Gaussian processes: surrogate models for continuous black-box optimization

Lukáš Bajer

MFF UK 04/2018

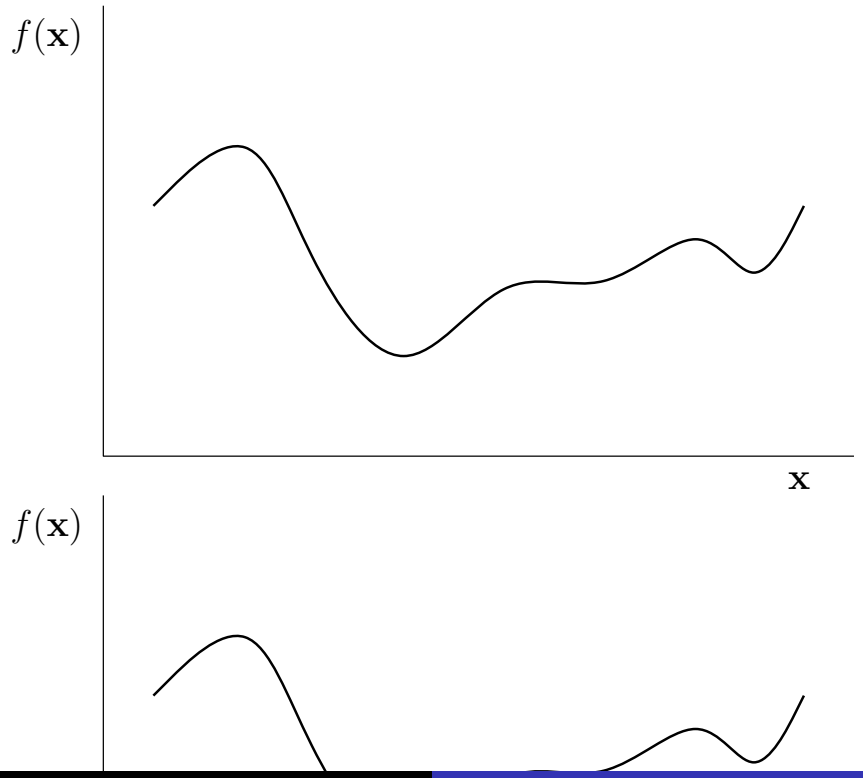
Contents

Optimization

- **optimization** (minimization) is finding such $\mathbf{x}^\star \in \mathbb{R}^n$ that

$$f(\mathbf{x}^\star) = \min_{\forall \mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

- "near-optimal" solution is usually sufficient



Continuous white-box optimization

also known as *numerical optimization methods*

requirements:

- *gradients*: $\nabla f(\mathbf{x})$
  - . . . can be approximated by *finite difference approximations*
- and sometimes also *Hessians*: $\nabla^2 f(\mathbf{x})$

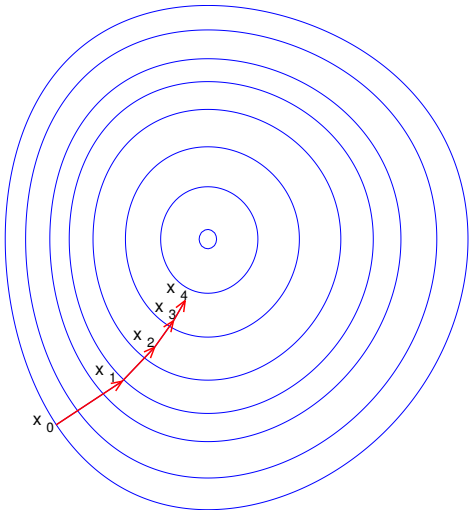
1. gradient descend (1^{st} order)
2. Newthon method (2^{nd} order)
3. quasi-Newthon methods (2^{nd} order approximated)
4. trust-region, conjugate gradients

1st order: gradient descend

- iterative steps in the direction of negative gradient

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \sigma \nabla f(\mathbf{x}^{(k)})$$

- σ – step size, usually changes every iteration, adapted, for example, using a *line search* along the gradient direction



source: (CC) Wikipedia

2nd order: Newton's method

- take into account the second-order term of a Taylor expansion of $f(\mathbf{x})$ around $\mathbf{x}^{(k)}$:

$$f(\mathbf{x}^{(k)} + \mathbf{h}) \approx q^{(k)}(\mathbf{h}) = f(\mathbf{x}^{(k)}) + \mathbf{h}^\top \nabla f^{(k)} + \frac{1}{2}\mathbf{h}^\top \left[\nabla^2 f^{(k)}\right] \mathbf{h}$$

- the next iterate is then

$$\mathbf{x}^{(k+1)} = (\mathbf{x}^{(k)} + \mathbf{h}^{(k)})$$

where $\mathbf{h}^{(k)}$ minimizes $q^{(k)}(\mathbf{h})$

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \gamma \left[\nabla^2 f^{(k)}\right]^{-1} \nabla f^{(k)}$$

Quasi-Newton's methods

- Hessian matrix $\nabla^2 f^{(k)}$ is not computed, only iteratively approximated $B_{(k)}, B_{(k+1)}, \ldots$
- Hessians' inverses are often calculated without inversions

BFGS

- the **most successful** for the last three decades
- independently discovered by 4 (!) people in 1970
  C. G. Broyden, R. Fletcher, D. Goldfarb and D. Shanno
- Hessian approximation updated via **rank-two updates**
- works **even without derivatives** (with finite differences)
- shown to behave well on a variety of (even multimodal) functions
- L-BFGS – a popular memory-limited version (Nocedal, 1980)
- in every optimization package (Matlab, Python,...)

Other numerical optimization techniques

- quadratic approximations:
  by far the most popular optimization technique
- **trust-region methods**
  - quadratic approximations around current point $\mathbf{x}^{(k)}$
  - minimizes the model within region of trust

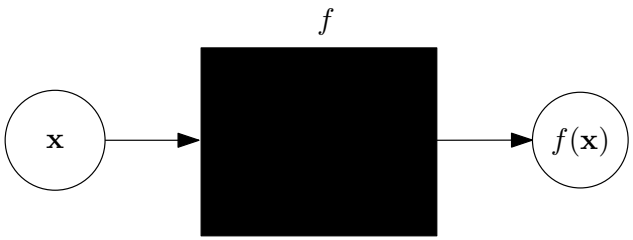
NEWOUA, BOBYQA (J. D. Powell, 2004, 2009)

- construct the quadratic model using much fewer points
  than $(n+1)(n+2)/2$ using additional minimizing a norm
- that saves time and enhances performance

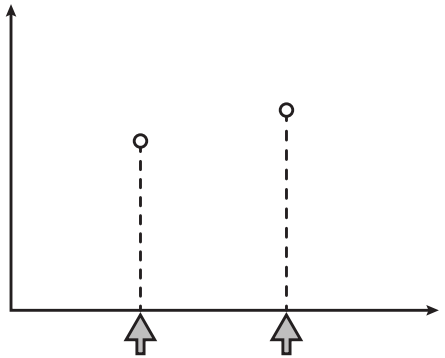
- **conjugate gradients**
  - do not approximate Hessians
  - *conjugate vectors* – a momentum guiding the search
  - cheaper variant to quasi-Newton's methods

Optimization of **black-box functions**

- **black-box functions**



- only evaluation of the function value, no derivatives or gradients $\rightarrow$ no gradient methods available

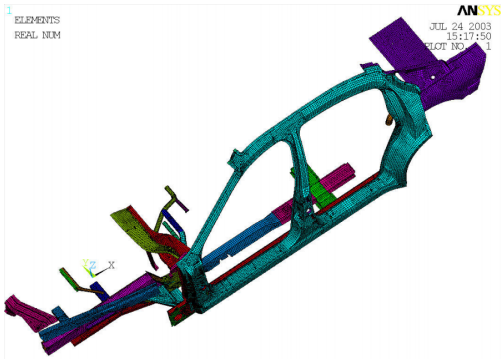


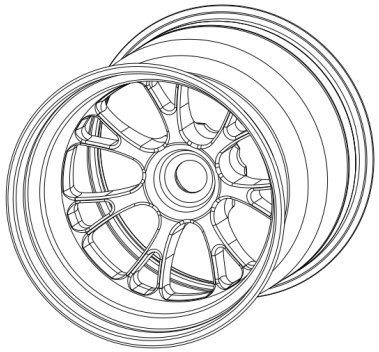
- we consider continuous domain: $\mathbf{x} \in \mathbb{R}^n$

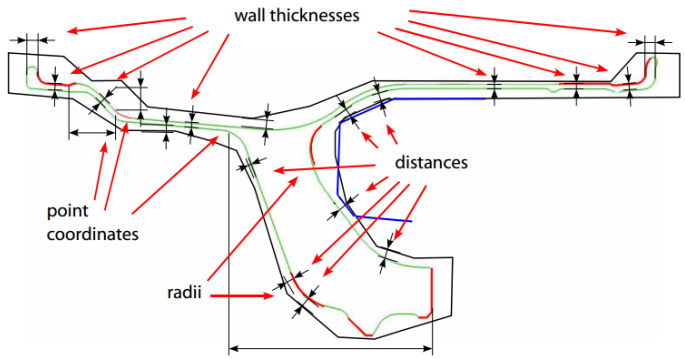
Optimization of **empirical** black-box functions

empirical function:

- assessing the function-value via an experiment (measuring, intensive calculation, evaluating a prototype)
- evaluating such functions are expensive (time and/or money)
- search cost $\sim$ the number of function evaluations







Metaheuristics

Metaheuristic

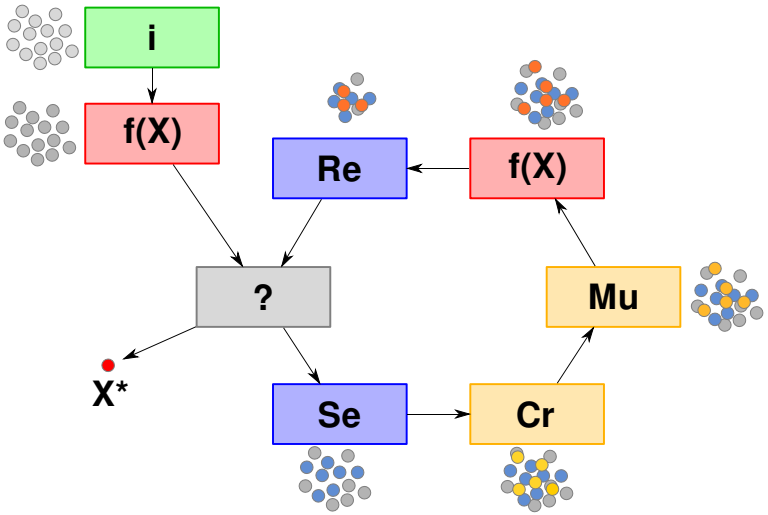
- optimization techniques finding sufficiently good solution
- treat the objective function as black-box
- sample a set of candidate solutions
  (search space often too large to be completely sampled)
- often nature-inspired

- particle/swarn optimization
- simulated annealing
- ...
- **evolutionary computation** (EA, GA, ES, ...)

EA's for empirical black-box optimization

what can help with decreasing
the number of function evaluations:

- utilize **already measured values**
  (at least prevent measuring the same thing twice)
- learn **the shape** of the function landscape
  or learn the (global) **gradient** or step direction & size



source: (GNU) Wikipedia, author: Johann "nojhan" Dréo

Model-based methods accelerating the convergence

several methods are used in order to **decrease**
the number of objective function **evaluations** needed by EA's

1. Bayesian optimization (EGO)
2. Surrogate modelling

Bayesian optimization

Bayesian optimizer

Input : objective function f, the size of the initial sample d
$\mathbf{x}_1, \ldots, \mathbf{x}_d \leftarrow$ generate an initial sample
$\mathcal{A} \leftarrow \{(\mathbf{x}_i, y_i)\}$ ** initialize the archive **

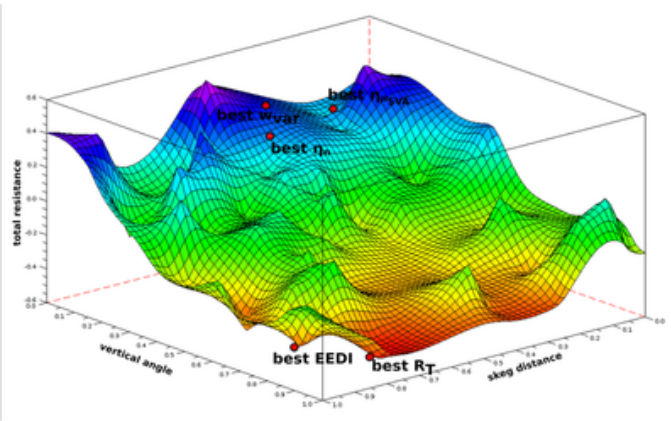
for *generation* $g = 1, 2, \ldots$ *until stopping conditions met* **do**
  $\mathcal{M} \leftarrow$ generate the probabilistic model based on $\mathcal{A}$
  $\mathbf{x}_1, \ldots \leftarrow$ choose next points $\mathbf{x} \in \mathcal{X}$ accord. to $\mathcal{C}_{\mathcal{M}}(\mathbf{x})$
  $y_1, \ldots \leftarrow f(\mathbf{x}_1), \ldots$ ** evaluate the new point(s) **
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{(\mathbf{x}_1, y_1), \ldots\}$ ** update the archive **

- suitable for very low budgets of f-evaluations ($\sim 10 \cdot D$)
- Gaussian processes used in the criterion $\mathcal{C}_{\mathcal{M}}$ most often
- existing algorithms: EGO (D. R. Jones, 1998), SPOT (T. Bartz-Beielstein, 2005), SMAC (F. Hutter, 2011) etc.

Surrogate modelling

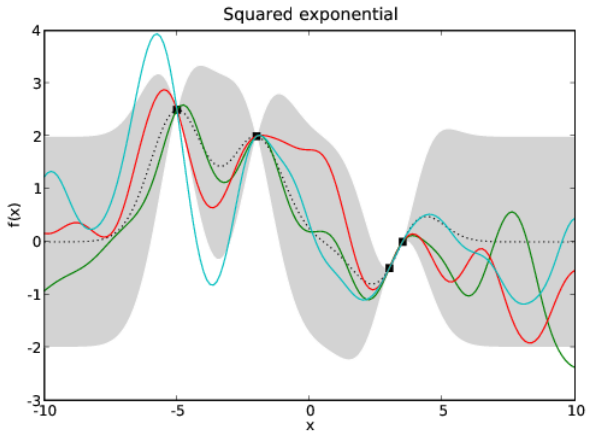
Surrogate modelling

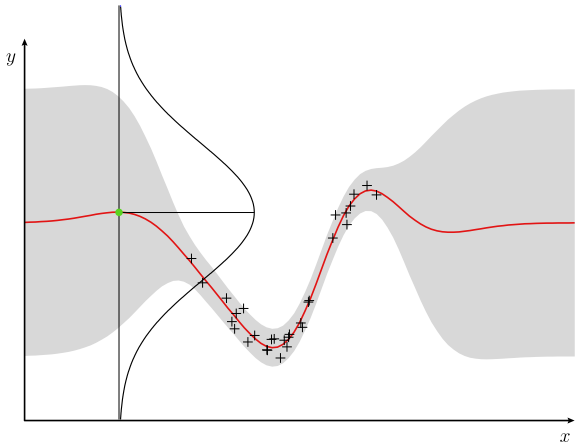
- technique which builds an **approximating model** of the fitness function landscape
- the model provides a cheap and fast, but also inaccurate replacement of the fitness function for part of the population
- inaccurate approximating model can **deceive** the optimizer



Gaussian Process

GP is a stochastic approximation method based on Gaussian distributions





GP can express **uncertainty** of the prediction in a new point $\mathbf{x}$: it gives a probability distribution of the output value

Gaussian Process

Gaussian Process

A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

A Gaussian process is completely specified by its

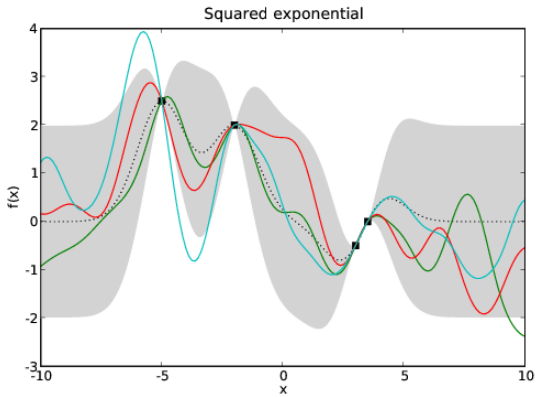
- mean function $m(\mathbf{x}) = \mathrm{E}\left[f_{GP}(\mathbf{x})\right]$
- covariance function $cov(\mathbf{x}_i, \mathbf{x}_j) = \mathrm{cov}(f_{GP}(\mathbf{x}_1), f_{GP}(\mathbf{x}_2))$

and we write the Gaussian process as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), cov(\mathbf{x}, \mathbf{x})).$$

(Rasmussen, Williams, 2006)

Gaussian Process



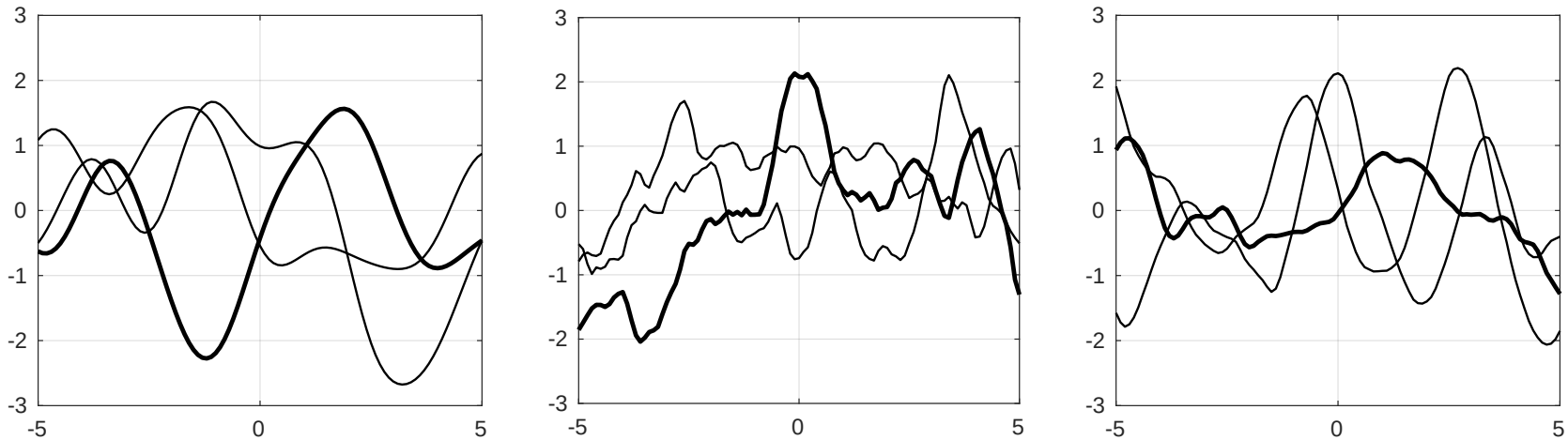
- given a set of N training points $\mathbf{X}_N = (\mathbf{x}_1 \dots \mathbf{x}_N)^\top$, $\mathbf{x}_i \in \mathbb{R}^d$, and measured values $\mathbf{y}_N = (y_1, \dots, y_N)^\top$ of a function f being approximated

$$y_i = f(\mathbf{x}_i), \quad i = 1, \dots, N$$

GP considers vector of these function values as a sample from N-variate Gaussian distribution

$$\mathbf{y}_N \sim \mathrm{N}(\mathbf{0}, \mathbf{C}_N)$$

Gaussian Process prior distribution



Draws from Gaussian processes prior for three different covariance functions K_{SE}, $K_{\text{Matern}}^{\nu=3/2}$, $K_{\text{Matern}}^{\nu=5/2}$ (in that order), all of them with the parameters $\ell = 1$ and $\sigma_f^2 = 1$ without noise

Gaussian Process prediction (posterior)

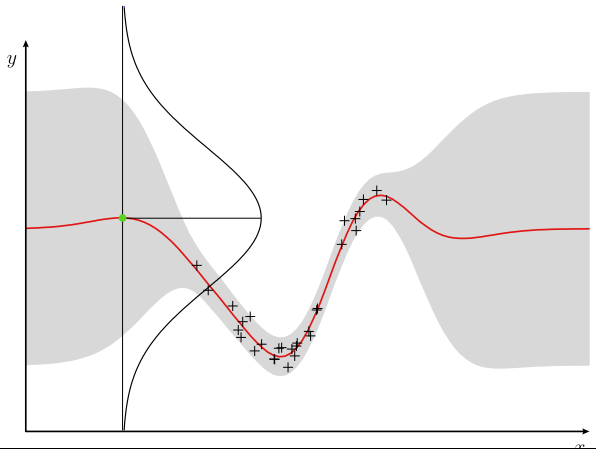
Making predictions

Let $\mathbf{C}_{N+1}$ be extended covariance matrix – extended by entries belonging to an unseen point $(\mathbf{x}, y^*)$. Because $\mathbf{y}_N$ is known and

the inverse $\mathbf{C}_{N+1}^{-1}$ can be expressed using inverse of the training covariance $\mathbf{C}_N{}^{-1}$,

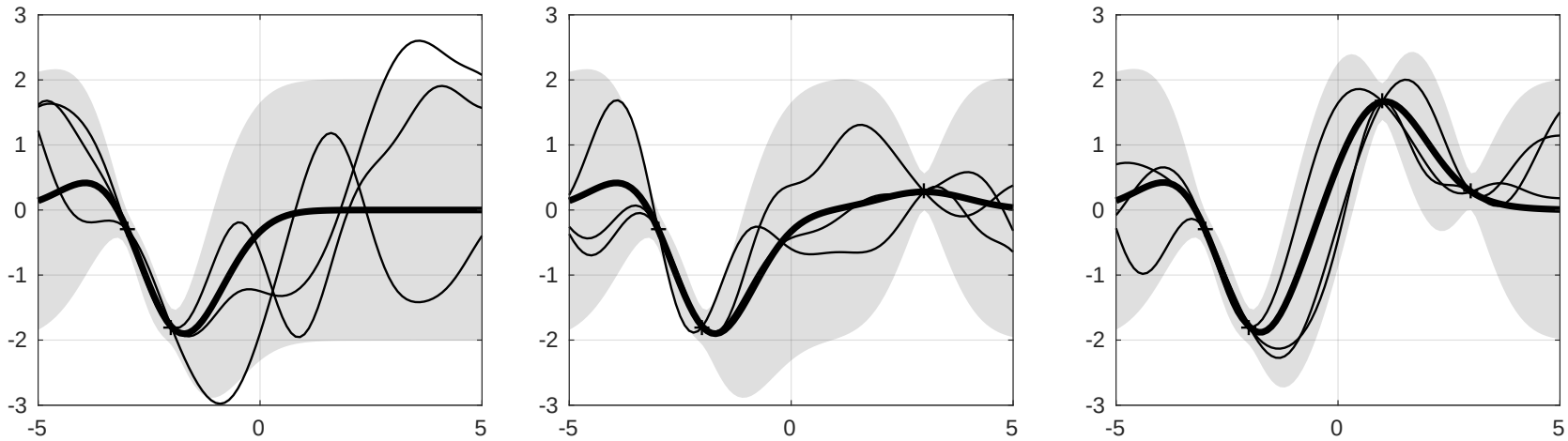
the density in a new point marginalize to 1D Gaussian density

$$p(y^* \mid \mathbf{X}_{N+1}, \mathbf{y}_N) \propto \exp\left(-\frac{1}{2}\frac{(y^* - \hat{y}_{N+1})^2}{s^2_{y_{N+1}}}\right)$$



where
the mean $\hat{y}_{N+1}$ and the variance $s^2_{y_{N+1}}$
is easily expressible from $\mathbf{C}_N{}^{-1}$ and $\mathbf{y}_N$.

Gaussian Process prediction (posterior)



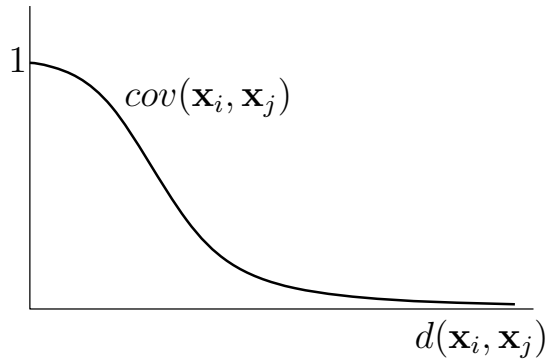
Graphs of Gaussian processes prediction $N = 2, 3, 4$ training data. $(+)$ – training set, thick line – mean prediction, thin lines – three draws from the GP posterior (without noise). Predictions $\hat{y}^*$ and $\pm 2\hat{s}^*$ are generated for 101 points, computationally stable as the matrix inversion only for the *training* covariace $\mathbf{C}_N$.

Gaussian Process covariance

The covariance matrix $\mathbf{C}_N$ is determined by the covariance function $cov(\mathbf{x}_i, \mathbf{x}_j)$ which is defined on pairs from the input space

$$(\mathbf{C})_{ij} = cov(\mathbf{x}_i, \mathbf{x}_j), \quad \mathbf{x}_{i,j} \in \mathbb{R}^d$$

expressing the degree of correlations between two points' values; typically decreasing functions on two points distance



Gaussian Process covariance

The most frequent covariance function is *squared-exponential*

$$(\mathbf{K})_{ij} = cov^{\mathrm{SE}}(\mathbf{x}_i, \mathbf{x}_j) = \theta \exp\left(\frac{-1}{2\ell^2}(\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{x}_i - \mathbf{x}_j)\right)$$

with the parameters (usually fitted by MLE)

- θ – signal variance (scales the correlation)
- ℓ – characteristic length scale

Gaussian Process covariance

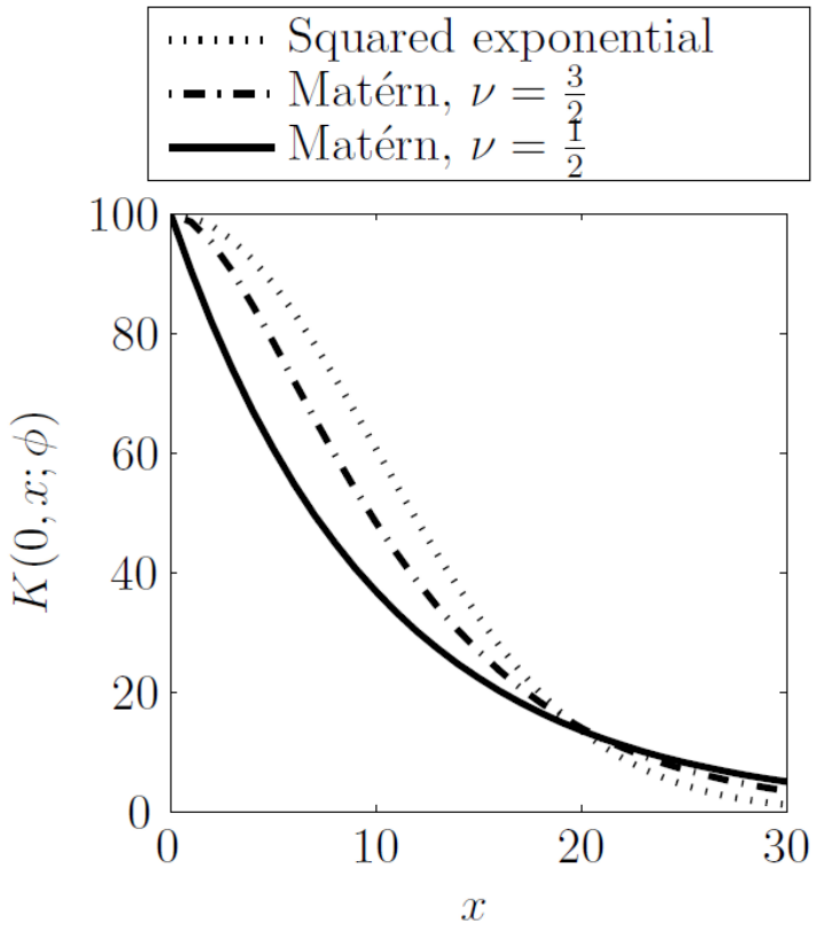
Another usual option in data-minig applications is *Matérn covariance*, which is for $r = (\mathbf{x}_i - \mathbf{x}_j)$

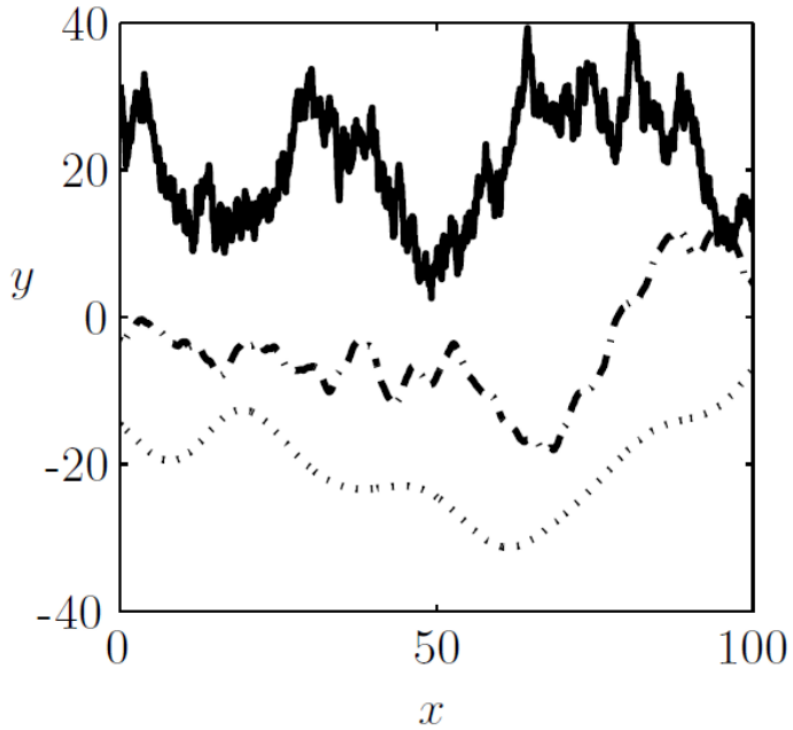
$$(\mathbf{K})_{ij} = cov_{\nu=5/2}^{\text{Matern}}(r) = \theta \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right).$$

with the parameters (same as for squared exponential)

- θ – signal variance
- ℓ – characteristic length scale

Gaussian Process covariance





source: (Rasmussen and Williams, 2006)

Stochastic search of Evolutionary algorithms

Stochastic black box search

initilize distribution parameters θ
set population size $\lambda \in \mathbb{N}$
while not terminate

1. sample distribution $P(\mathbf{x}|\theta) \rightarrow \mathbf{x}_1, \ldots, \mathbf{x}_\lambda \in \mathbb{R}^n$
2. evaluate $\mathbf{x}_1, \ldots, \mathbf{x}_\lambda$ on f
3. update parameters θ

(A. Auger, Tutorial CMA-ES, GECCO 2013)

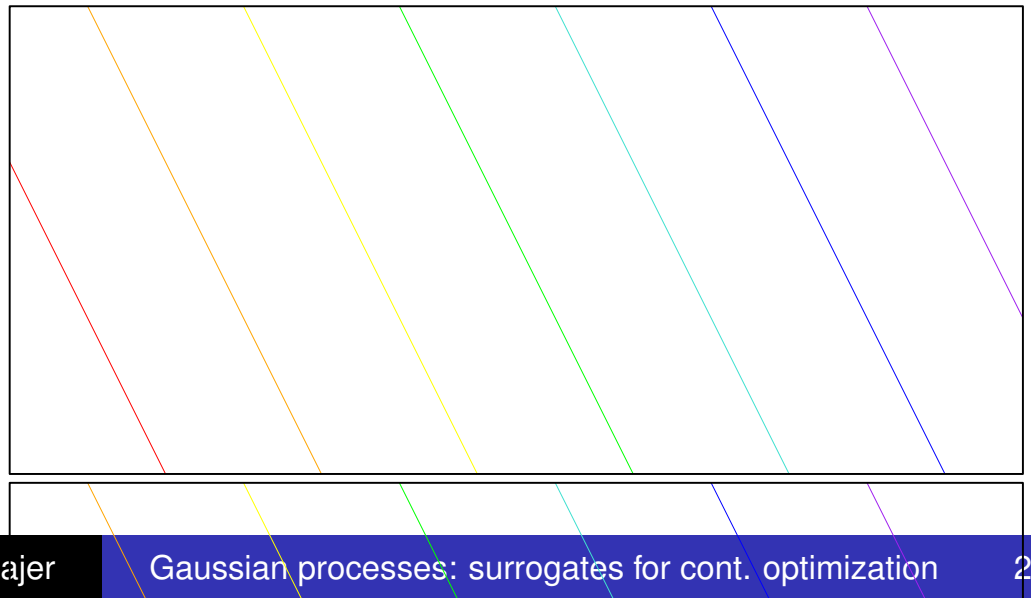
- schema of most of the evolutionary strategies (and EDA algorithms)
- as well as CMA-ES (Covariance Matrix Adaptation ES) – current state of the art in continuous optimization

The CMA-ES

Input: $\mathbf{m} \in \mathbb{R}^n, \sigma \in \mathbb{R}_+, \lambda \in \mathbb{N}$
Initialize: $\mathbf{C} = \mathbf{I}$ (and several other parameters)
Set the weights $w_1, \ldots w_\lambda$ appropriately

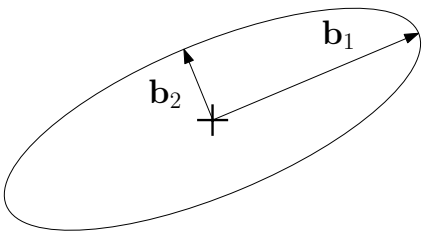
while not terminate

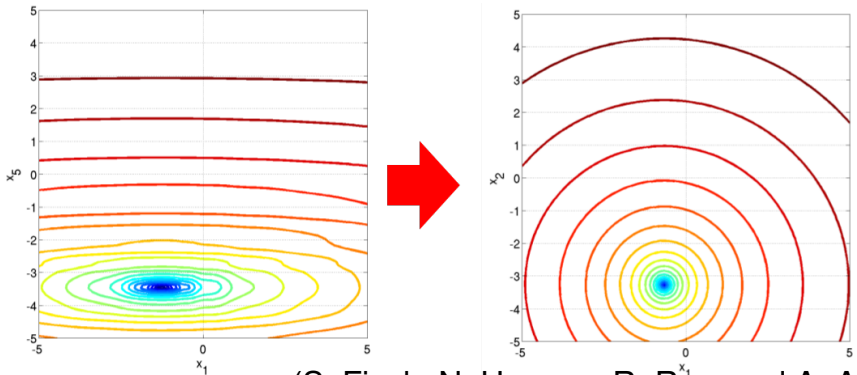
1. $\mathbf{x}_i = \mathbf{m} + \sigma \mathbf{y}_i, \quad \mathbf{y}_i \sim N(\mathbf{0}, \mathbf{C}), \quad$ for $i = 1, \ldots, \lambda$ sampling
2. $\mathbf{m} \leftarrow \sum_{i=1}^{\mu} w_i \, \mathbf{x}_{i:\lambda} = \mathbf{m} + \sigma \mathbf{y}_w \quad$ where $\mathbf{y}_w = \sum_{i=1}^{\mu} w_i \, \mathbf{y}_{i:\lambda}$ update mean
3. update $\mathbf{C}$
4. update step-size σ



Covariance matrix adaptation

- eigenvectors of the covariance matrix $\mathbf{C}$ are the principle components – the principle axes of the mutation ellipsoid
- CMA-ES learns and updates a new Mahalanobis metric
- successively approximates the inverse Hessian on quadratic functions
  – transforms ellipsoid function into sphere function
  – it somehow holds for other functions, too (up to some degree)



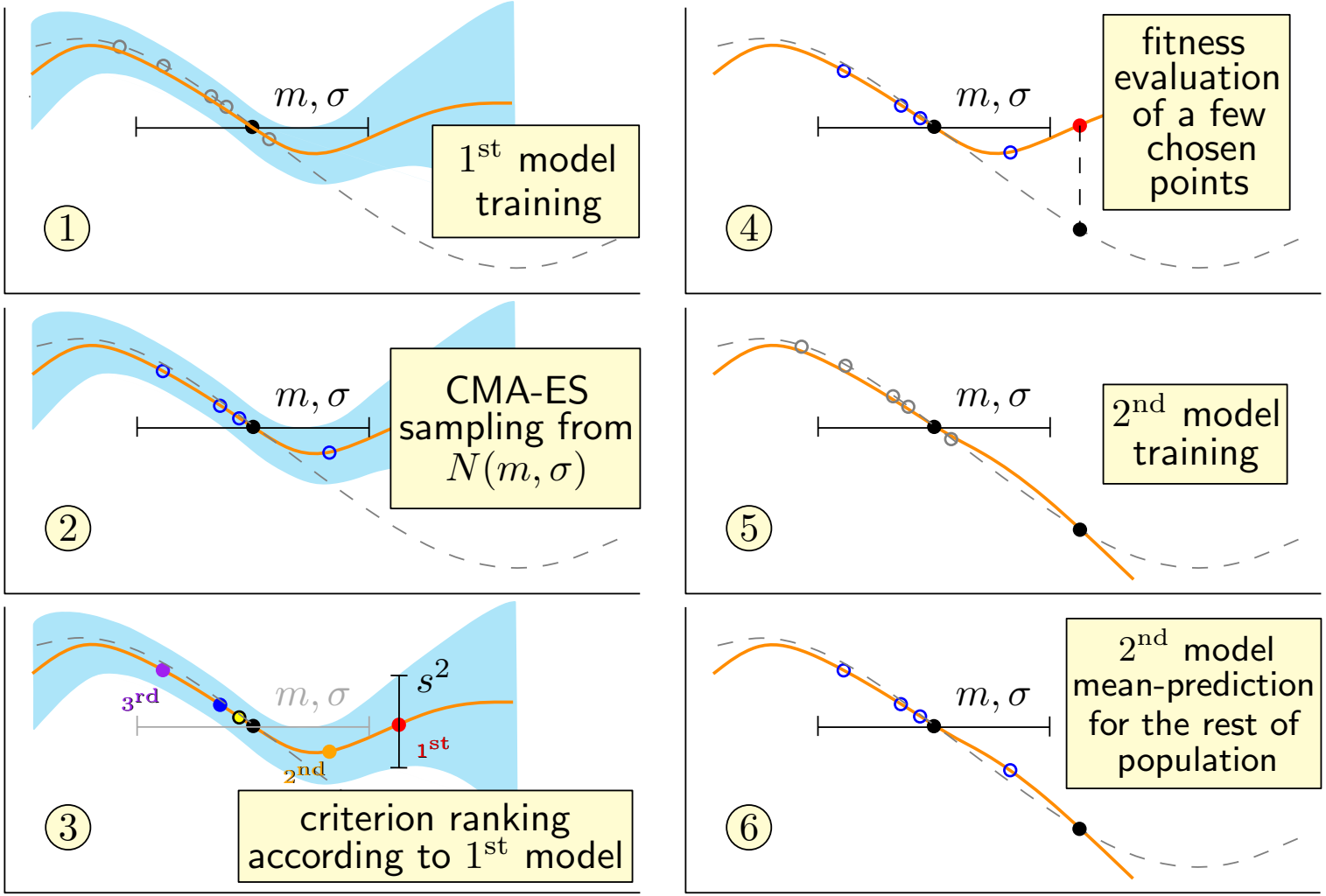


source: (S. Finck, N. Hansen, R. Ros, and A. Auger, 2009)

Is the CMA-ES the best for everything?

- CMA-ES is state-of-the-art optimization algorithm, especially for rugged and ill-conditioned objective functions
- however, **not the fastest** if we can afford only very few objective function evaluations
- what we have already seen: use a surrogate model!
- however, original evaluated solutions are available only along the search path
- solution: construct local surrogate models

Doubly trained Surrogate CMA-ES



Doubly trained Surrogate CMA-ES

1. sample a new population of size λ (standard CMA-ES offspring),
2. train the *first* surrogate model on the original-evaluated points from the archive $\mathcal{A}$,
3. select $\lceil \alpha\lambda \rceil$ point(s) wrt. a criterion $\mathcal{C}$, which is based on the *first* model's prediction,
4. evaluate these point(s) with the original fitness,
5. re-train the surrogate model also using these new point(s), and
6. predict the fitness for the non-original evaluated points with this *second* model.

Criteria for the selection of original-evaluated points

- GP predictive mean

$$\mathcal{C}_{\mathsf{M}}(\mathbf{x}) = -\hat{y}(\mathbf{x})$$

- GP predictive standard deviation

$$\mathcal{C}_{\mathsf{STD}}(\mathbf{x}) = \hat{s}(\mathbf{x})$$

Criteria for the selection of original-evaluated points

- Expected improvement (EI). $y_{\min}$ – the minimum so-far fitness

$$\mathcal{C}_{\mathrm{EI}}(\mathbf{x}) = E((y_{\min} - \hat{f}(\mathbf{x}))I(\hat{f}(\mathbf{x}) < y_{\min}) \,|\, y_1, \ldots, y_N)\,, \text{ where}$$

$$I(f(\mathbf{x}) < y_{\min}) = \begin{cases} 1 & \text{for } \hat{f}(\mathbf{x}) < y_{\min} \\ 0 & \text{for } \hat{f}(\mathbf{x}) \geq y_{\min} \end{cases}$$

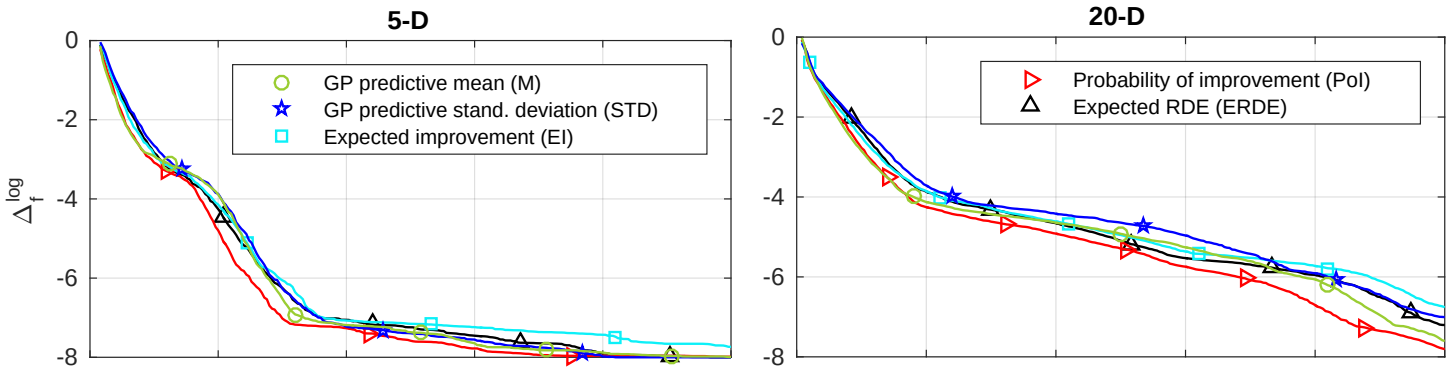
- Probability of improvement (PoI). the probability of finding lower fitness than some threshold T

$$\mathcal{C}_{\mathrm{PoI}}(\mathbf{x}, T) = P(f(\mathbf{x}) \leq T \,|\, y_1, \ldots, y_N) = \Phi\left(\frac{T - \hat{y}(\mathbf{x})}{\hat{s}(\mathbf{x})}\right)$$

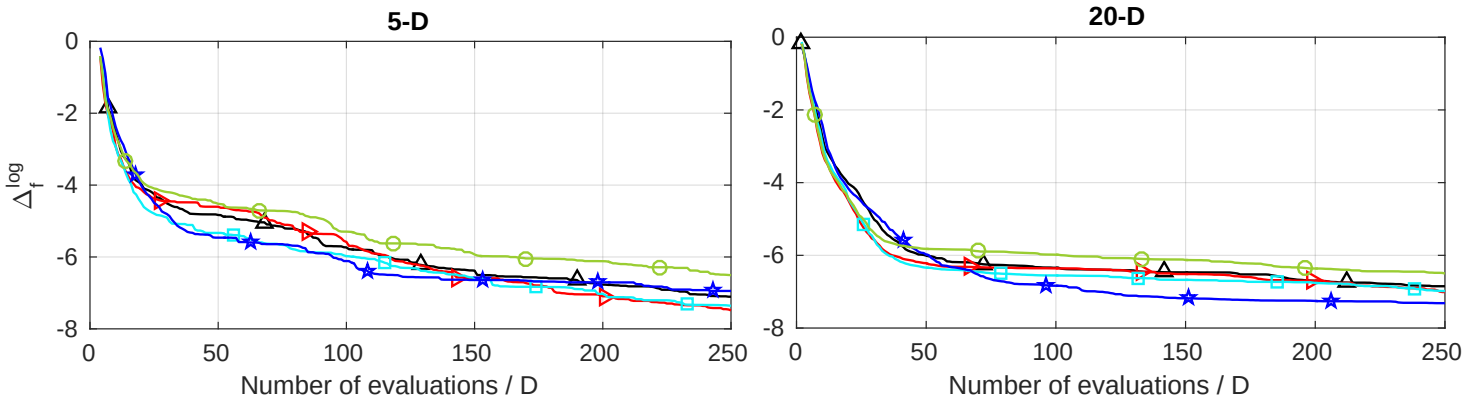
where Φ is the CDF of $\mathcal{N}(0, 1)$, $T = y_{\min}$ or a slightly higher value

Criteria for the selection of original-evaluated points

selected unimodal COCO functions $f_{1,2}, f_{8\ldots14}$



multimodal COCO functions $f_{3,4}, f_{15\ldots24}$



The $\log_{10}$ of the median best f-value distances to the benchmarks' optima were scaled linearly to $[-8, 0]$ for each COCO function.

GP model training

trainModel($\mathcal{A}, N_{\text{max}}, TSS, r^{\mathcal{A}}_{\text{max}}, K, \sigma^{(g)}, \mathbf{C}^{(g)}, \mathbf{m}^{(g)}$)

$(\mathbf{X}_N, \mathbf{y}_N) \leftarrow$ select at most N_{max} points from the archive $\mathcal{A}$ using TSS and $r^{\mathcal{A}}_{\text{max}}$

$\mathbf{X}_N \leftarrow$ transform the selected points into the $(\sigma^{(g)})^2\mathbf{C}^{(g)}$ basis with the origin at $\mathbf{m}^{(g)}$

$\mathbf{y}_N \leftarrow$ standardize the f-values in $\mathbf{y}_N$ to zero mean and unit variance

$(m_\mu, \sigma_f^2, \ell, \sigma_n) \leftarrow$ fit the hyperparameters of $\mu(\mathbf{x})$ and K using ML estimation

---

Training set selection

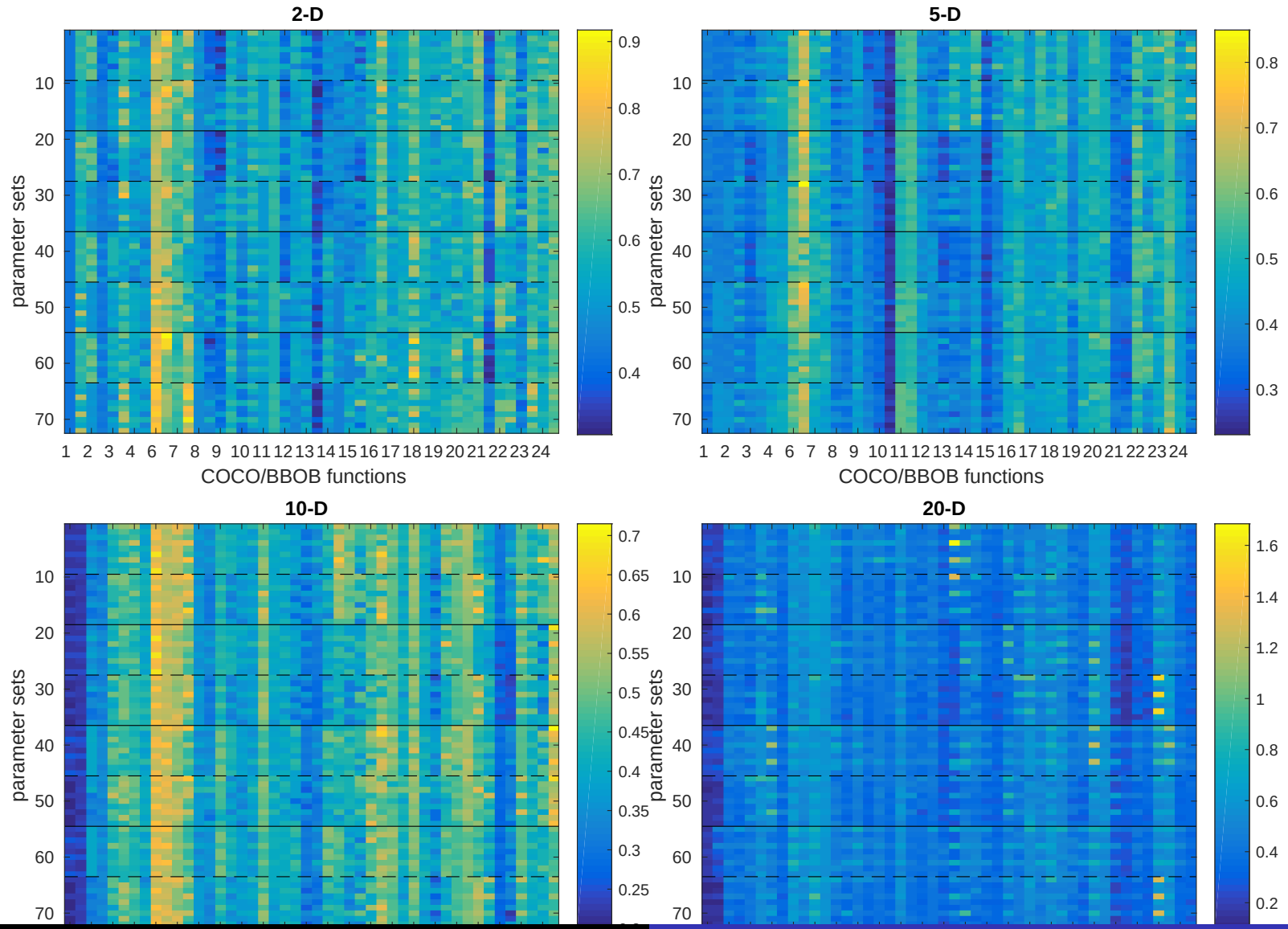
1. TSS1 taking up to N_{max} most recently evaluated points
2. TSS2 selecting the union of the k nearest neighbors of every point for which the fitness should be predicted, where k is maximal such that the total number of selected points does not exceed N_{max},
3. TSS3 clustering the points in the input space into N_{max} clusters and taking the points nearest to clusters' centroids
4. TSS4 selecting N_{max} points which are closest to any point in the current population.

GP model parameters

parameter	**considered values**
training set selection method TSS	TSS1, TSS2 TSS3, TSS4
maximum distance $r^{\mathcal{A}}_{\text{max}}$	$2\sqrt{Q_{\chi^2}(0.99, D)}$, $4\sqrt{Q_{\chi^2}(0.99, D)}$
N_{max}	$10 \cdot D$, $15 \cdot D$, $20 \cdot D$
covariance function K	K_{SE}, $K^{\nu=3/2}_{\text{Matern}}$, $K^{\nu=5/2}_{\text{Matern}}$

Parameters of the GP surrogate models. The maximum distance $r^{\mathcal{A}}_{\text{max}}$ is derived using the Mahalanobis distance given by the covariance matrix $\sigma^2\mathbf{C}$. $Q_{\chi^2}(0.99, D)$ is the 0.99-quantile of the χ^2_D distribution, and therefore $\sqrt{Q_{\chi^2}(0.99, D)}$ is the 0.99-quantile of the norm of a D-dimensional normal distributed random vector.

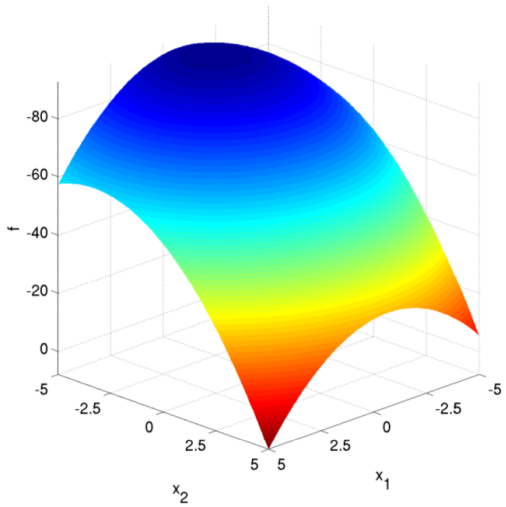
Gaussian process parameter settings – heatmap



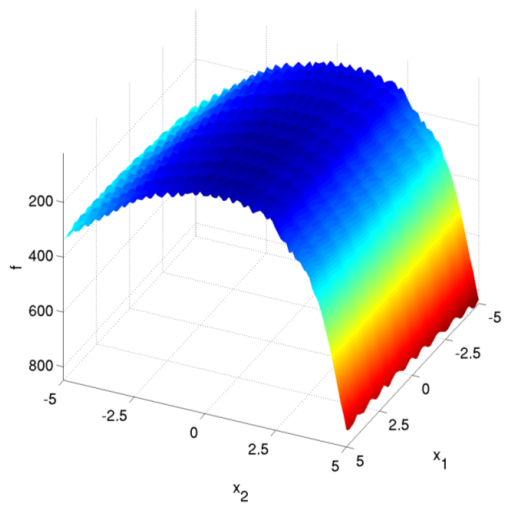
Testing framework

Black-Box Optimization Benchmarking (BBOB)
COmparing Continuous Optimisers (COCO)

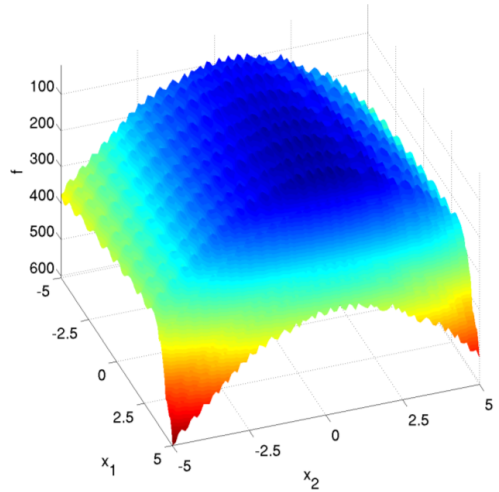
- 24 artificial functions
- different degree of separability, conditioning, modality or with or without a global structure
- testing sets defined for dimensions 2, 3, 5, 10, 20 (and 40:)



f_1

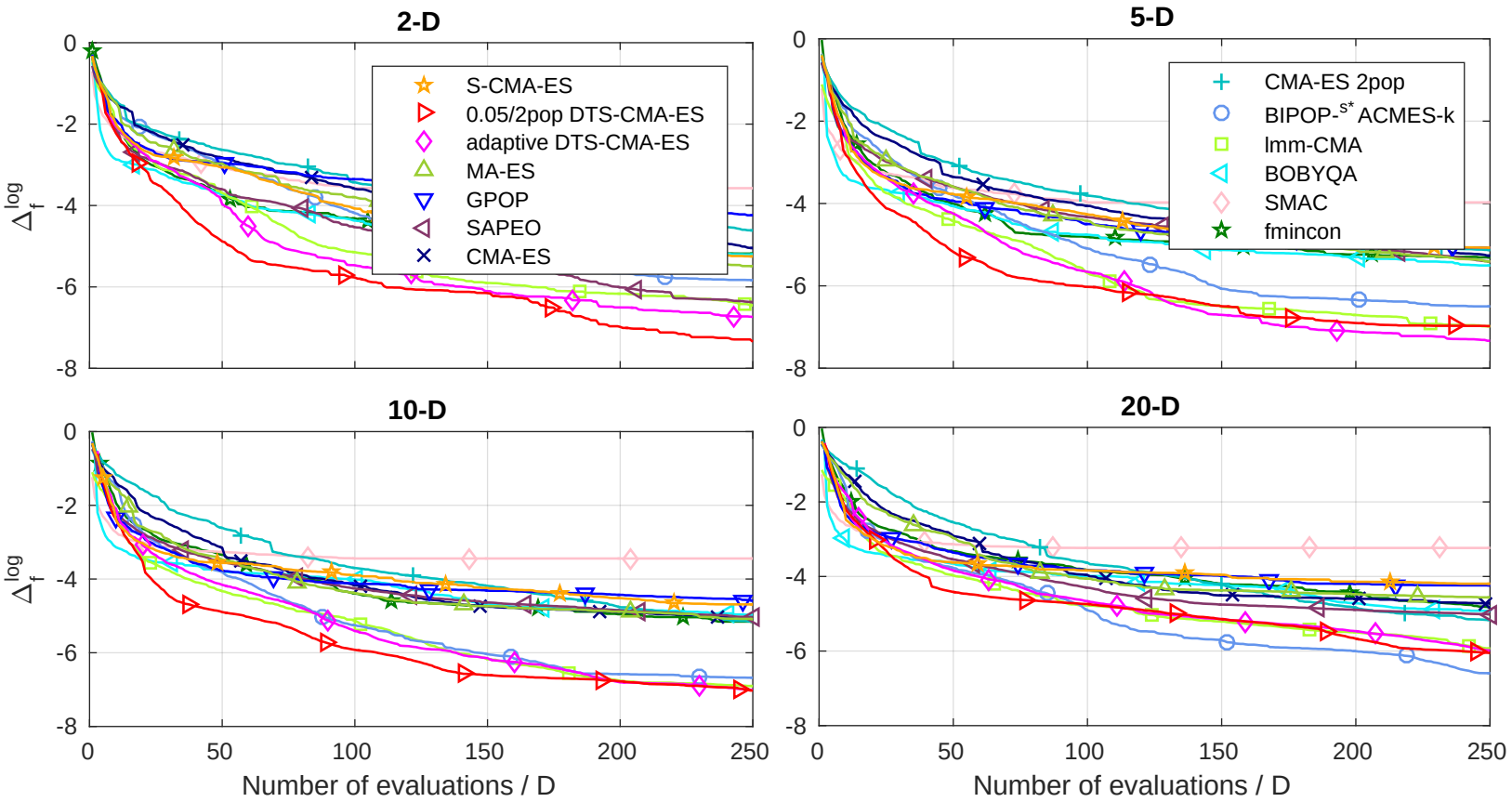


f_3

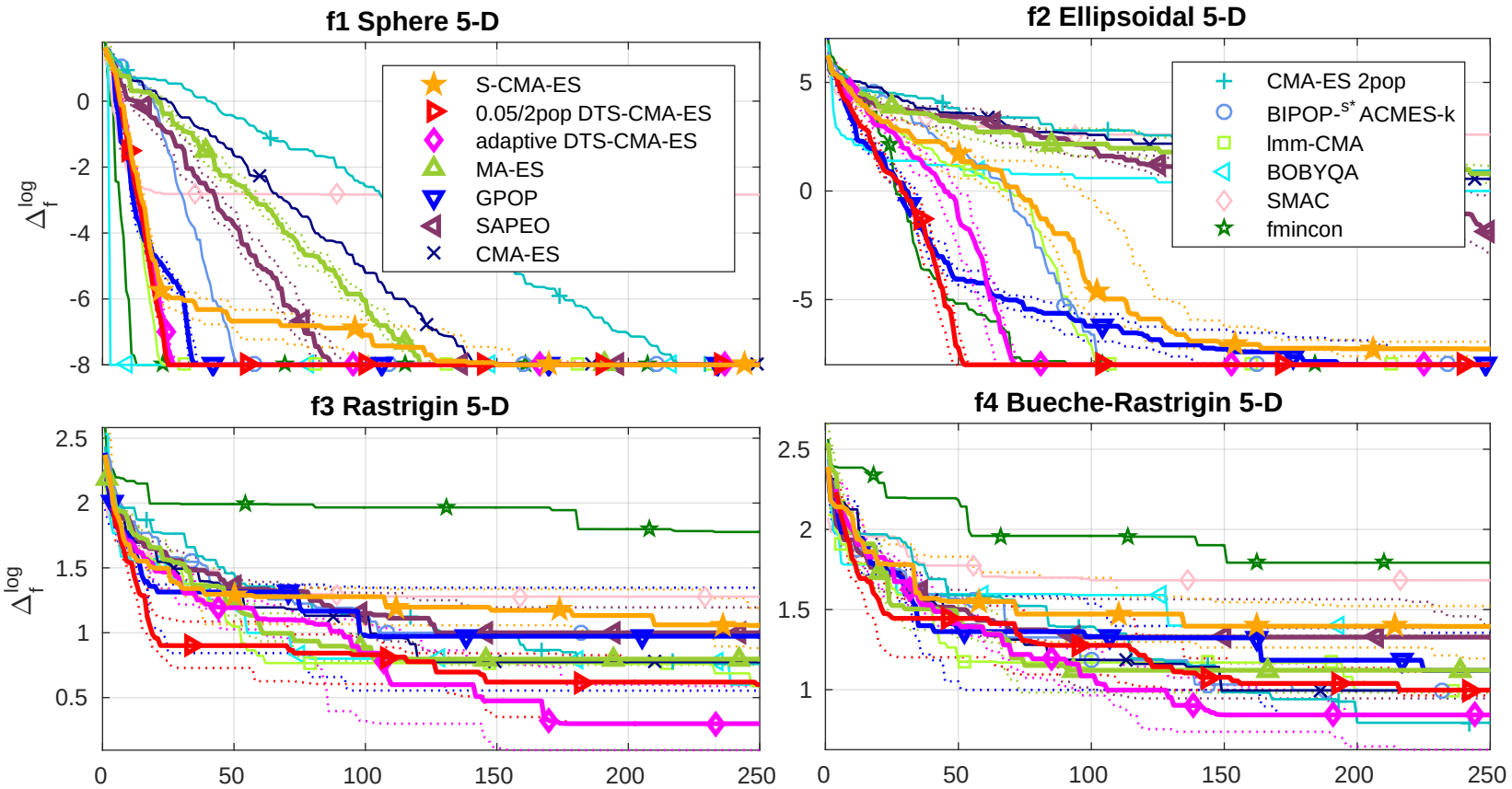


f_4

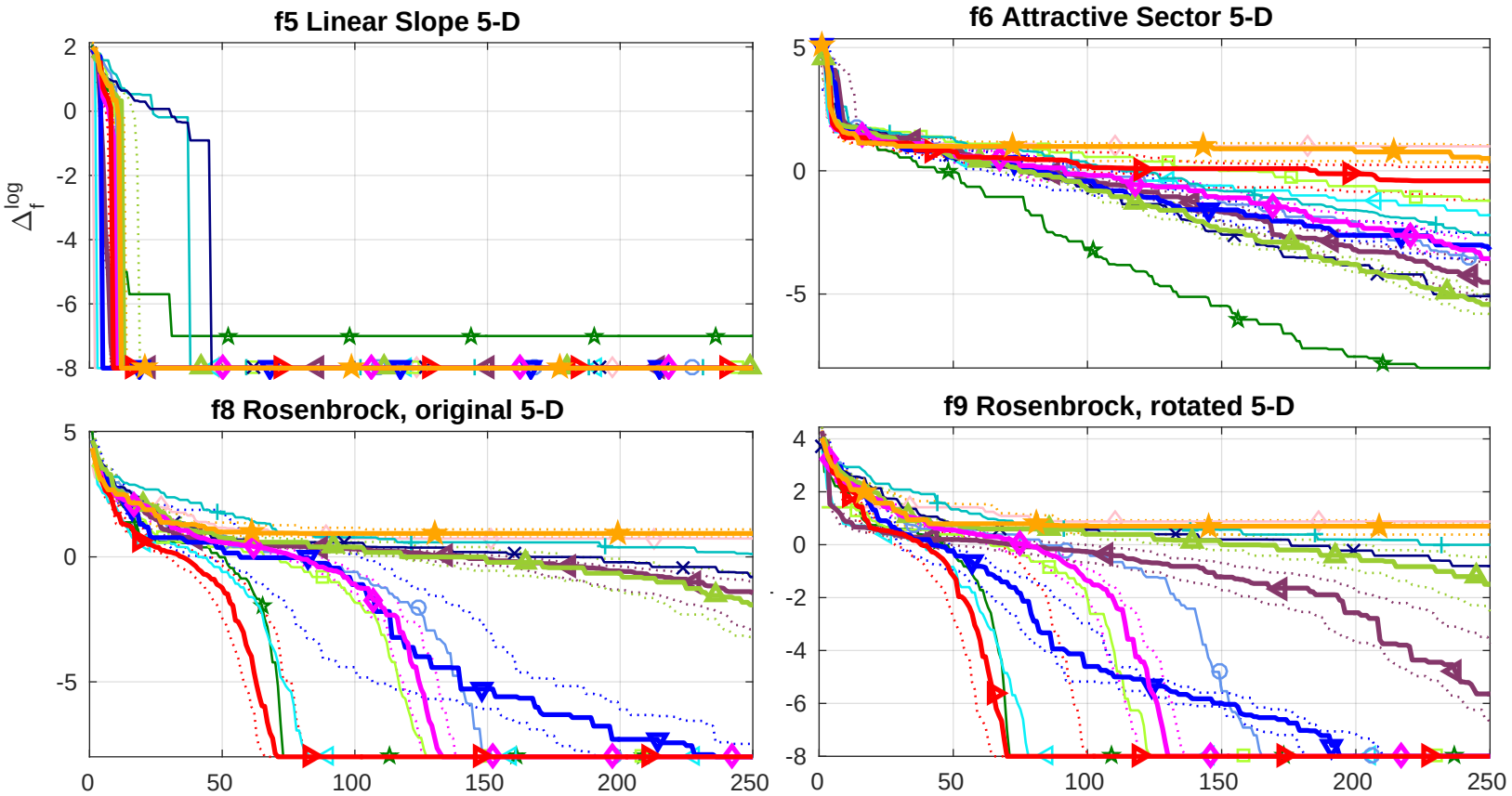
Aggregated experimental results on BBOB



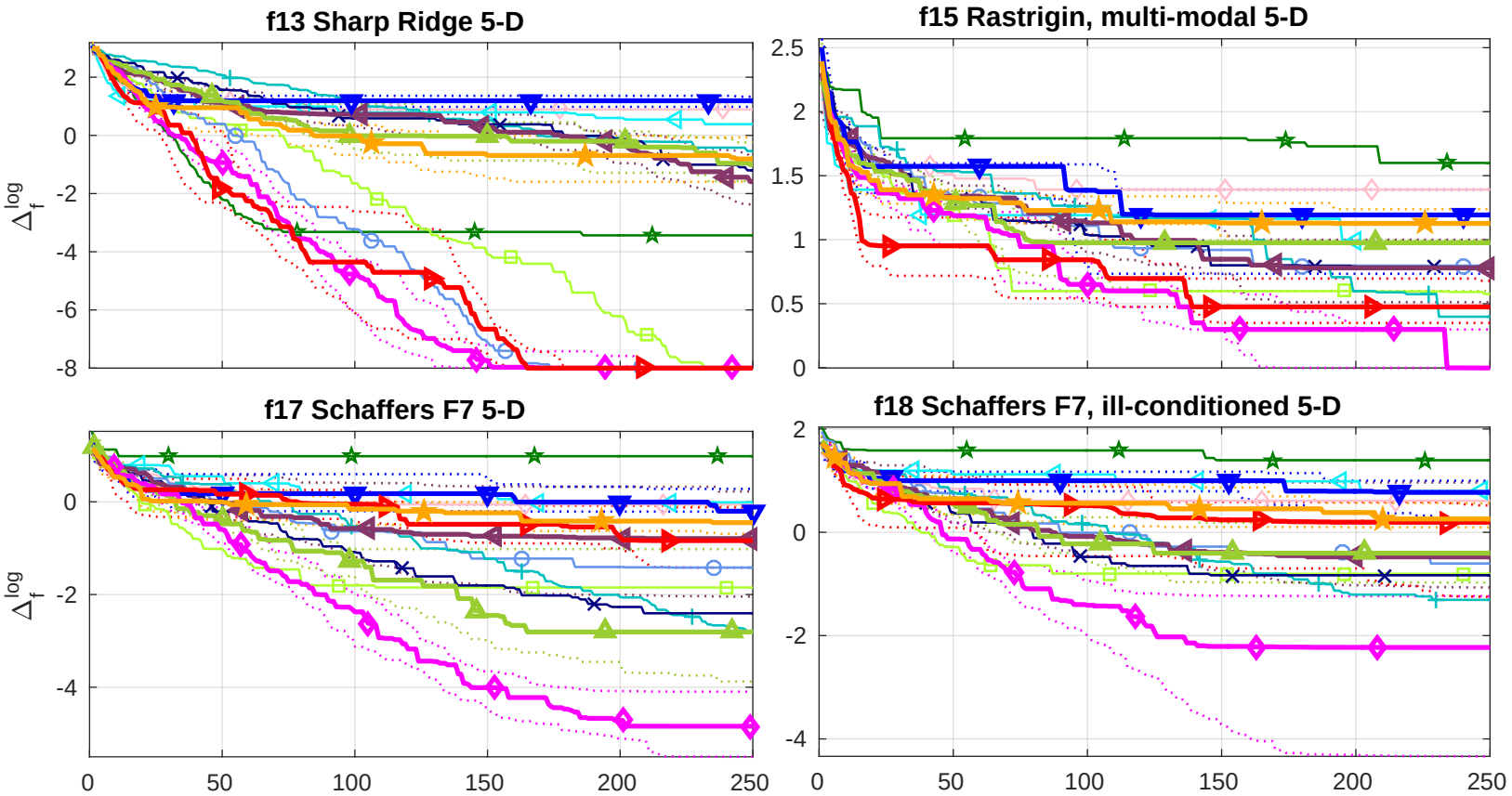
Experimental results on BBOB (5 D)



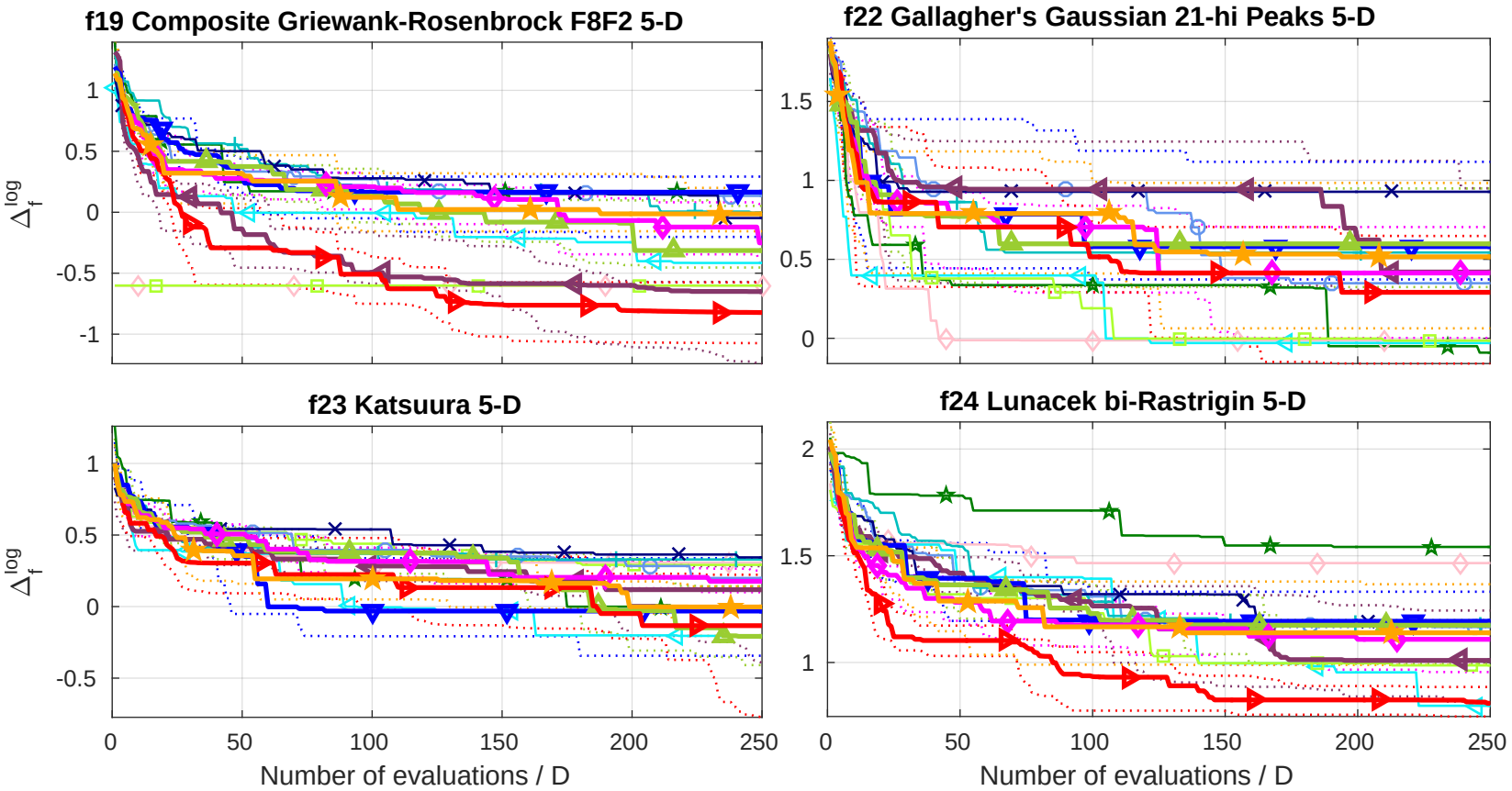
Experimental results on BBOB (5 D)



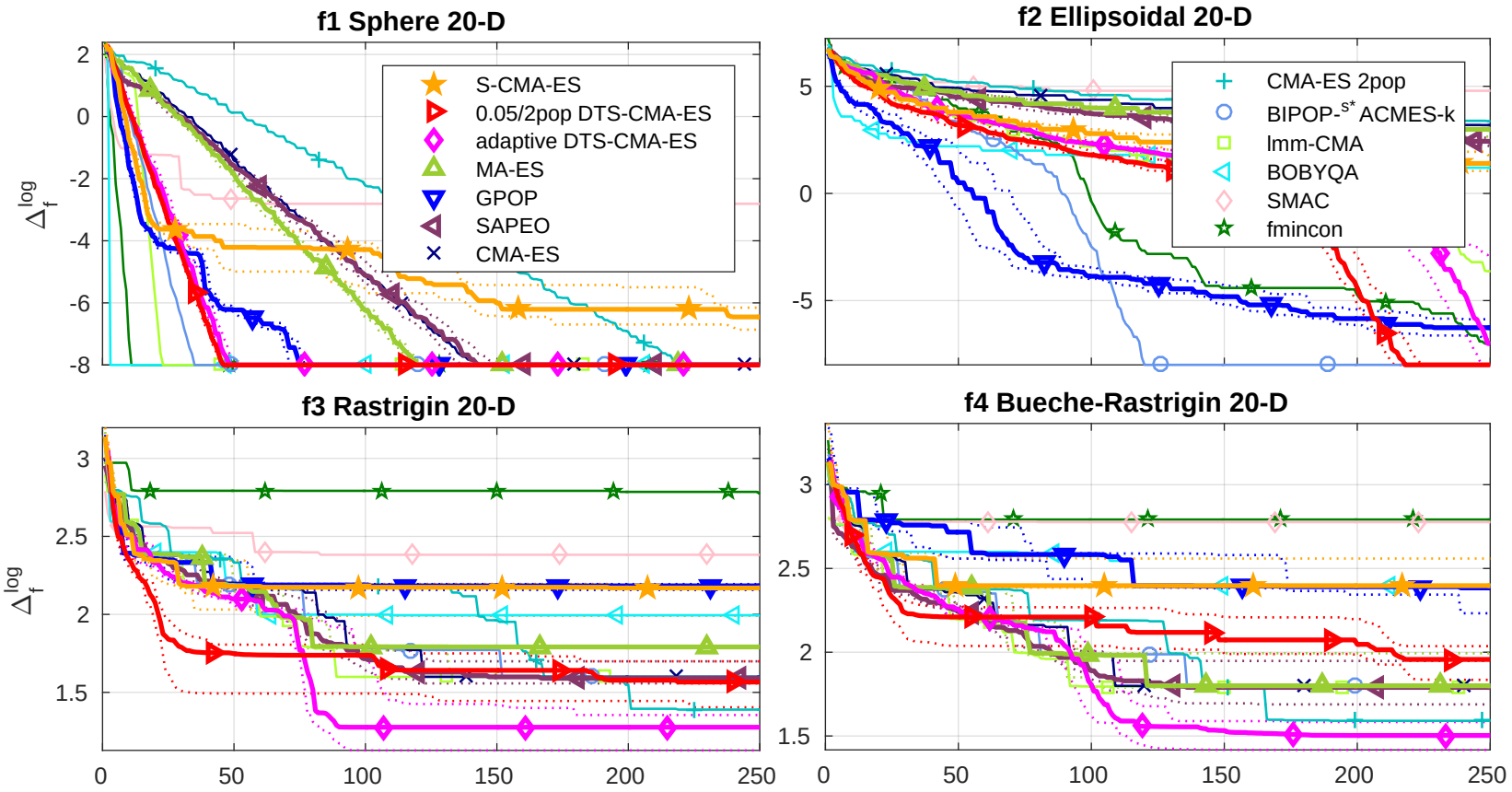
Experimental results on BBOB (5 D)



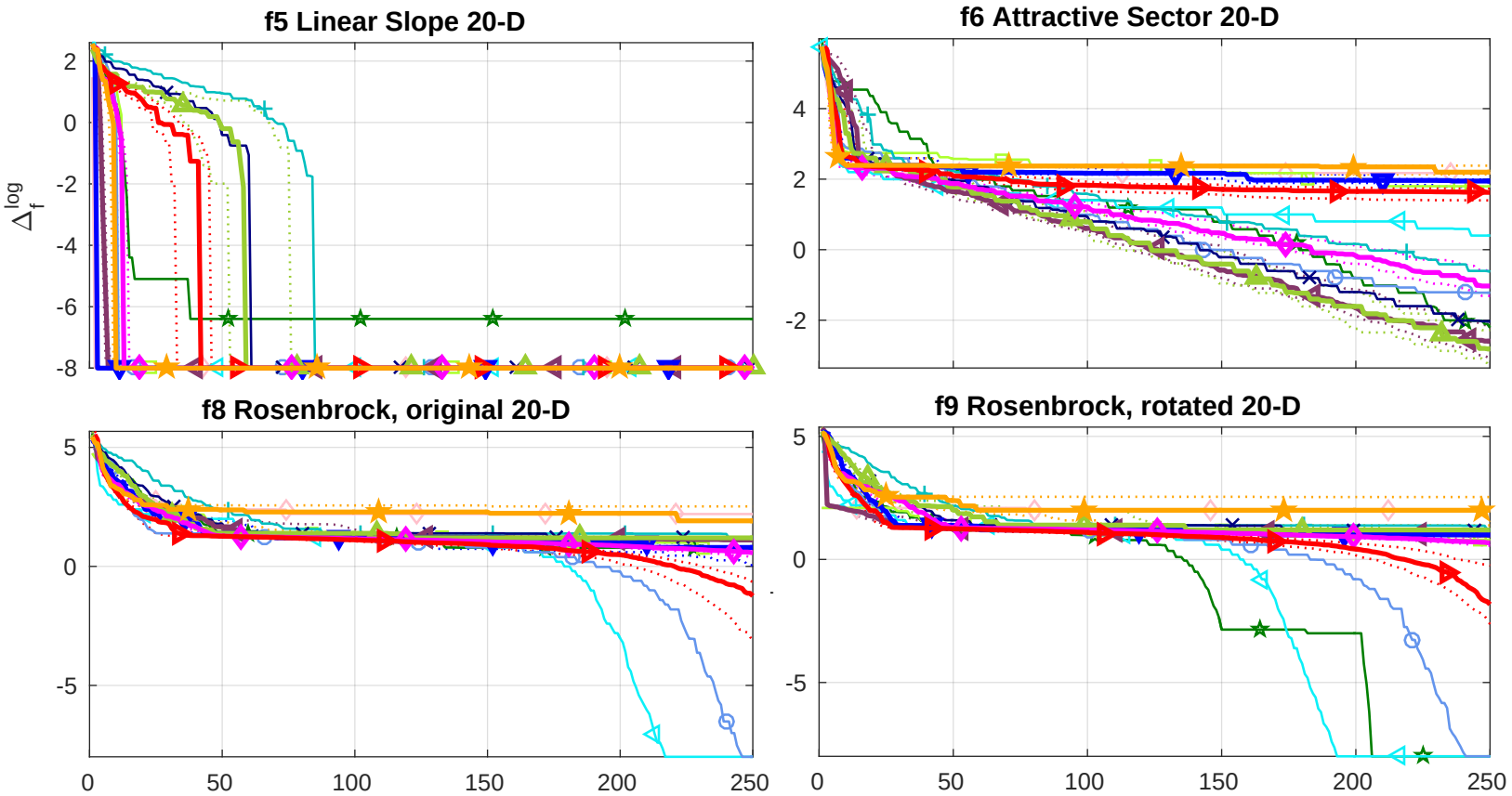
Experimental results on BBOB (5 D)



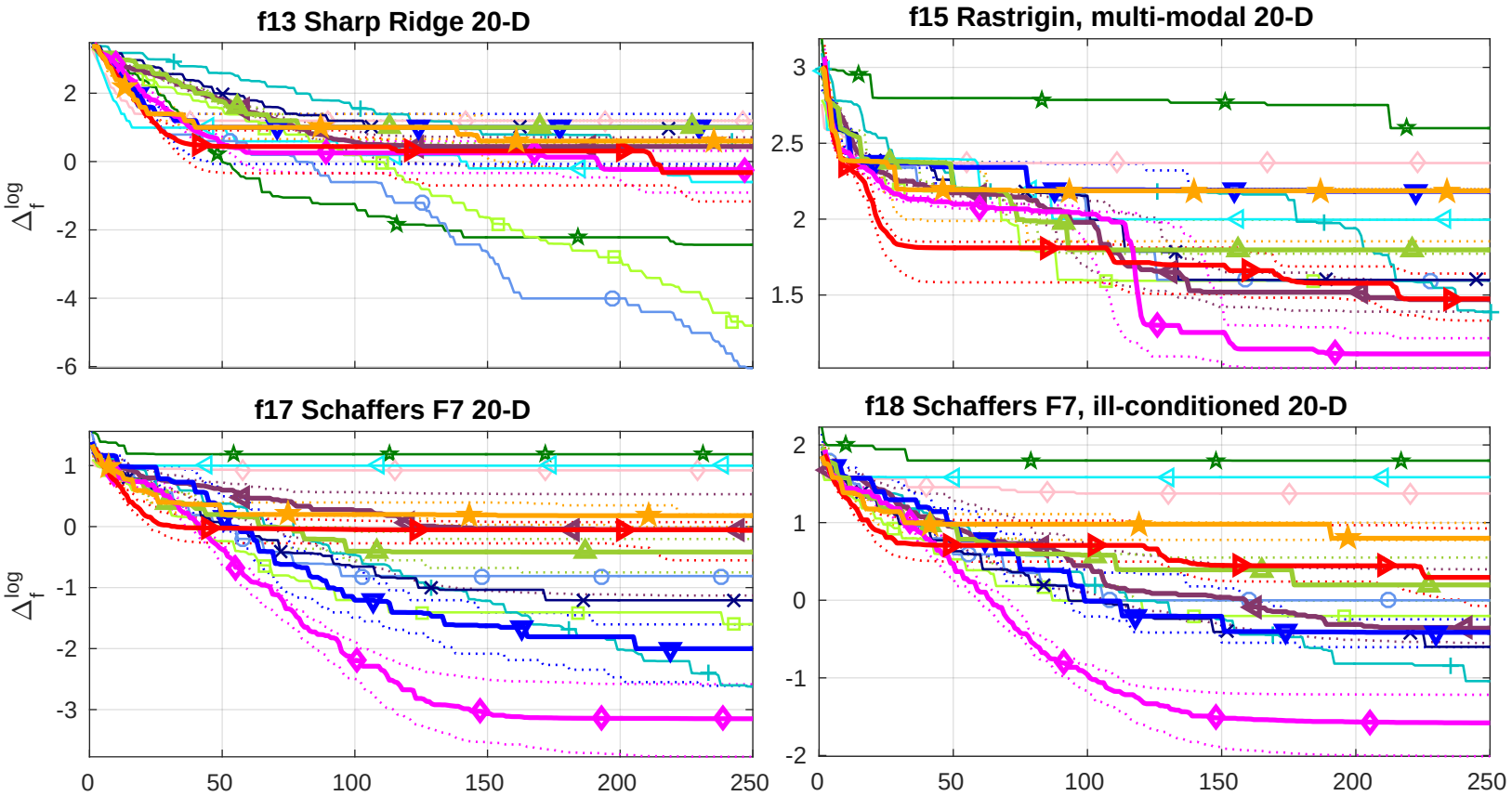
Experimental results on BBOB (20 D)



Experimental results on BBOB (20 D)



Experimental results on BBOB (20 D)



Experimental results on BBOB (20 D)