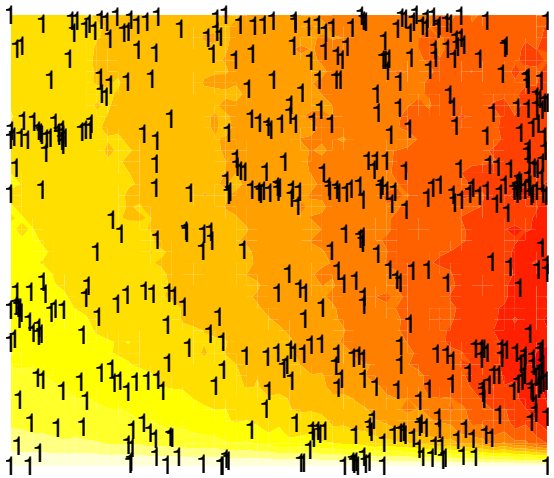


Results focusing on the mean accuracy relative to predicted variance based on a proper scoring rule (Gneiting et al., 2007):

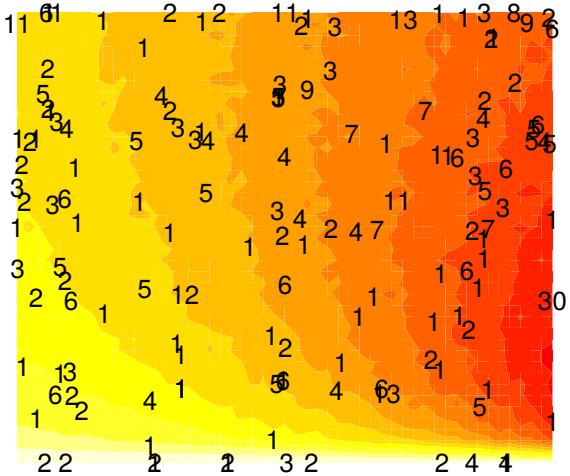
$$S(P, y) = -\left(\frac{y - \mu_P}{\sigma_P}\right)^2 - log(\sigma_P^2)$$

Example 2: results on the SIR problem

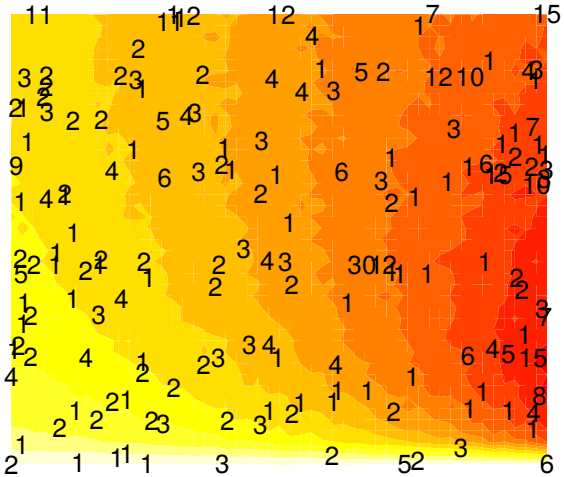
Variance surfaces and replication:



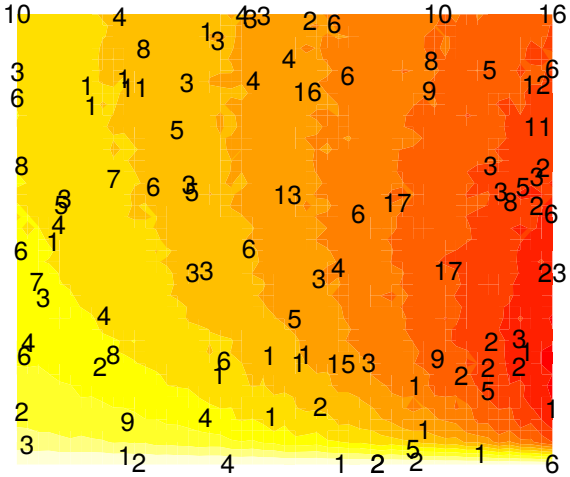
h = -1



Adapt



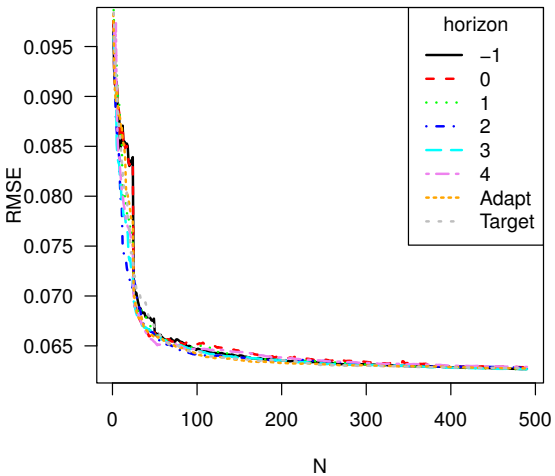
h = 1

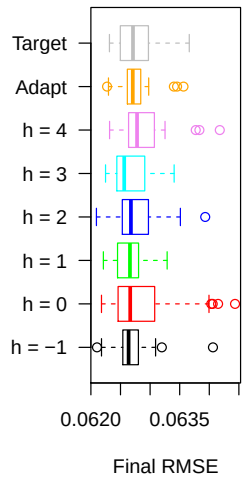


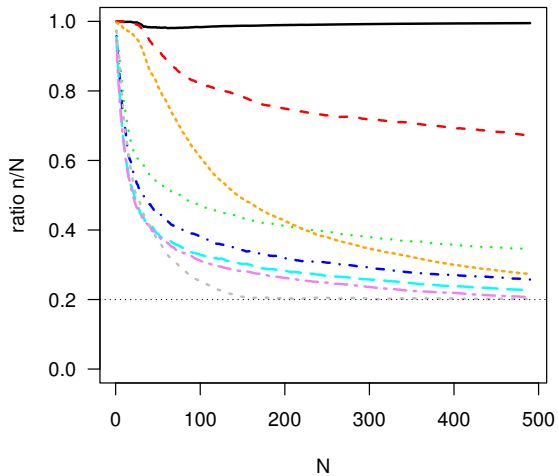
Target

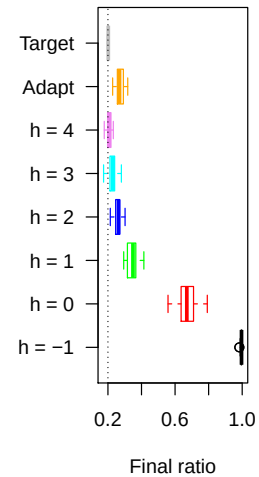
Example 2: results on the SIR problem

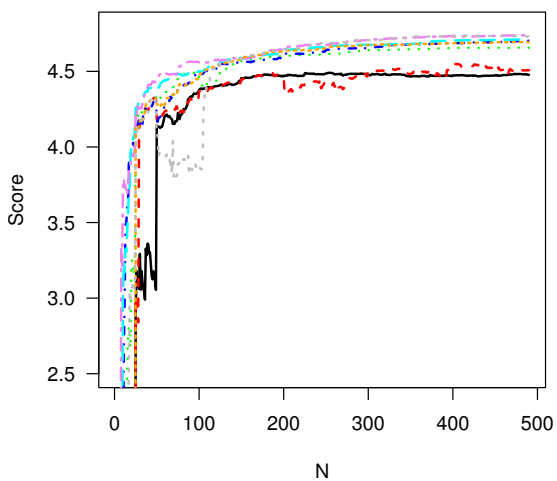
Sequential performance averaged over 30 repetitions

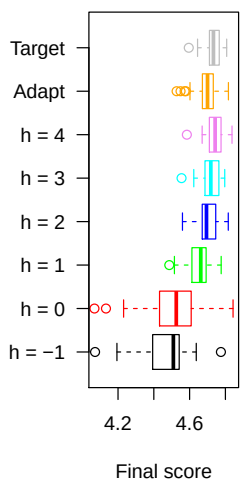


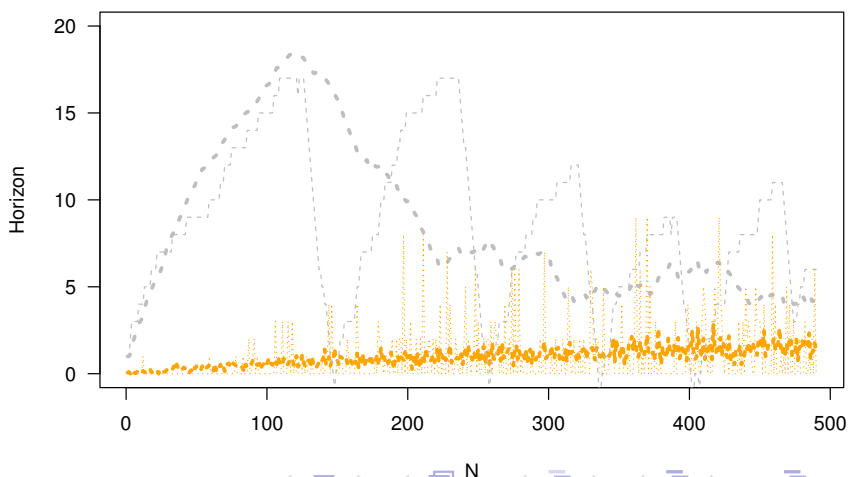












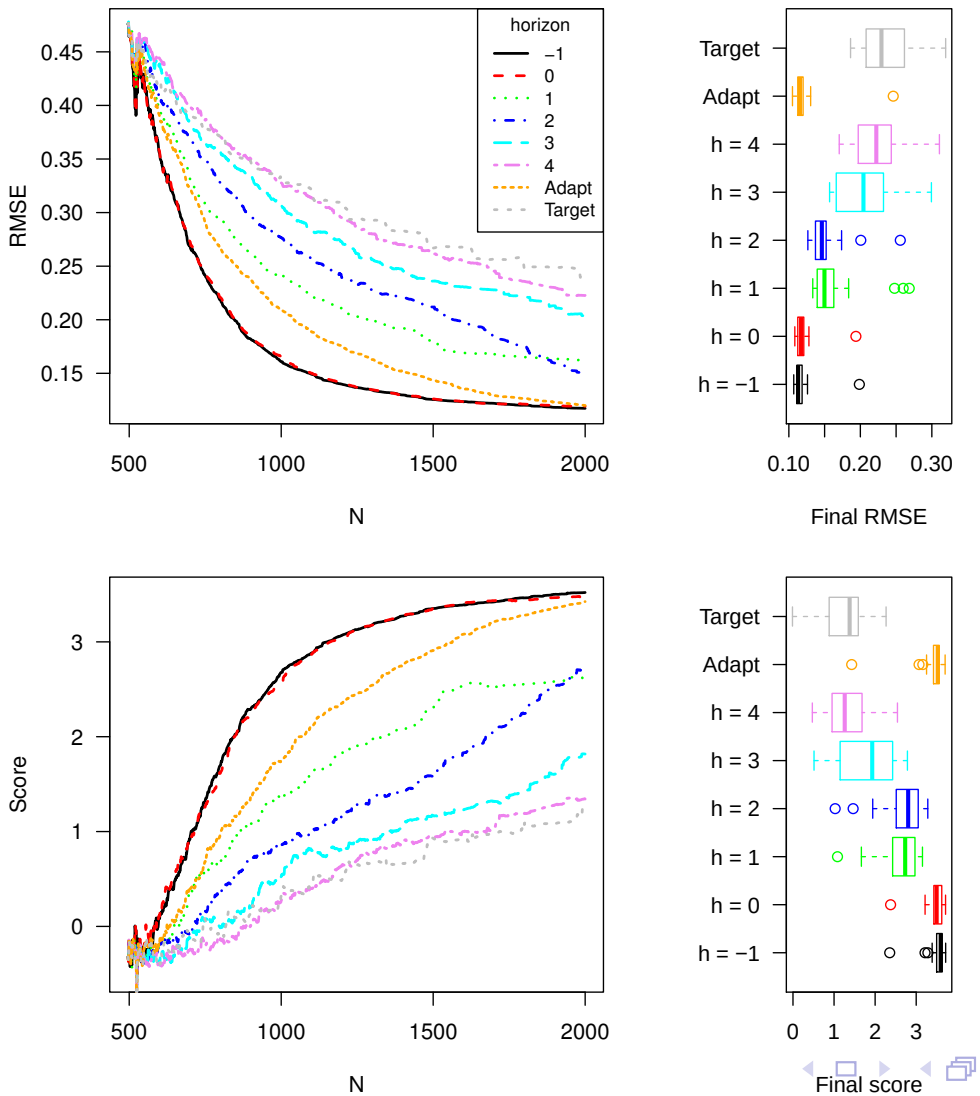
Example 3: Assemble to order (Hong et al., 2006; Xie et al., 2012)

Inventory management problem:

- 5 products are produced, requiring some of 8 different items
- selling products bring profit, storing items have a cost
- orders come randomly over a time period
- random replenishment of items
- variables are the target stock of each item ($[0, 20]^8$)
- output is the profit per unit time

Example 3: results on the ATO problem

Sequential performance averaged over 30 repetitions



Can we use the same framework for other goals?

Other criteria:

- optimization, e.g., with the Expected Improvement [Mockus et al., 1978]:

$$I : \mathbf{x} \in \mathbb{R}^d \to \max \left(\min_{1 \leq i \leq n} Y(\mathbf{x}_i) - Y(\mathbf{x}) \right)$$

- contour finding, e.g., with the Maximum Contour Uncertainty for level 0 [Lyu et al., 2018+]:

$$MCU : \mathbf{x} \in \mathbb{R}^d \to \Phi \left(-\frac{|\mu(\mathbf{x})|}{\sigma(\mathbf{x})} \right)$$

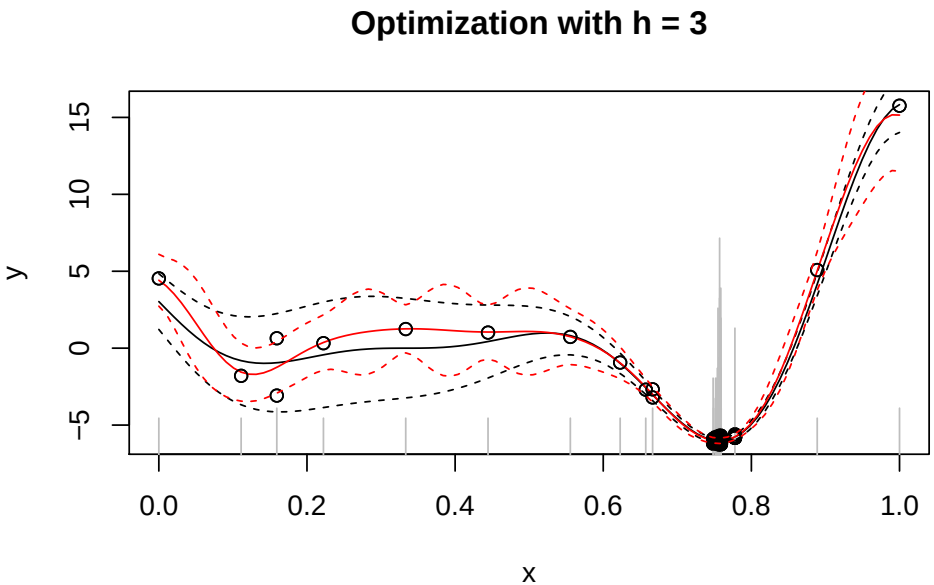
- multi-objective, constraints, equilibria, ...

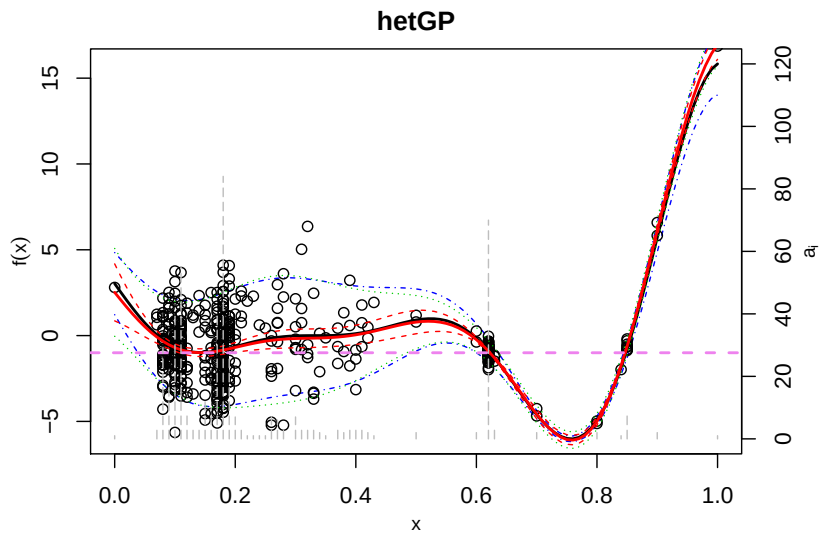
Open question: can replication be optimal in those cases too?

Looking ahead is still encouraging replication.

Illustration of optimization/contour finding

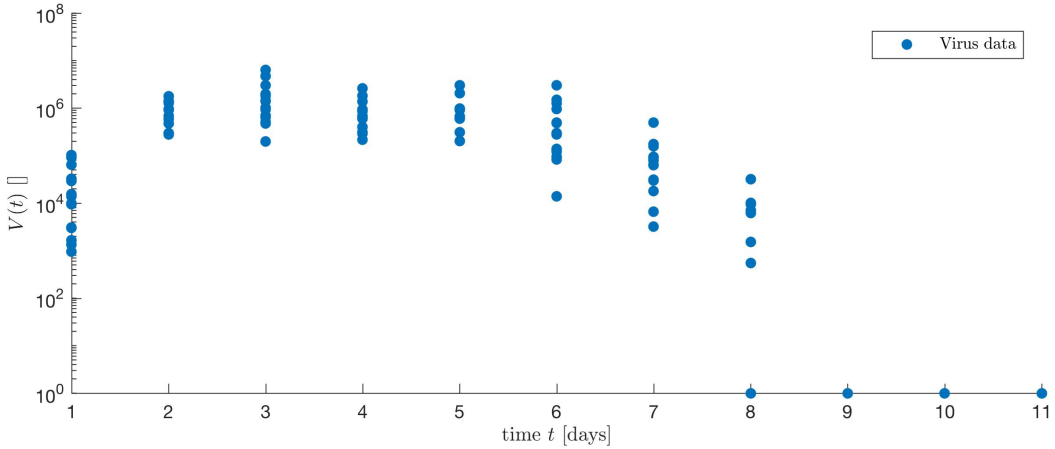
Setup: 1d test case, 10 initial points, 490 infill points. Left: minimization
Right: -1 level set estimation





Handling noise with larger tails than Gaussian

GPs are not robust to outliers. Consider this influenza data on mice:



[Shah2014] generalized GPs to Student-t processes, with homoskedastic noise.
Denote α the degree of freedom parameter. The predictive equations are:

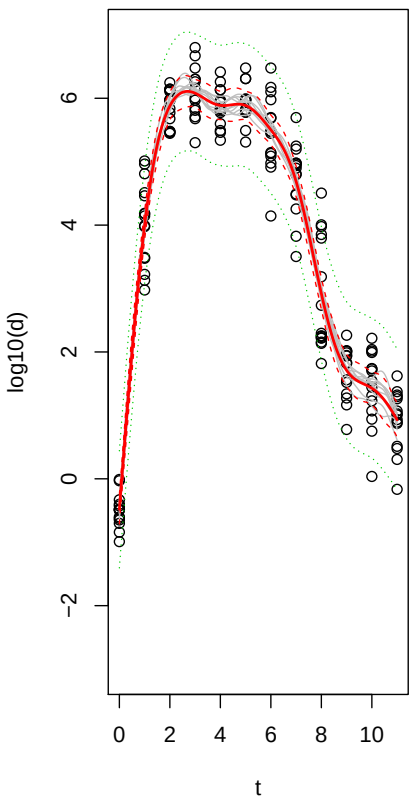
$$\mu_{TP}(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top (\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{Y},$$

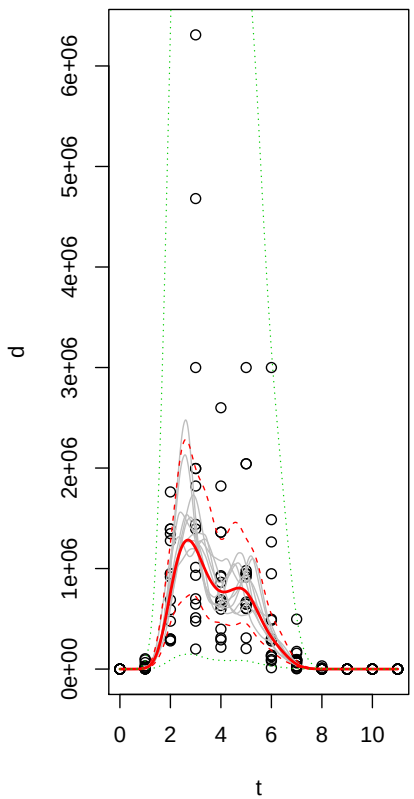
$$\sigma^2(\mathbf{x})_{TP} = \frac{\alpha + \mathbf{Y}^\top (\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{Y} - 2}{\alpha + N - 2}\left(k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x})^\top (\mathbf{K}_N + \mathbf{\Sigma}_N)^{-1}\mathbf{k}(\mathbf{x})\right) + r(\mathbf{x})$$

Turns out that it can be extended further as we showed for GPs.

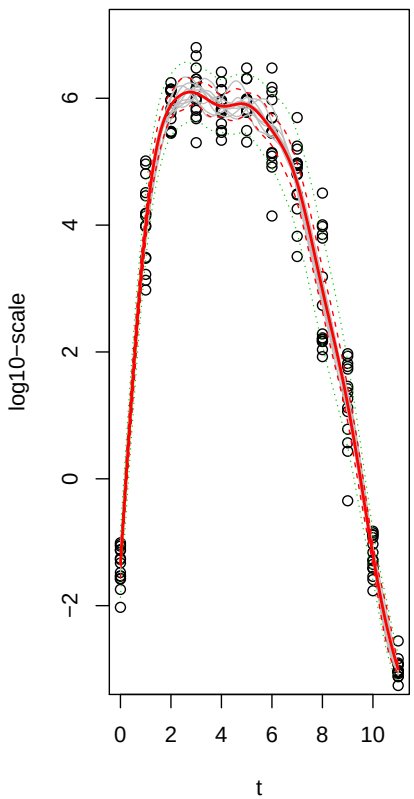
Comparison of heteroskedastic GPs and TPs

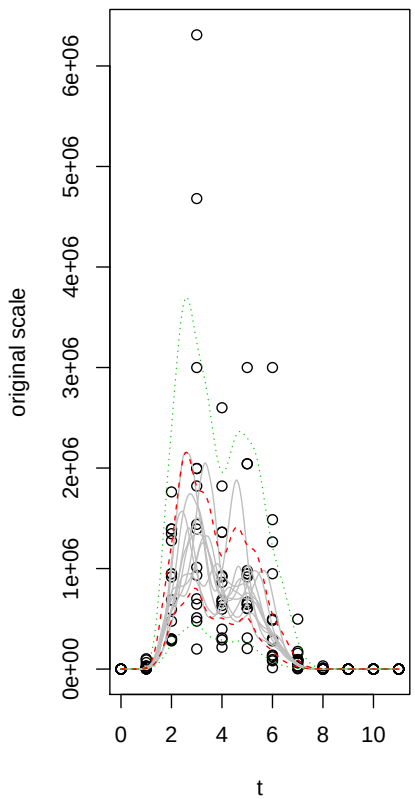
GP





TP

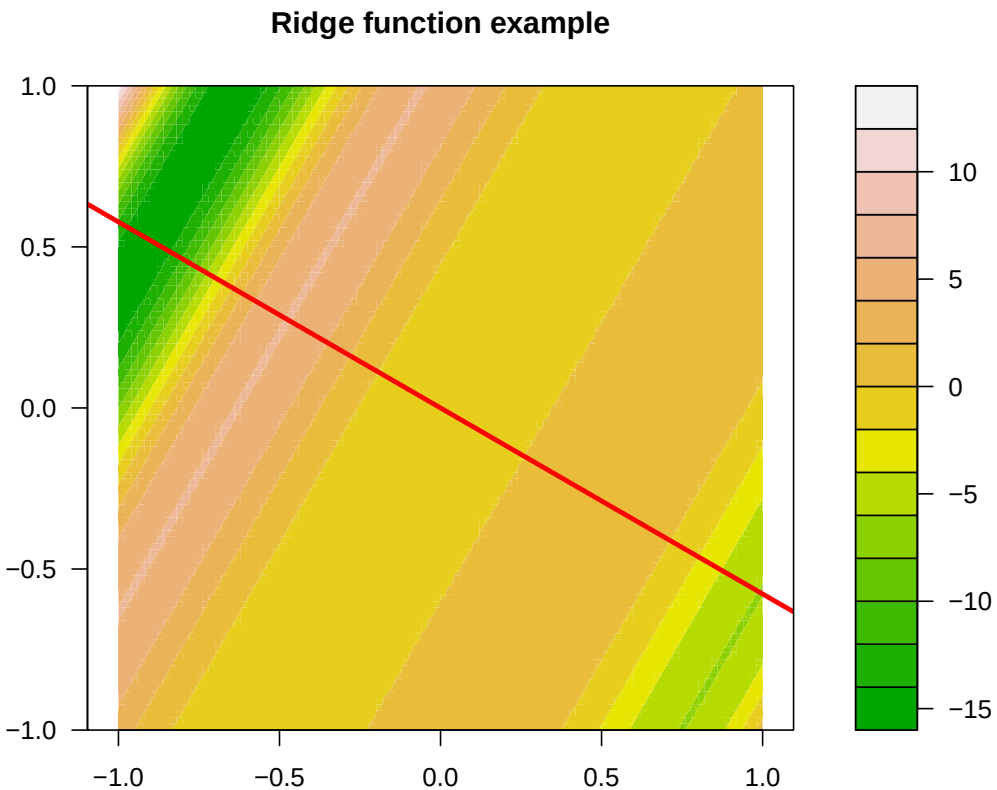




Scaling up BO to many variables: active subspaces

Observation: in many cases, the variation is concentrated around a few directions

Model: $f(\mathbf{x}) = g(\mathbf{A}^\top \mathbf{x})$ with $\mathbf{A} \in \mathbb{R}^{D \times d}$ (ridge function)



Scaling up BO to many variables: active subspaces

Observation: in many cases, the variation is concentrated around a few directions

$$\text{Model: } f(\mathbf{x}) = g(\mathbf{A}^\top \mathbf{x}) \text{ with } \mathbf{A} \in \mathbb{R}^{D \times d} \text{ (ridge function)}$$

Backed by empirical and theoretical evidence, e.g., Constantine et al. (2016)

Options exist to estimate $\mathbf{A}$, most rely either:

- on the gradient of f, to estimate $\mathbf{C} = \int \nabla(f(\mathbf{x}))^\top \nabla(f(\mathbf{x}))\mu(dx)$, see e.g., Djolonga et al. (2013), Constantine (2015). They are usually in two phases.
- on treating $\mathbf{A}$ as an hyperparameter, see e.g., Garnett et al. (2014); Tripathy et al. (2016); Marcy (2018)

In both cases, it is unclear how to split the budget between learning $\mathbf{A}$ and optimizing.

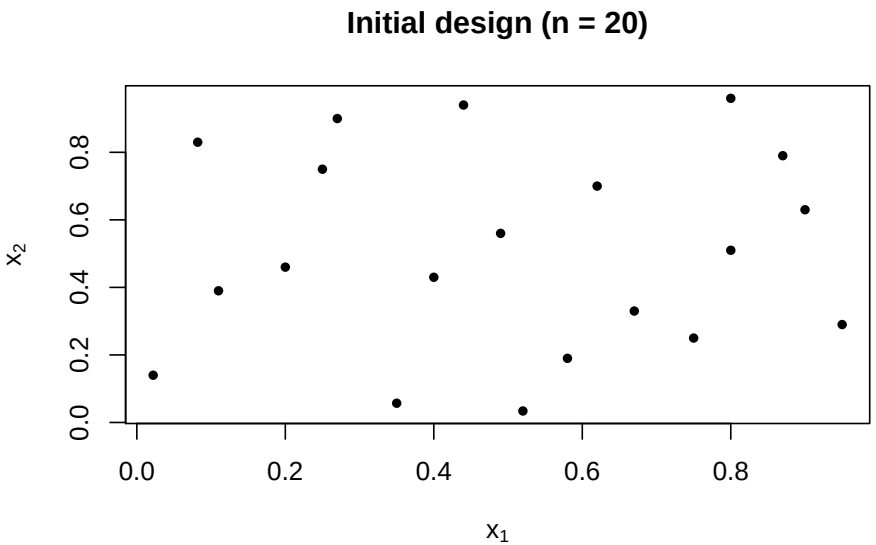
C) Active subspace estimation (2)

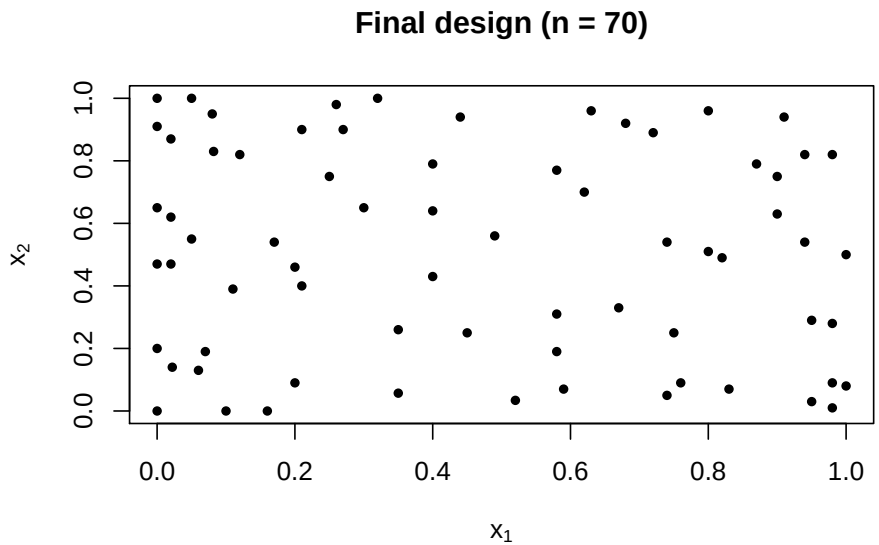
Commonly, $\mathbf{C}$ is estimated by Monte Carlo: $\hat{\mathbf{C}} = \sum_{i=1}^{P} \nabla(f(X_i))^\top \nabla(f(X_i))$ with iid $X_1, \ldots, X_P$ in Ω, see, e.g., Constantine (2015).

Main limitations: 1) f may not have a gradient and, 2) iid assumption

For a GP, we show that a closed-form expression of $\mathbf{C}$ is directly available, which only depends on the D lengthscale hyperparameters.

These expressions also enable learning the active subspace sequentially.





Outline

Conclusion

When the mean and variance are changing non-linearly in the input space:

- coupled GPs gives accurate fits
- it can be advantageous to have replication in the design.

The more heteroskedastic the more replication:

- intuitively, that must be true: both signal and noise are changing and replication is the only reliable tool for separating the two.

Replication has the added benefit of yielding faster fitting of GPs.

Extension to Student-t processes is possible to handle larger tails.

Corresponding codes are available in the `R` CRAN package `hetGP`.

Bibliography I

Alvarez, M., Luengo, D. and Lawrence, N. (2009) Latent force models. AISTATS, 9–16.

Ankenman, B. E., Nelson, B. L., and Staum, J. (2010). Stochastic kriging for simulation metamodeling. *Operations research*, 58:371–382.

B., M., Gramacy, R., and Ludkovski, M. (2017). Practical heteroskedastic Gaussian process modeling for large simulation experiments. Journal of Computational and Graphical Statistics. ArXiv preprint arXiv:1611.05902.

B., M., Huang, G., Gramacy, R., and Ludkovski, M. (2017). Replication or exploration? Sequential design for stochastic simulation experiments. Technometrics. ArXiv preprint arXiv:1611.05902.

B., M., and Gramacy, R. `hetGP`: Heteroskedastic Gaussian Process Modeling and Sequential Design in `R` Package vignette

Boukouvalas, A. and Cornford, D. (2009). Learning heteroscedastic Gaussian processes for complex datasets. Technical report, Aston University, Neural Computing Research Group.

Chen, X. and Zhou, Q. (2017). Sequential design strategies for mean response surface metamodeling via stochastic kriging with adaptive exploration and exploitation *European Journal of Operational Research*, 262(2):575–585.

Chung, M., Binois, M., Gramacy, R., Moquin, D., Smith, A. and Smith, A. (2018) Parameter and uncertainty estimation for dynamical systems using surrogate stochastic processes Preprint

Bibliography II

Chevalier, C., Ginsbourger, D., and Emery, X. (2014). Corrected kriging update formulae for batch-sequential data assimilation. In *Mathematics of Planet Earth*, 119–122. Springer.

Constantine, P. G. (2015). *Active subspaces: Emerging ideas for dimension reduction in parameter studies*. SIAM.

Constantine, P. G., del Rosario, Z., and Iaccarino, G. (2016). Many physical laws are ridge functions. *arXiv preprint arXiv:1605.07974*.

Das, A. and Kempe, D. (2008). Algorithms for subset selection in linear regression. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 45–54. ACM.

Djolonga, J., Krause, A., and Cevher, V. (2013). High-dimensional Gaussian process bandits. In *Neural Information Processing Systems*, pages 1025–1033.

Durrande, N. (2013) Kernel design. Gaussian process summer school, http://gpss.cc/gpss13/assets/Sheffield-GPSS2013-Durrande.pdf

Garnett, R., Osborne, M. A., and Hennig, P. (2014). Active learning of linear embeddings for Gaussian processes. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pages 230–239. AUAI Press.

Ginsbourger, D. and Roustant, O. and Durrande, N. (2013). Invariances of random fields paths, with applications in Gaussian Process Regression. arXiv preprint arXiv:1308.1359.

Goldberg, P. W., Williams, C. K., and Bishop, C. M. (1998). Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems*, volume 10, pages 493–499, Cambridge, MA. MIT press.

Bibliography III

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102, 477, 359–378.

Hong, L. and Nelson, B. (2006). Discrete optimization via simulation using COMPASS. *Operations Research*, 54(1):115–129.

Hu, R. and Ludkovski, M. (2015). Sequential design for ranking response surfaces. *arXiv preprint arXiv:1509.00980*.

Kersting, K., Plagemann, C., Pfaff, P., and Burgard, W. (2007). Most likely heteroscedastic Gaussian process regression. In *Proceedings of the International Conference on Machine Learning*, pages 393–400, New York, NY. ACM.

Krause, A., Singh, A., and Guestrin, C. (2008). Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9, Feb, 235–284.

Lazaro-Gredilla, M. and Titsias, M. (2011). Variational heteroscedastic Gaussian process regression. In *Proceedings of the International Conference on Machine Learning*, pages 841–848, New York, NY. ACM.

Liu, M. and Staum, J. (2010). Stochastic kriging for efficient nested simulation of expected shortfall. *The Journal of Risk*, 12, 3, 3.

Lyu, X., Binois, M. and Ludkovski, M. (2018) Evaluating Gaussian Process Metamodels and Sequential Designs for Noisy Level Set Estimation arxiv 1807.06712

Bibliography IV

Marcy, P. (2018). Bayesian Gaussian process models for dimension reduction uncertainties. ASA Joint research conference.

Tripathy, R., Bilionis, I., and Gonzalez, M. (2016). Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. *Journal of Computational Physics*, 321:191 – 223.

Wang, W. and Chen, X. (2016). The effects of estimation of heteroscedasticity on stochastic kriging. *Proceedings of the 2016 Winter Simulation Conference*, 326-337.

Wycoff, N., Binois, M., and Wild, S. Sequential Learning of Active Subspaces *In preparation*

Xie, J., Frazier, P., and Chick, S. (2012). Assemble to order simulator.

Thanks

Questions?