Exploiting Structure in Bayesian Optimization using Multi-Fidelity Probabilistic Models

Boris Bogaerts, Rudi Penne

Op3Mech
University of Antwerp

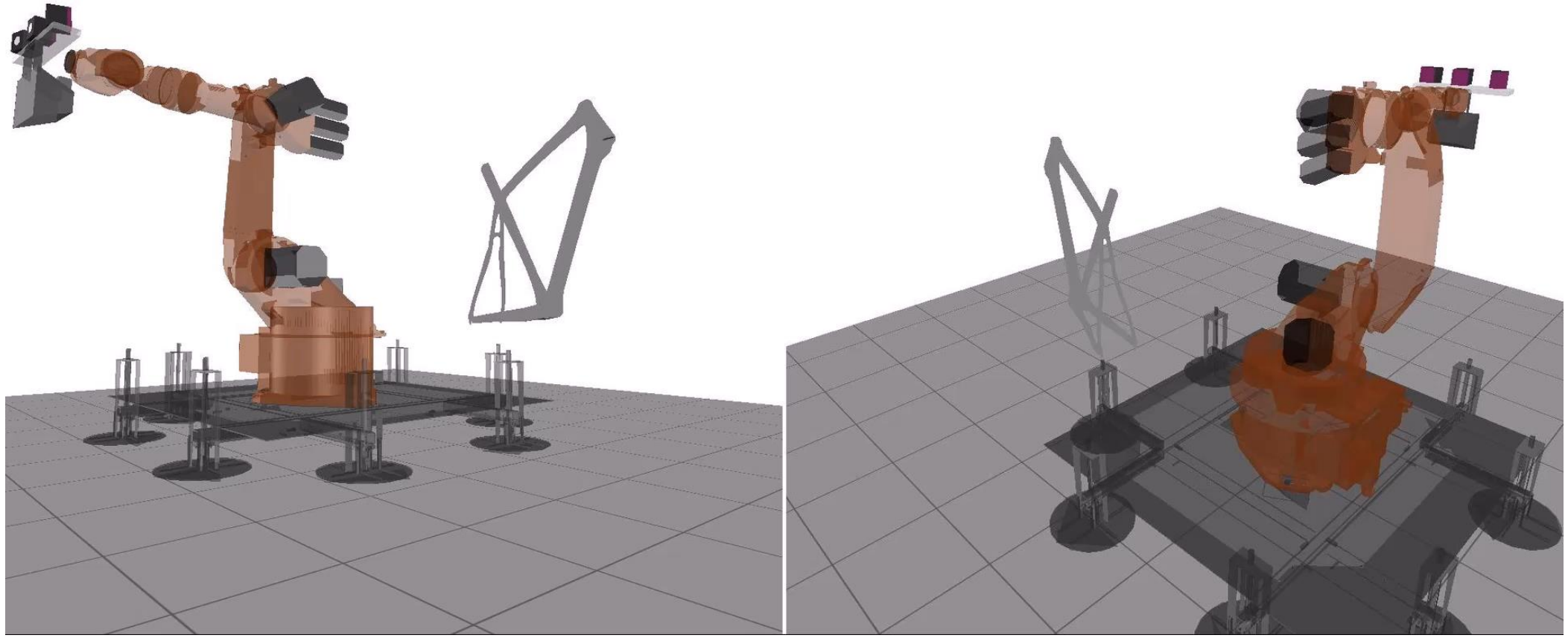
Overview

- Application
- Machinery
- Theory
- Examples

Overview

- Application
- Machinery
- Theory
- Examples

Practical problem



Preliminary conclusion

- Submodular orienteering problem (coverage function)
- Multilinear extention can be estimated as an integral of the pointwise product of two functions
- Problem is factorized in a function dependent on the constraints, and a function dependent on the problem : $\bar{F}_x = \int x(v)w(v)dv$
- Functions are complex

Overview

- Application
- Machinery
- Theory
- Examples

General optimization idea

We are interested in following problem

$$\int g(x).h(x)dx$$

But we do not have access to these possibly complex functions, so we will learn them

$$P\left(f_g^*|x, y_g, x^*\right) \; and \; P(f_h^*|x, y_h, x^*)$$

We also learn the dependence on the parameters which we want to optimize

$$P\left(f_g^*|[x, \theta_g], y_g, [x, \theta_g]^*\right) \; and \; P(f_h^*|[x, \theta_h], y_h, [x, \theta_h]^*)$$

General optimization idea

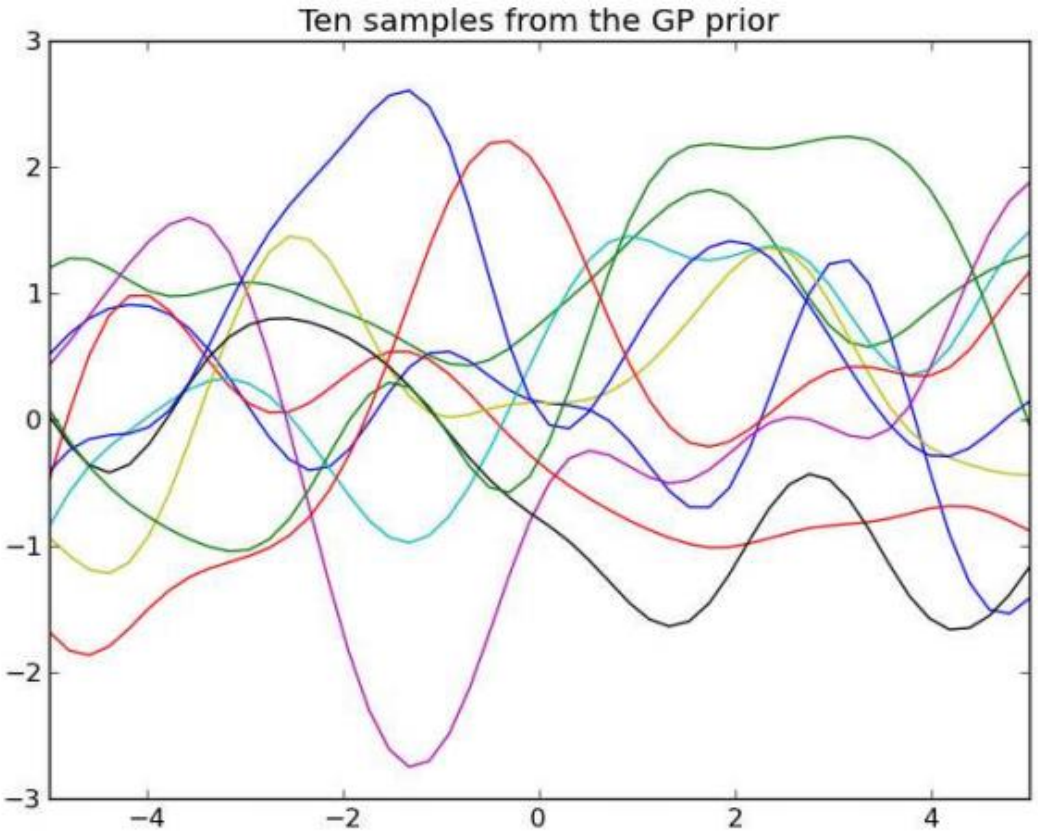
The final optimization problem becomes

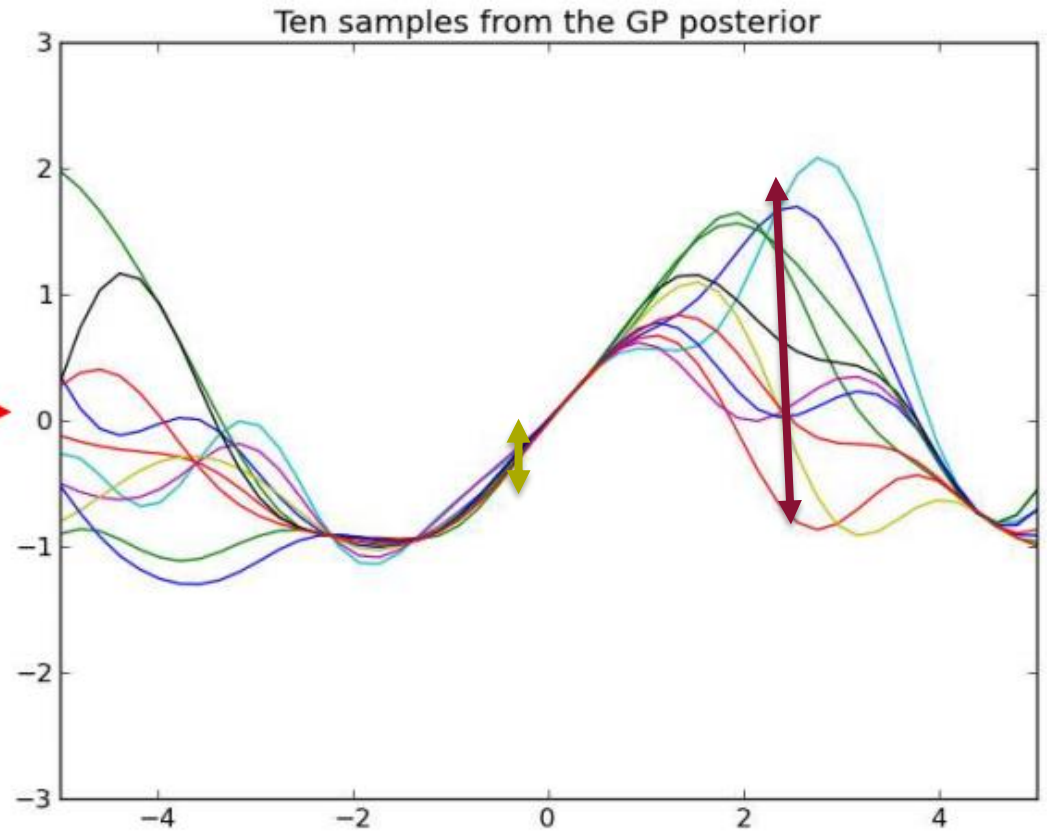
$$\underset{\theta_g,\theta_h}{\operatorname{argmax}} E\left[\int f_g^*(x) f_h^*(x) dx\right]$$

Our optimization strategy:

- Learn both distributions in advance (Gaussian Processes)
- In the optimization loop use optimized quadrature points to evaluate integral.
- Effectively sample the parameter space using Bayesian Optimisation

Gaussian Process





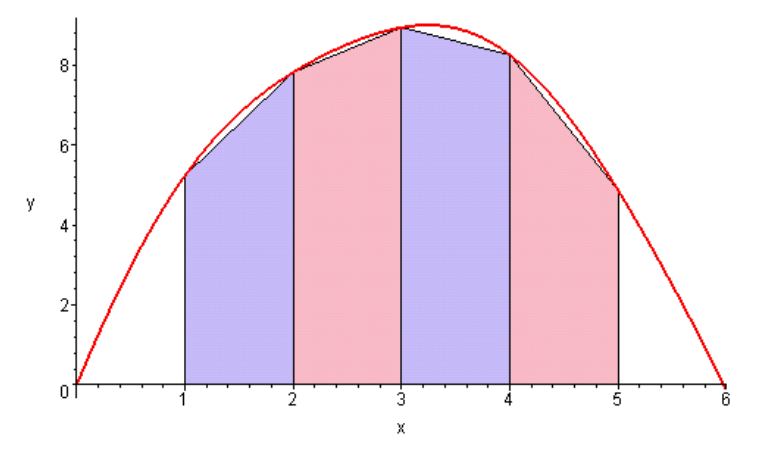
Gaussian distribution over RKHS

Analytic exact expression for posterior after point measurements

Bayesian quadrature

Integration without close form expression

Trapezoidal rule

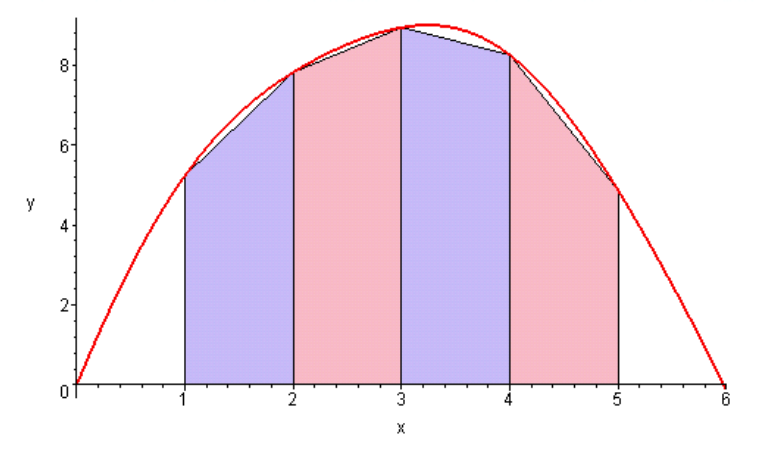


$$\int f(x)dx \approx \sum_{i=1}^{n} \frac{\Delta x}{2}\left(f(x_{i-1}) + f(x_i)\right)$$

Bayesian quadrature

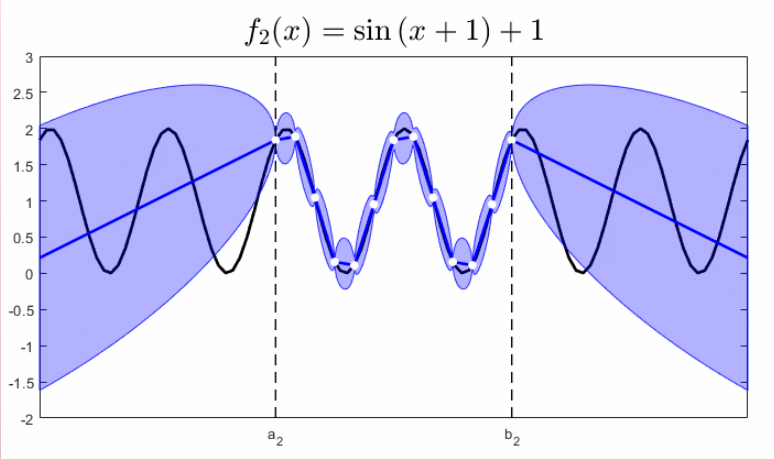
Integration without close form expression

Trapezoidal rule



$$\int f(x)dx \approx \sum_{i=1}^{n} w_i\left(f(x_{i-1}) + f(x_i)\right)$$

Bayesian quadrature

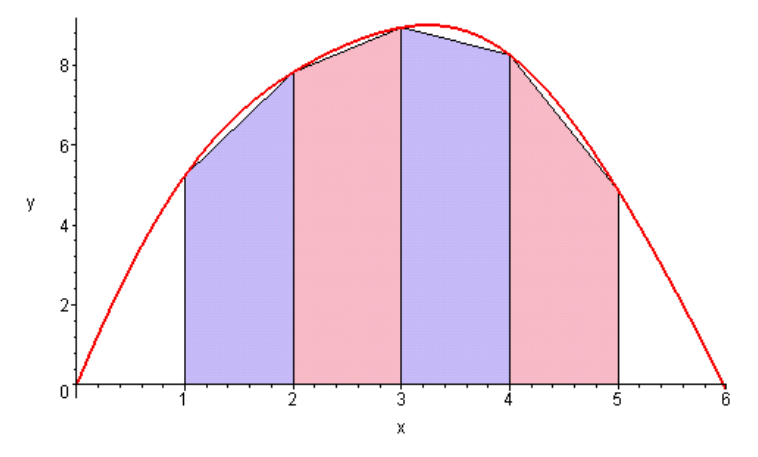


$$\int f(x)dx \approx \boldsymbol{y}\boldsymbol{K}^{-1}\int K(X,x)dx$$

Bayesian quadrature

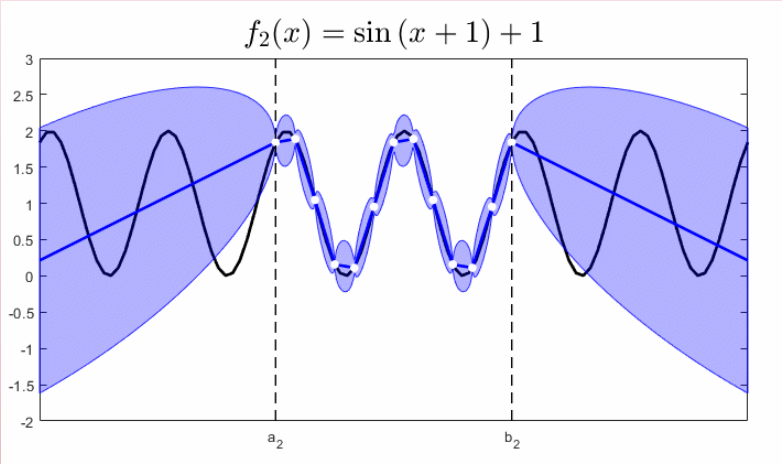
Integration without close form expressio

Trapezoidal rule



$$\int f(x)dx \approx \sum_{i=1}^{n} w_i\left(f(x_{i-1}) + f(x_i)\right)$$

Bayesian quadrature



$$\int f(x)dx \approx \boldsymbol{y}\,\boldsymbol{K}^{-1}\int K(X,x)dx$$

$$[f(x_1) \quad \cdots \quad f(x_n)]\begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$$

Product of Gaussian Processes

Let h$\sim GP\left(m_h(x), C_h(x,x')\right)$ and $g \sim GP\left(m_g, C_g(x,x')\right)$ then we are interested the distribution over $f.g$ given by its mean:

$$m_{gh}(x) = E[h(x)]E[g(x)] = m_h(x).m_g(x)$$

And covariance

$$C_{gh}(x,x') = C\left(f(x).g(x), f(x')g(x')\right)$$

The covariance of products of Gaussian random variables is analytically available (Bohrnstedt et al [7]) and for GP's given by:

$$C_{gh}(x,x') = m_h(x)C_g(x,x')m_h(x') + m_g(x)C_h(x,x')m_g(x') + C_h(x,x')C_g(x,x')$$

We choose to conveniently neglect other moments

Product quadrature

Further derivations are tedious!!! But three types of integrals need to be computed:

$$\int \mathcal{K}_h\left(X_{h_i}, x\right)\mathcal{K}_g\left(X_{g_j}, x\right)dx$$

$$\int\int \mathcal{K}_h(x, x')\mathcal{K}_g(x, x')dx'dx$$

$$\int \mathcal{K}_h\left(x, X_{h_i}\right)\int \mathcal{K}_h\left(X_{h_j}, x'\right)\mathcal{K}_g(x, x')dx'dx$$

Computational cost
Mean : $O(MN)$
Variance : $O(M^2N)$

Where X_{h_i} and X_{h_j} are training points

A non-exhaustive list of possible kernel combinations is given by Briol et al [3]

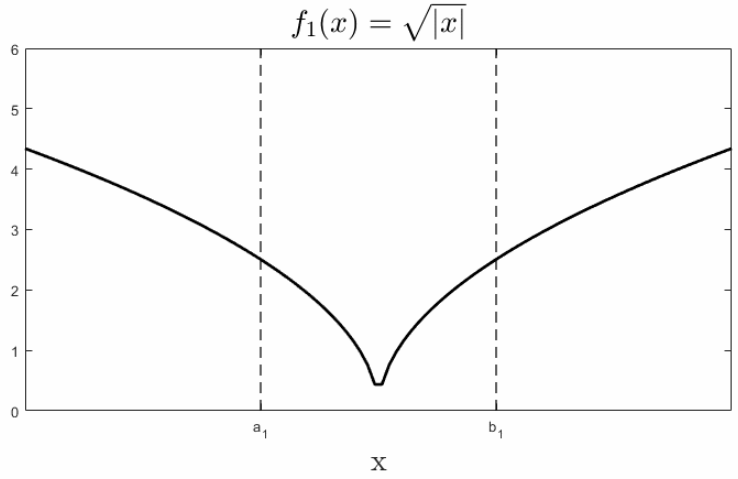
Our choice of kernel

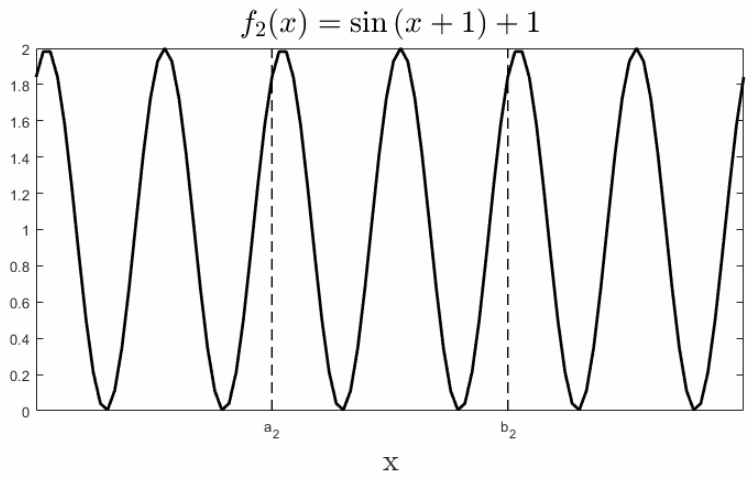
Product of one dimensional piecewise polynomials (1st order) with compact support

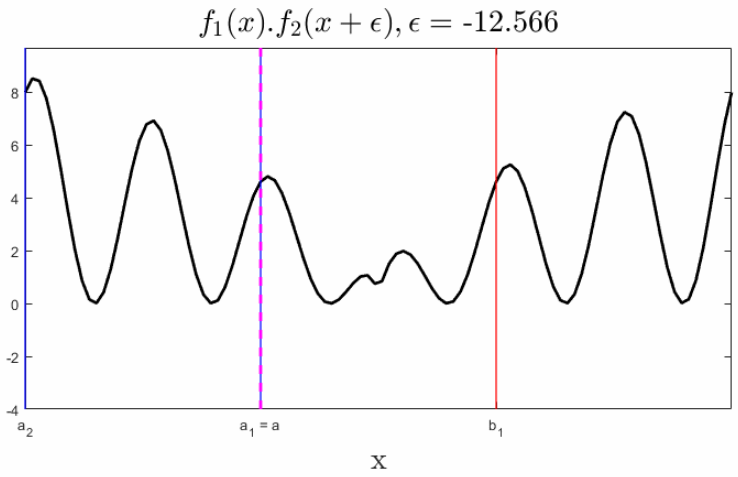
$$\mathcal{K}(x, x') = \prod_{i=1}^{D} \max\left(\theta_{1,i}\left(1 - \theta_{2,i}|x - x'|\right), 0\right)$$

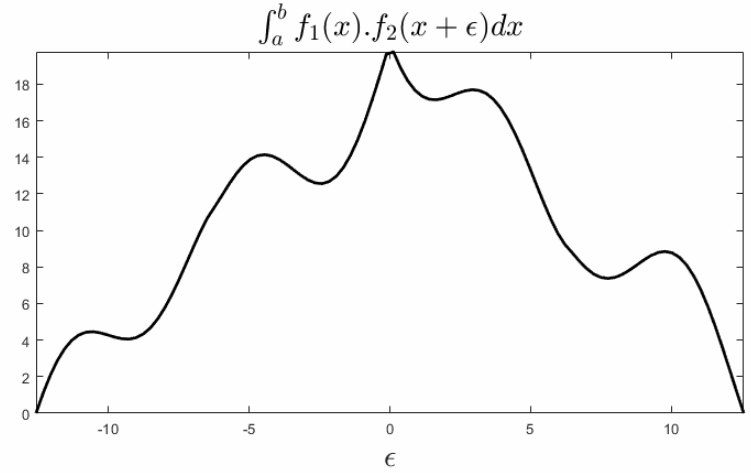
- Brownian motion prior (nonlinearities)
- Compact support
- Sparse

Example

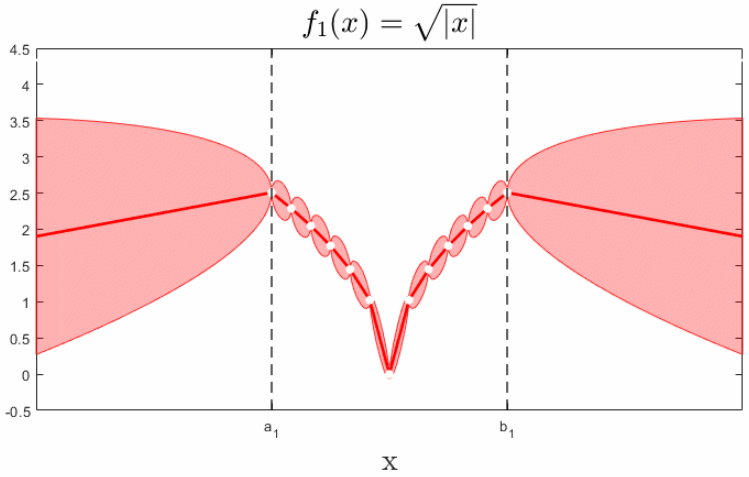

$f_1(x) = \sqrt{|x|}$
6
5
4
3
2
1
0
a1
b1
x

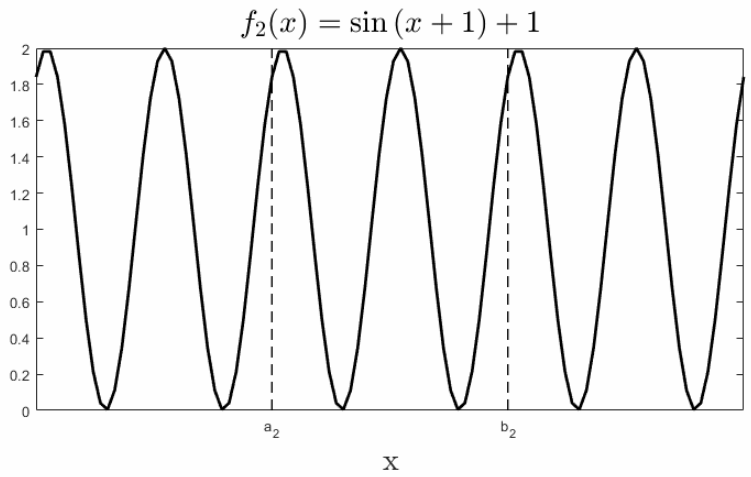

$f_2(x) = \sin(x+1) + 1$
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
a2
b2
x

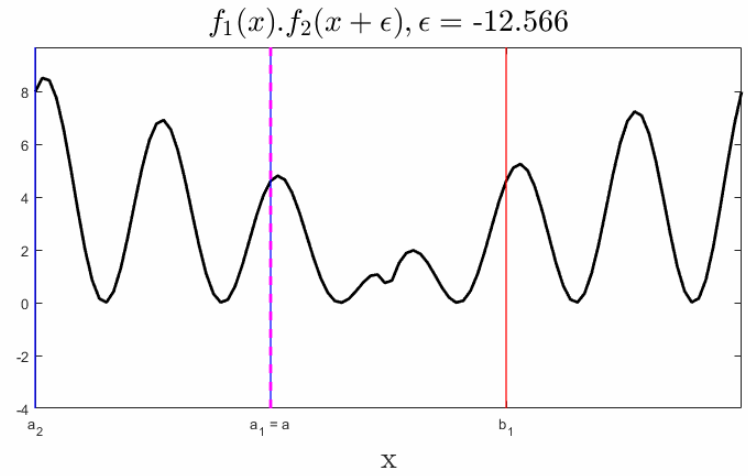

$f_1(x).f_2(x+\epsilon), \epsilon = -12.566$
8
6
4
2
0
-2
-4
a2
a1 = a
b1
x

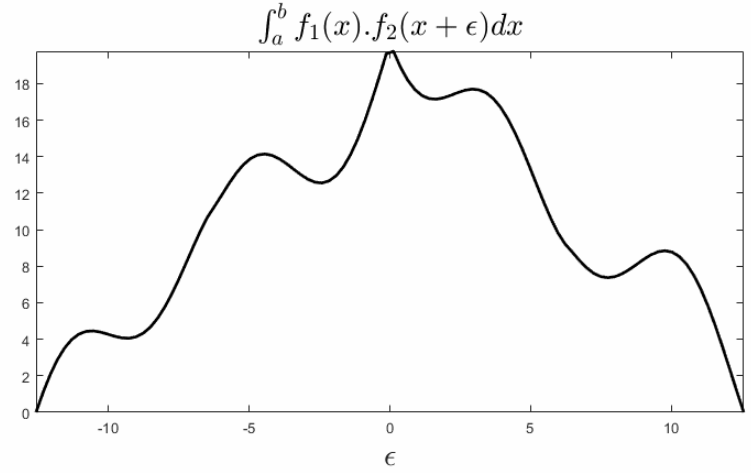

$\int_a^b f_1(x).f_2(x+\epsilon)dx$
18
16
14
12
10
8
6
4
2
0
-10
-5
0
5
10
ϵ

Example

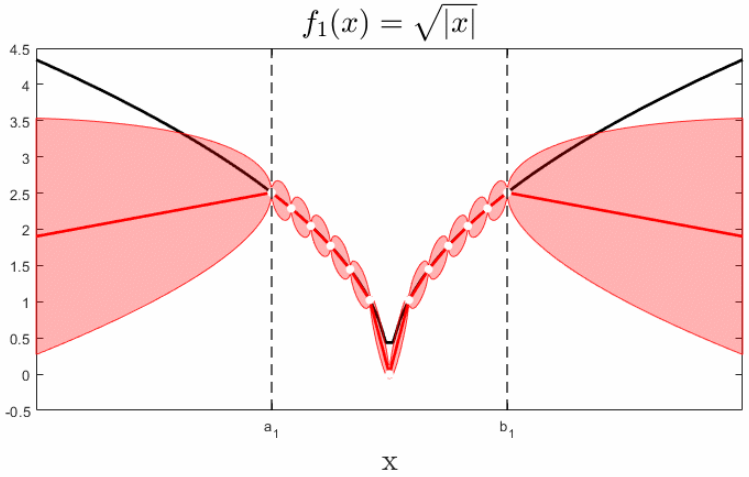

$f_1(x) = \sqrt{|x|}$
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
a_1
b_1
x

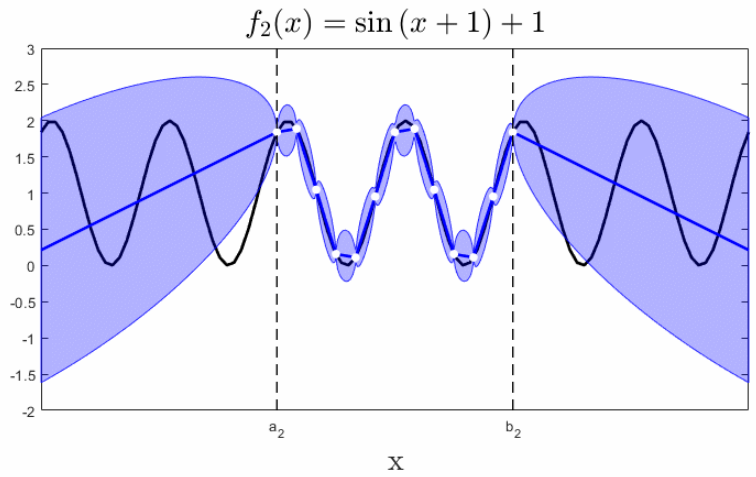

$f_2(x) = \sin(x+1) + 1$
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
a_2
b_2
x

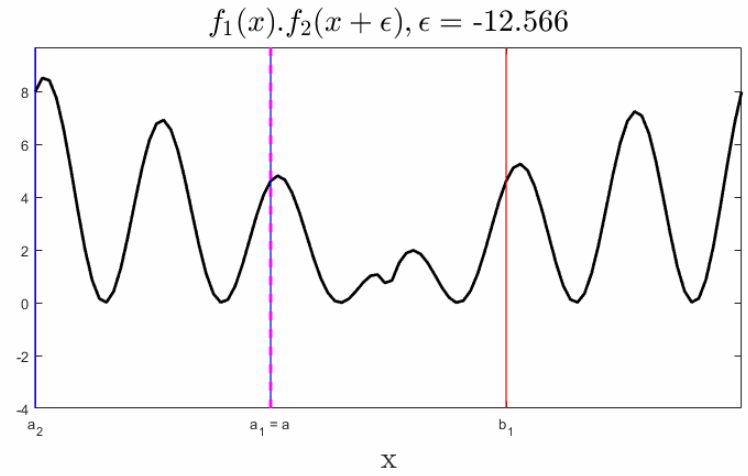

$f_1(x).f_2(x+\epsilon), \epsilon = -12.566$
8
6
4
2
0
-2
-4
a_2
$a_1 = a$
b_1
x

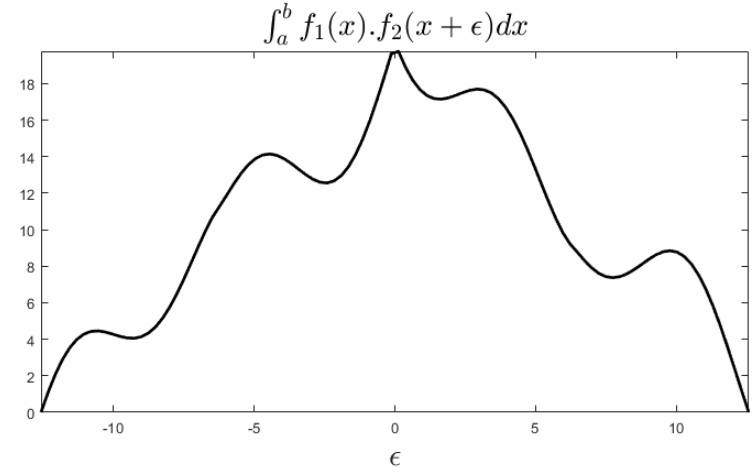

$\int_a^b f_1(x).f_2(x+\epsilon)dx$
18
16
14
12
10
8
6
4
2
0
-10
-5
0
5
10
ϵ

Example

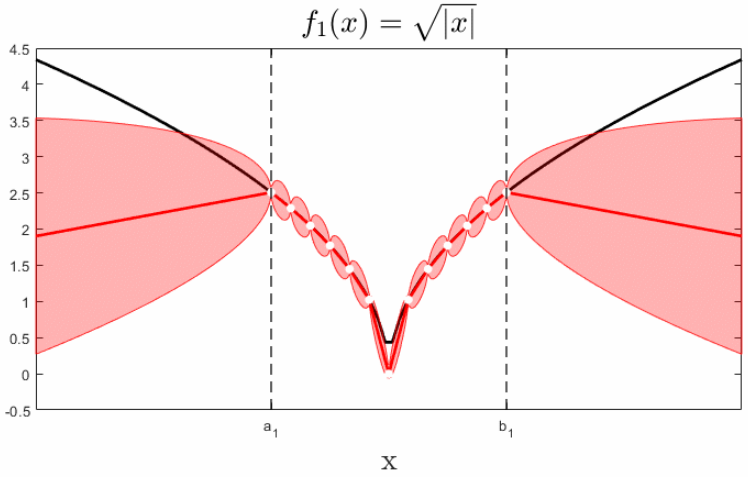

$f_1(x) = \sqrt{|x|}$
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
a_1
b_1
x

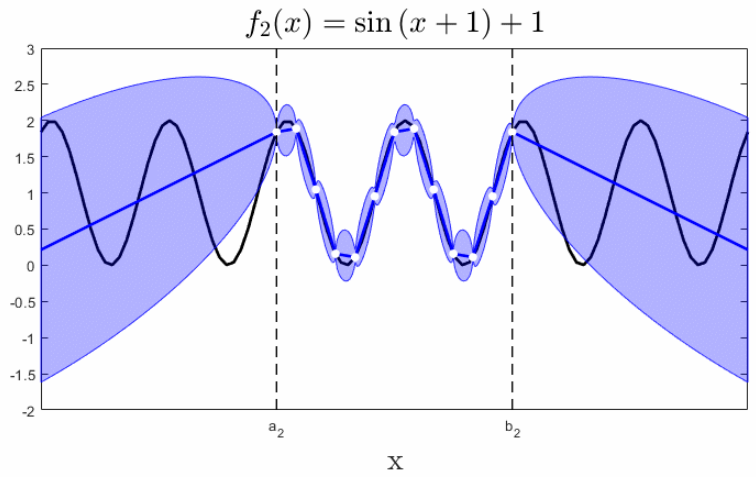

$f_2(x) = \sin(x+1) + 1$
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
a_2
b_2
x

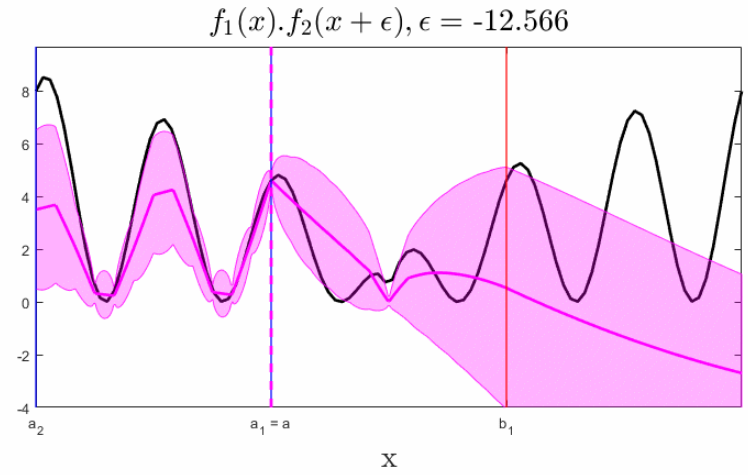

$f_1(x).f_2(x+\epsilon), \epsilon = -12.566$
8
6
4
2
0
-2
-4
a_2
$a_1 = a$
b_1
x

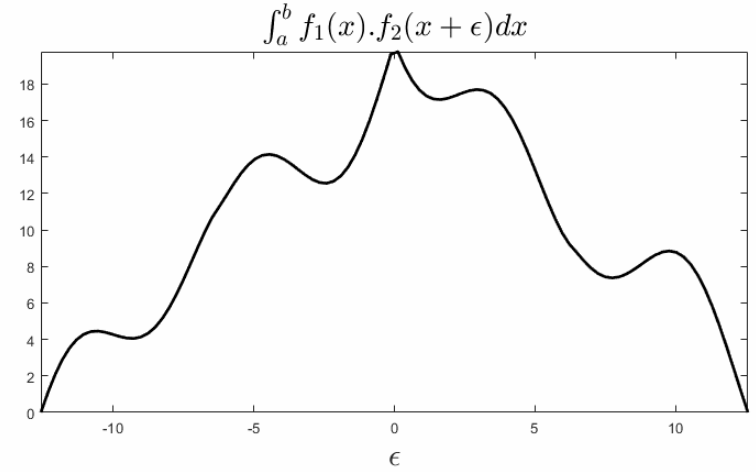

$\int_a^b f_1(x).f_2(x+\epsilon)dx$
18
16
14
12
10
8
6
4
2
0
-10
-5
0
5
10
ϵ

Example


$f_1(x) = \sqrt{|x|}$
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
a_1
b_1
x


$f_2(x) = \sin(x+1) + 1$
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
a_2
b_2
x


$f_1(x).f_2(x+\epsilon), \epsilon = -12.566$
8
6
4
2
0
-2
-4
a_2
$a_1 = a$
b_1
x


$\int_a^b f_1(x).f_2(x+\epsilon)dx$
18
16
14
12
10
8
6
4
2
0
-10
-5
0
5
10
ϵ

Note on difference with BQ

- Bayesian quadrature

Choose next quadrature point to minimize variance on the estimated integral

- Our setting

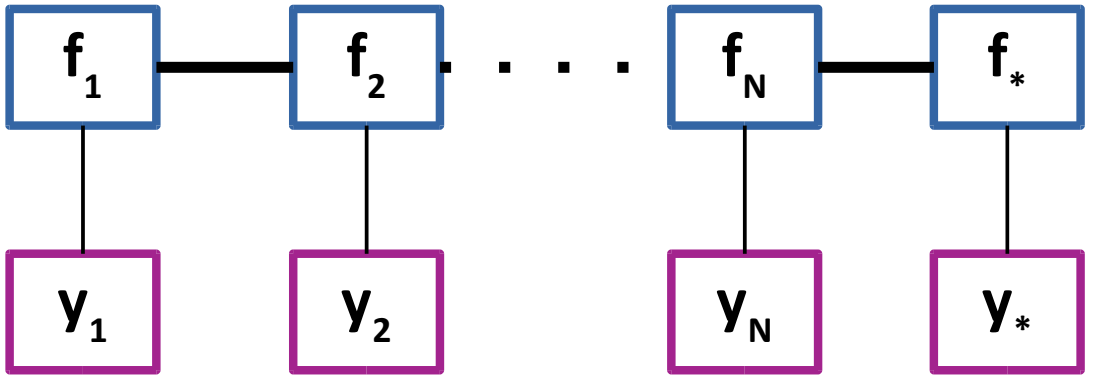
Remove n quadrature points that minimize the variance increase on the estimated integral

Gaussian process pseudo-inputs

We use the fully independent training conditional approximation by Snelson and Ghahramani (2006) (**FITC**, using classification of Quiñonero-Candela et al(2005))

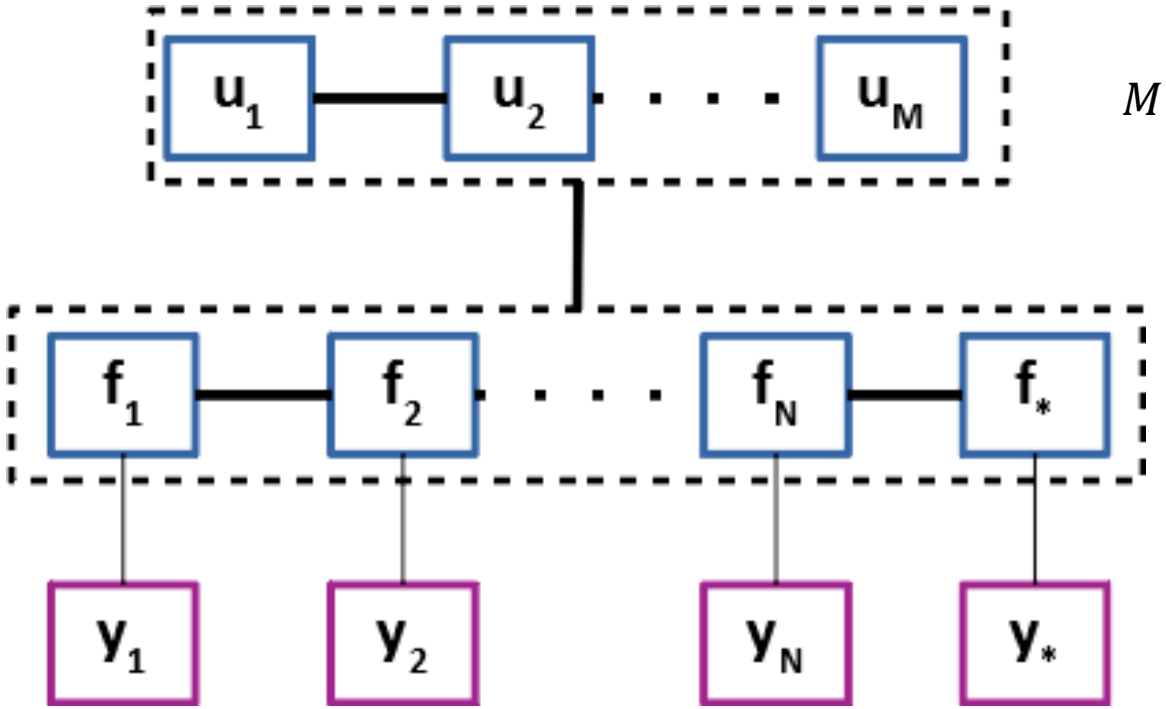
Gaussian process pseudo-inputs

We use the fully independent training conditional approximation by Snelson and Ghahramani (2006) (**FITC**, using classification of Quiñonero-Candela et al(2005))



Gaussian process pseudo-inputs

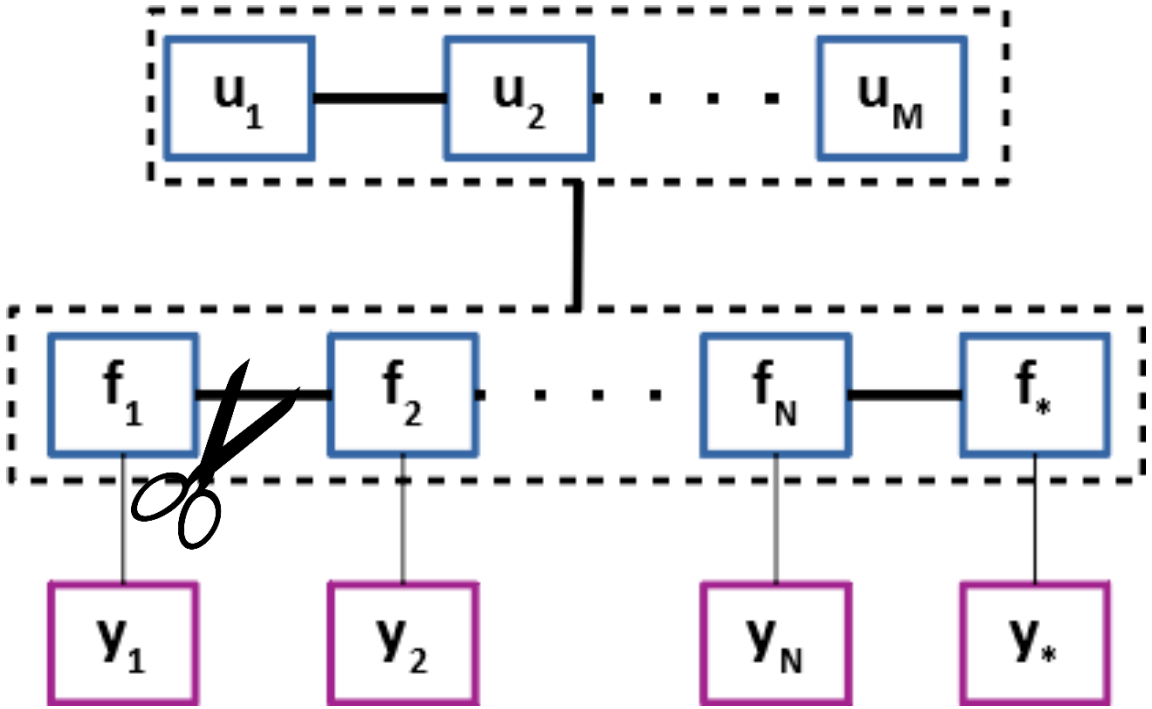
We use the fully independent training conditional approximation by Snelson and Ghahramani (2006) (**FITC**, using classification of Quiñonero-Candela et al(2005))



$M << N$

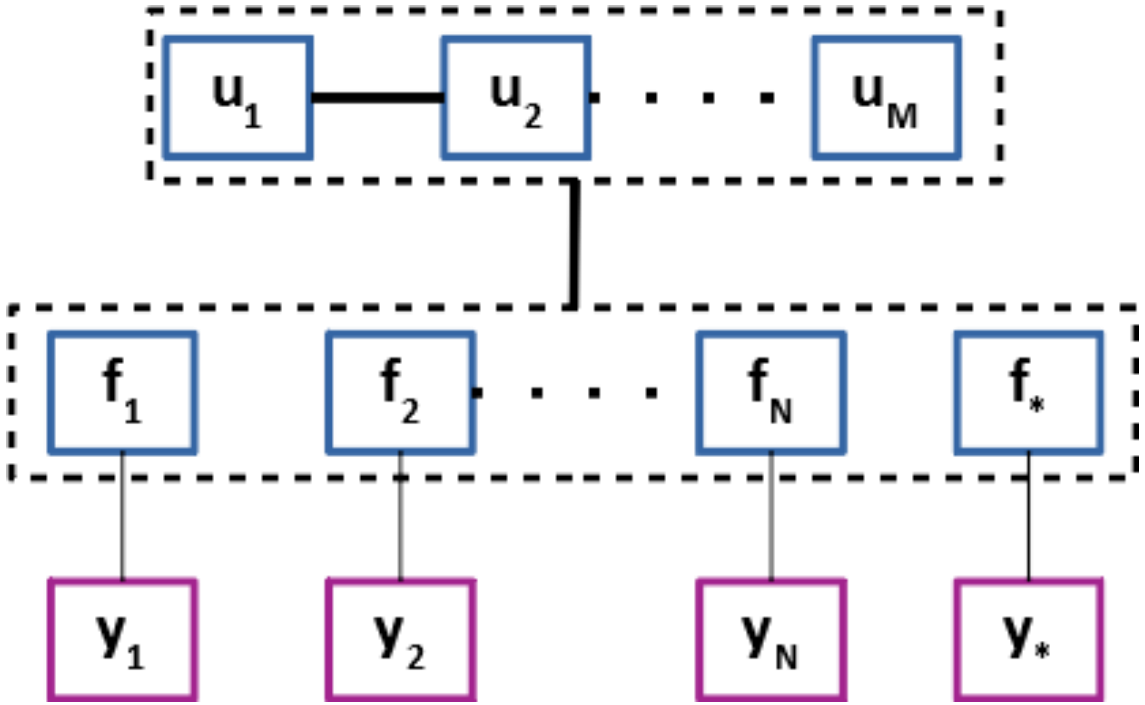
Gaussian process pseudo-inputs

We use the fully independent training conditional approximation by Snelson an Ghahramani (2006) (**FITC**, using classification of Quiñonero-Candela et al(2005))



Gaussian process pseudo-inputs

We use the fully independent training conditional approximation by Snelson and Ghahramani (2006) (**FITC**, using classification of Quiñonero-Candela et al(2005))



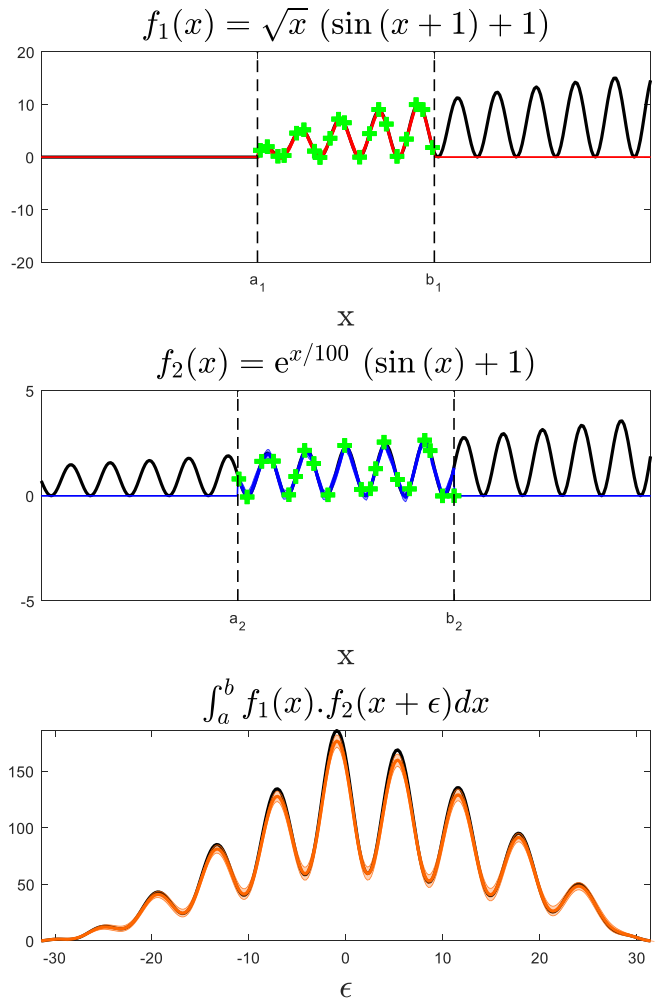
Gaussian process pseudo-inputs

- Positions of pseudo input locations u are optimized to maximize the marginal likelihood (details in the original work by Snelson an Ghahramani (2006))
- Inference is possible by marginalizing out pseudo inputs u
- Exact test conditional (Quiñonero-candela, Rasmussen, & Herbrich, 2005)

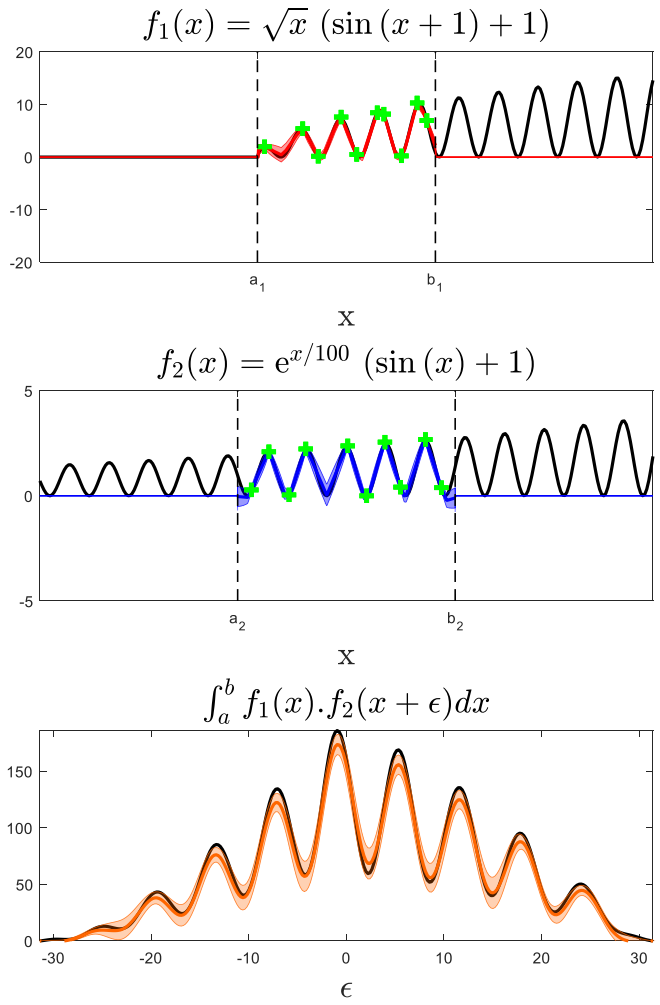
$$p(f_*|\boldsymbol{u}) = \mathcal{N}(K_{*u}K_{uu}^{-1}\boldsymbol{u},\, K_{**} - K_{*u}K_{uu}^{-1}K_{u*})$$

Example: varying number of pseudo inputs

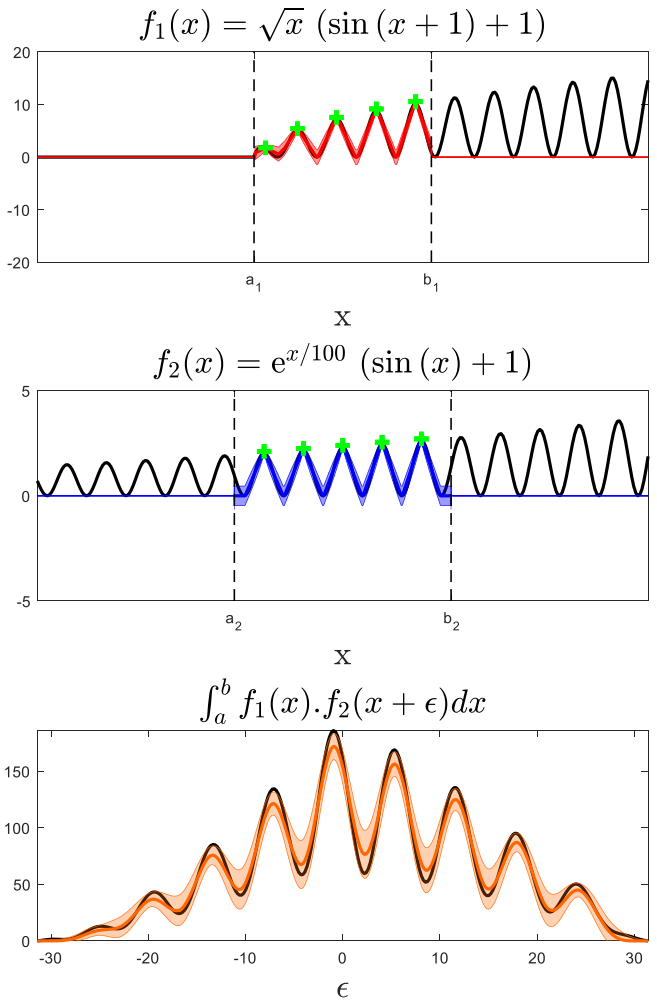
20 pseudo-inputs



10 pseudo-inputs

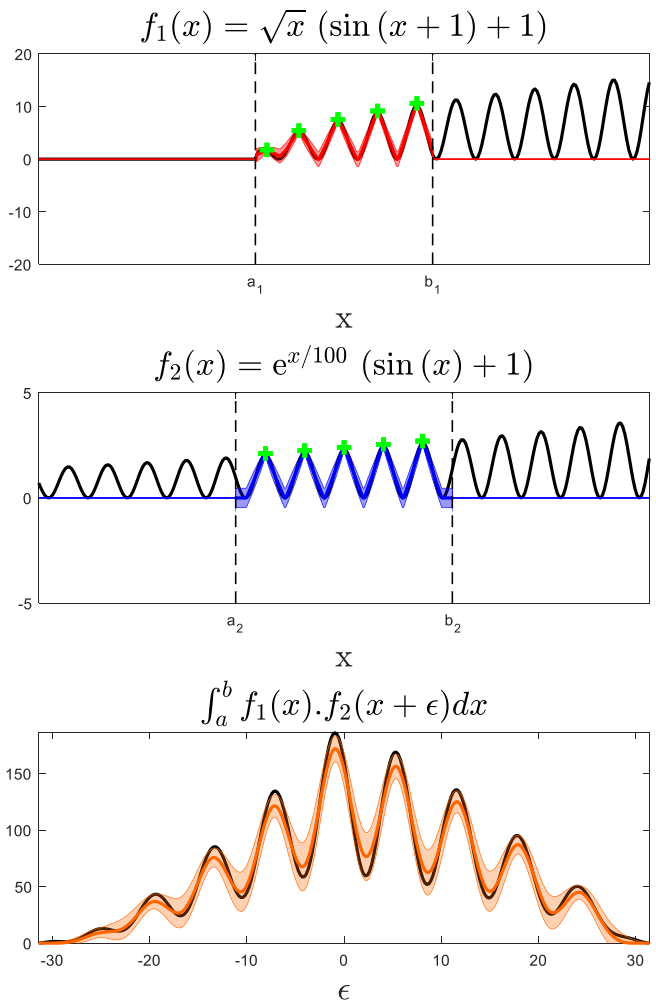


5 pseudo-inputs

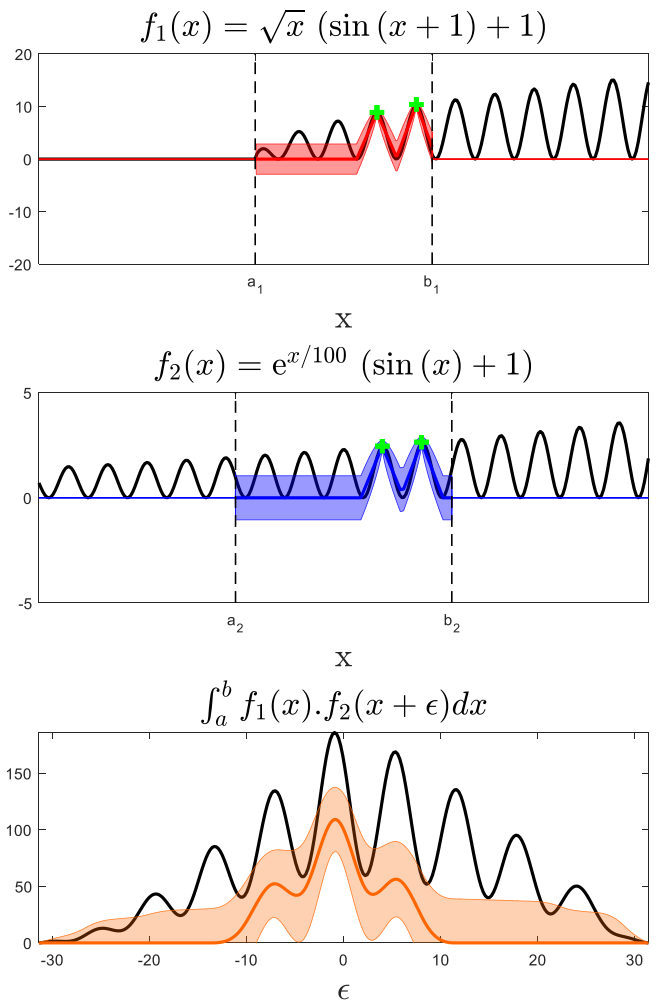


Example: varying number of pseudo inputs

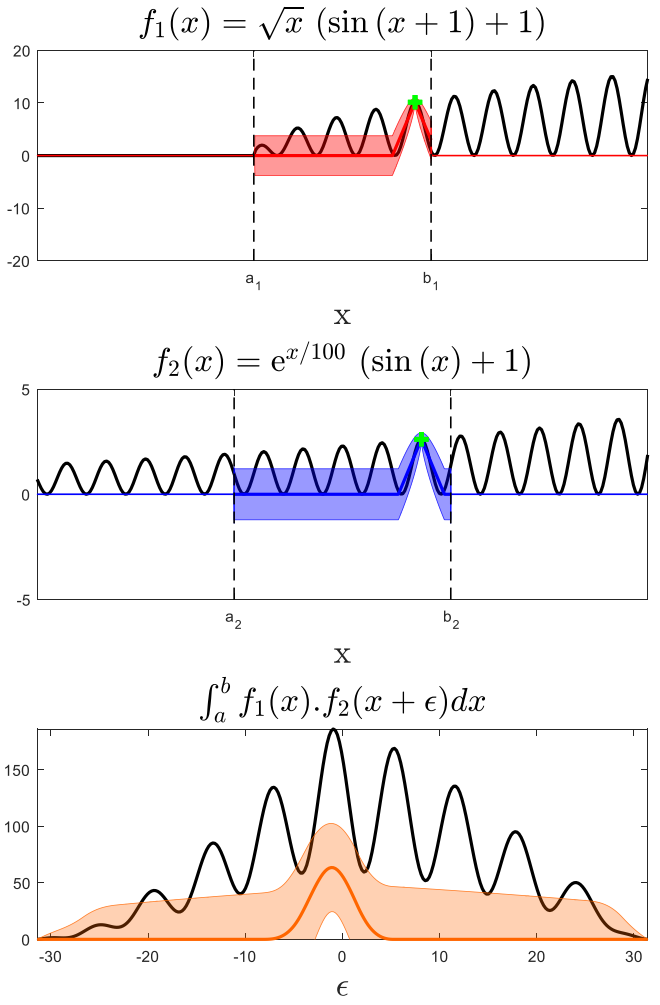
5 pseudo-inputs



2 pseudo-inputs



1 pseudo-input !!!



Optimization

The goal is not to evaluate every point. We consider following optimization problem:

$$\underset{\Delta \in \mathbb{R}^n}{\operatorname{argmax}} E\left[\int P\left(f_g^* | x, y_g, x^*\right) . P\left(f_h^* | x + \Delta, y_h, x^*\right) dx\right]$$

Strategy (Bayesian optimization):

- Evaluate functions at points Δ^*
- Put Gaussian process prior on all $\Delta \in \mathbb{R}^n$
- Select new location with highest expected improvement/…

Overview

- Application
- Machinery
- Theory
- Examples

Numerical estimation vs optimization

$$\underset{\Delta \in \mathbb{R}^n}{\operatorname{argmax}} E\left[\int P\left(f_g^* | x, y_g, x^*\right).P(f_h^* | x + \Delta, y_h, x^*)\, dx\right]$$

$$g(a, b)$$

Learning g is what we can do

Numerical estimation vs optimization

$$\underset{\Delta \in \mathbb{R}^n}{\operatorname{argmax}} E\left[\int P\left(f_g^* | x, y_g, x^*\right).P(f_h^* | x + \Delta, y_h, x^*)\, dx\right]$$

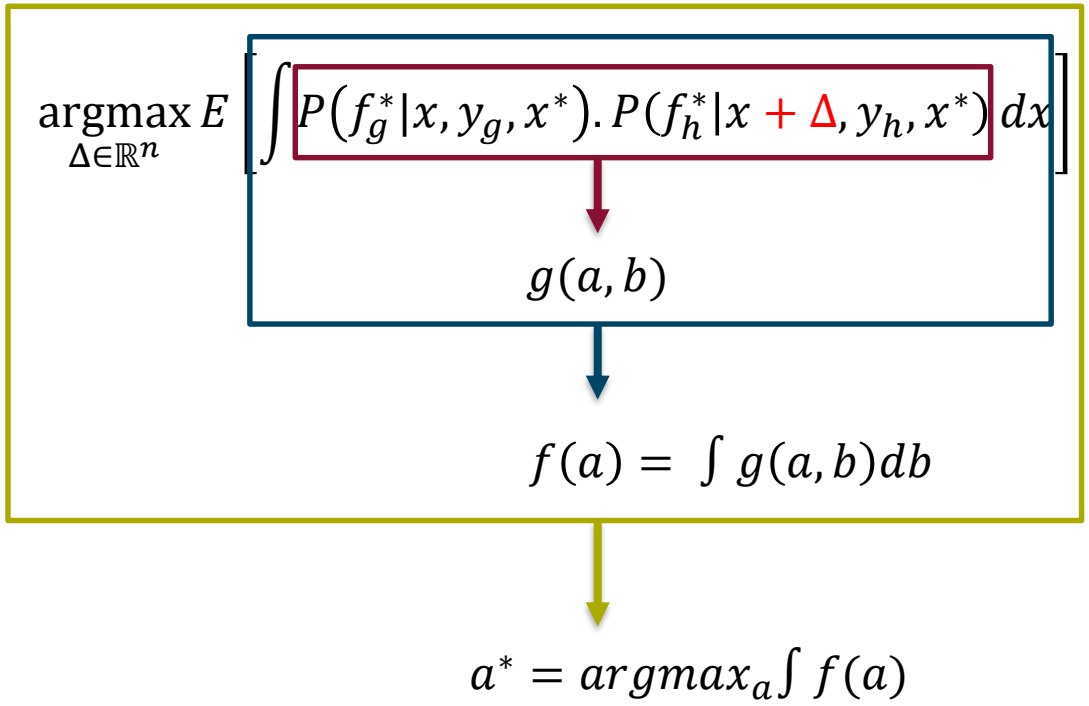
$$g(a, b)$$

$$f(a) = \int g(a, b) db$$

Learning g is what we can do

Sample g to acurately estimate f?

Numerical estimation vs optimization

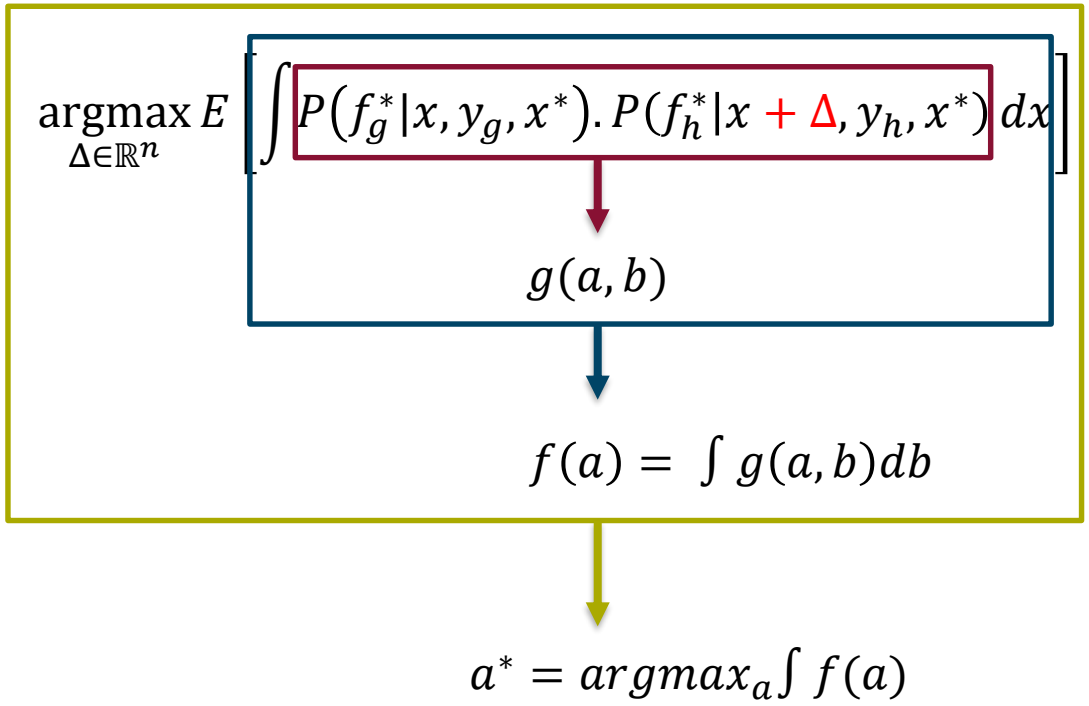


Learning g is what we can do

Sample g to acurately estimate f?

Or optimally find a^*?

Numerical estimation vs optimization



Learning g is what we can do

↕

Sample g to acurately estimate f?

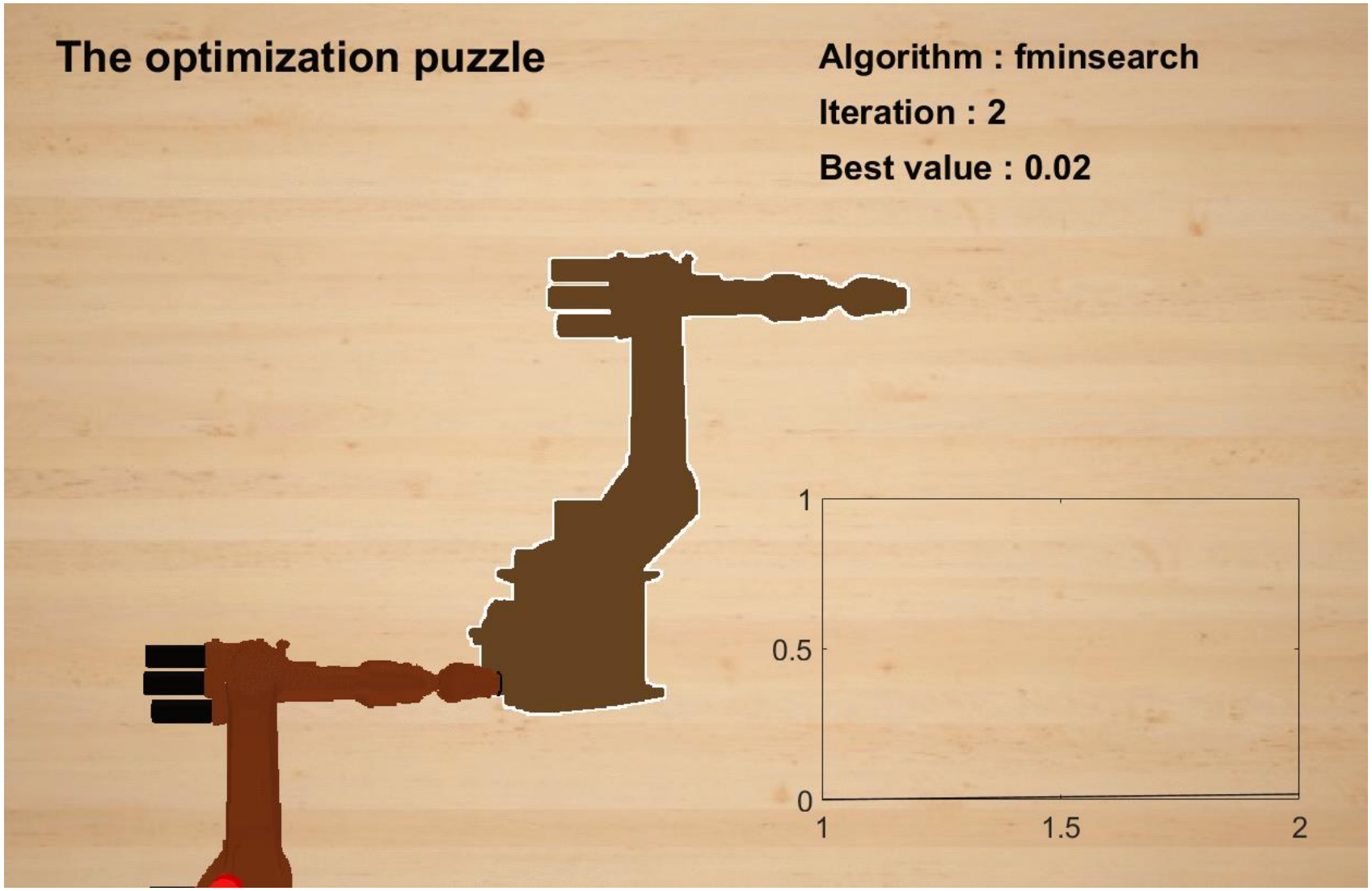
↕

Or optimally find a^*?

Overview

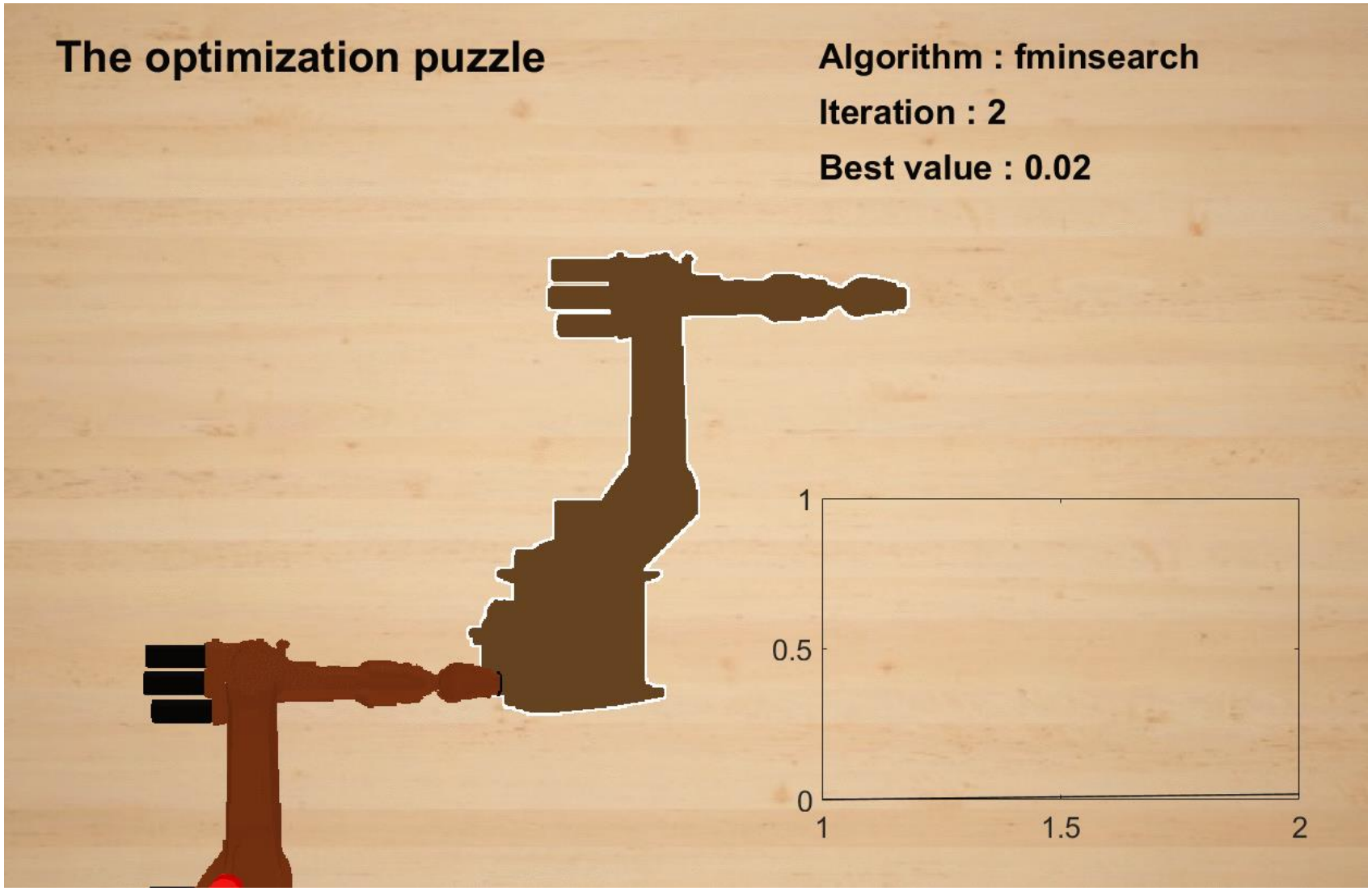
- Application
- Machinery
- Theory
- Examples

The optimization puzzle (1)

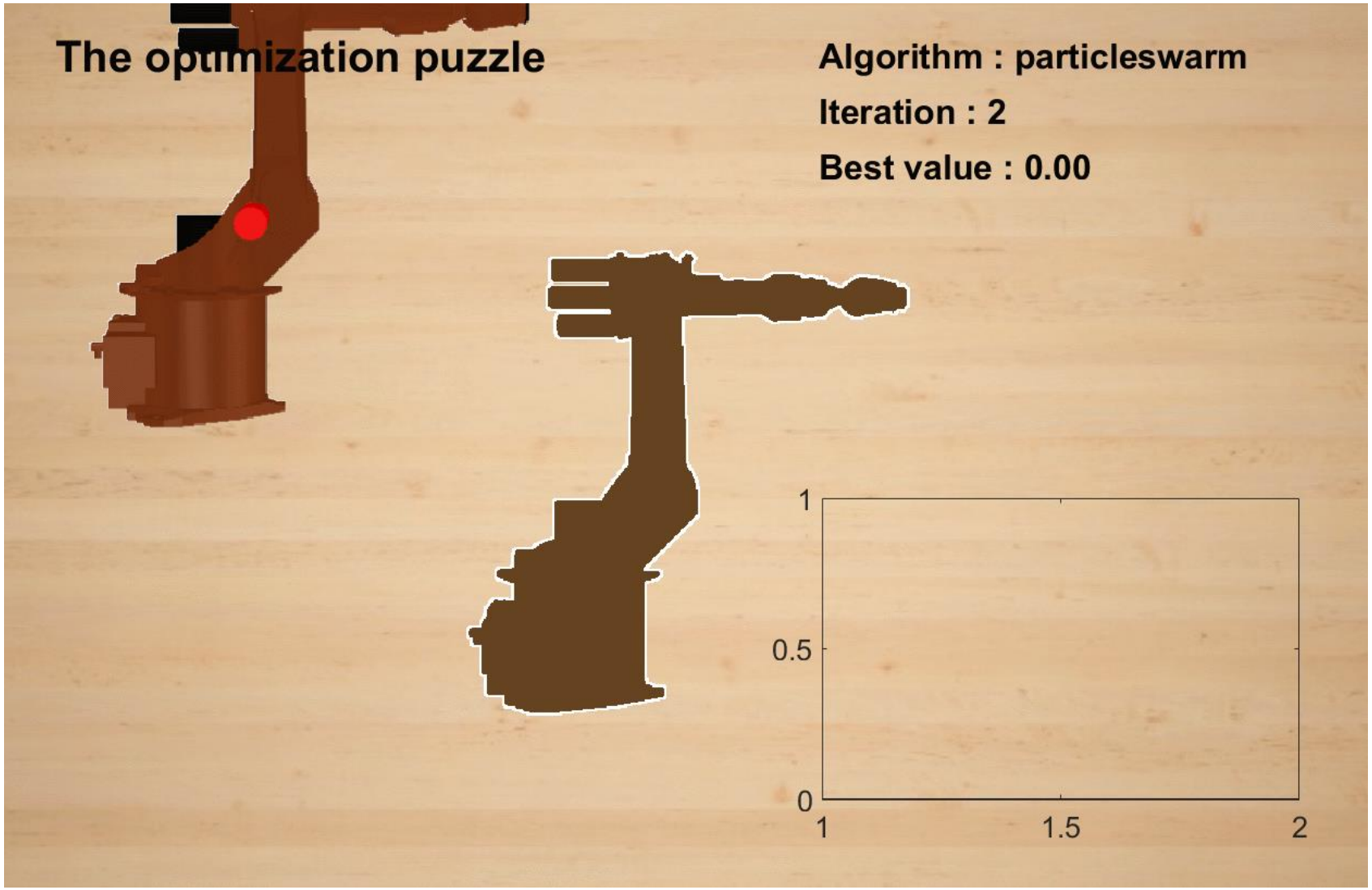


The optimization puzzle (1)

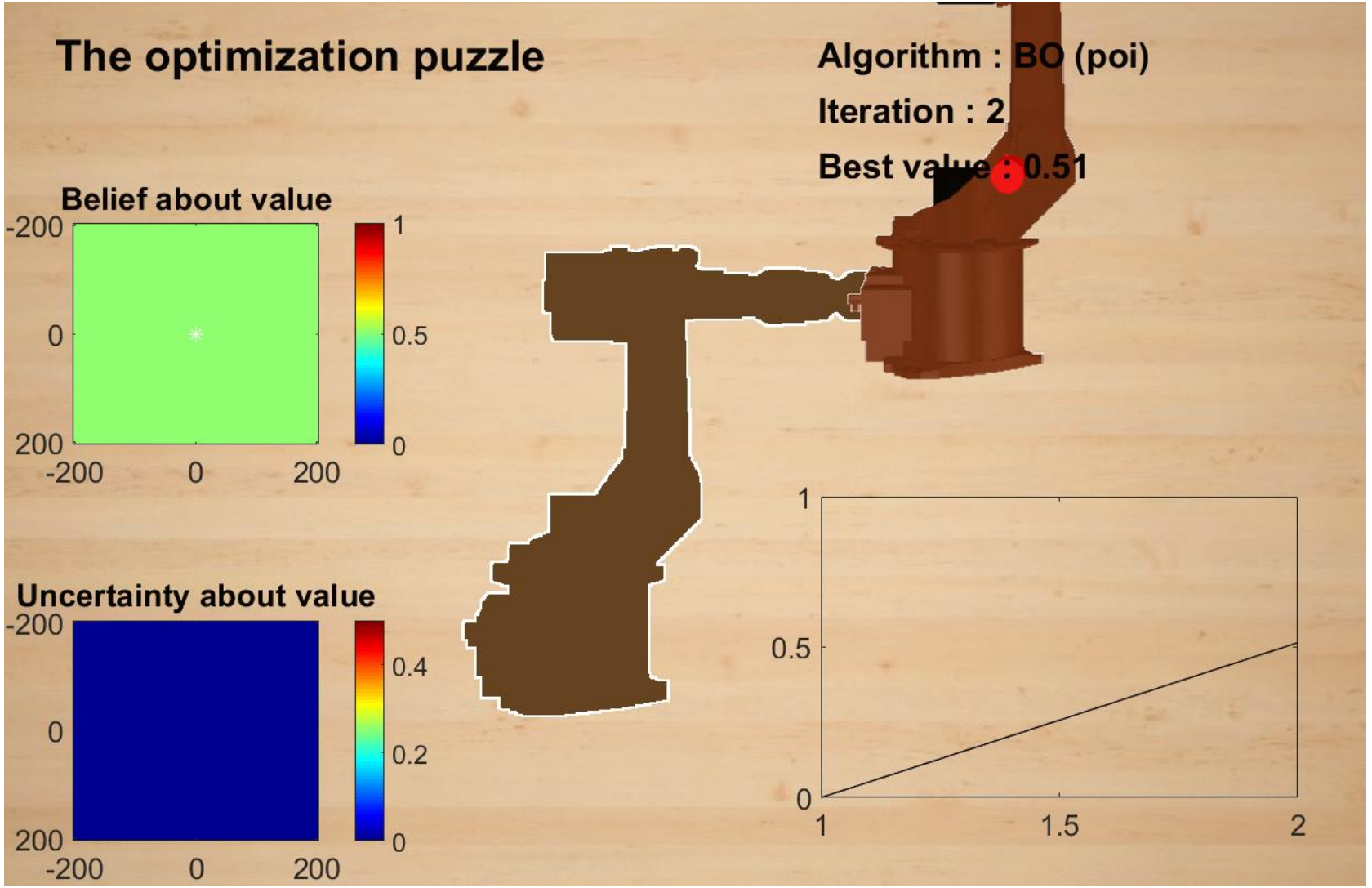
The optimization puzzle (1)



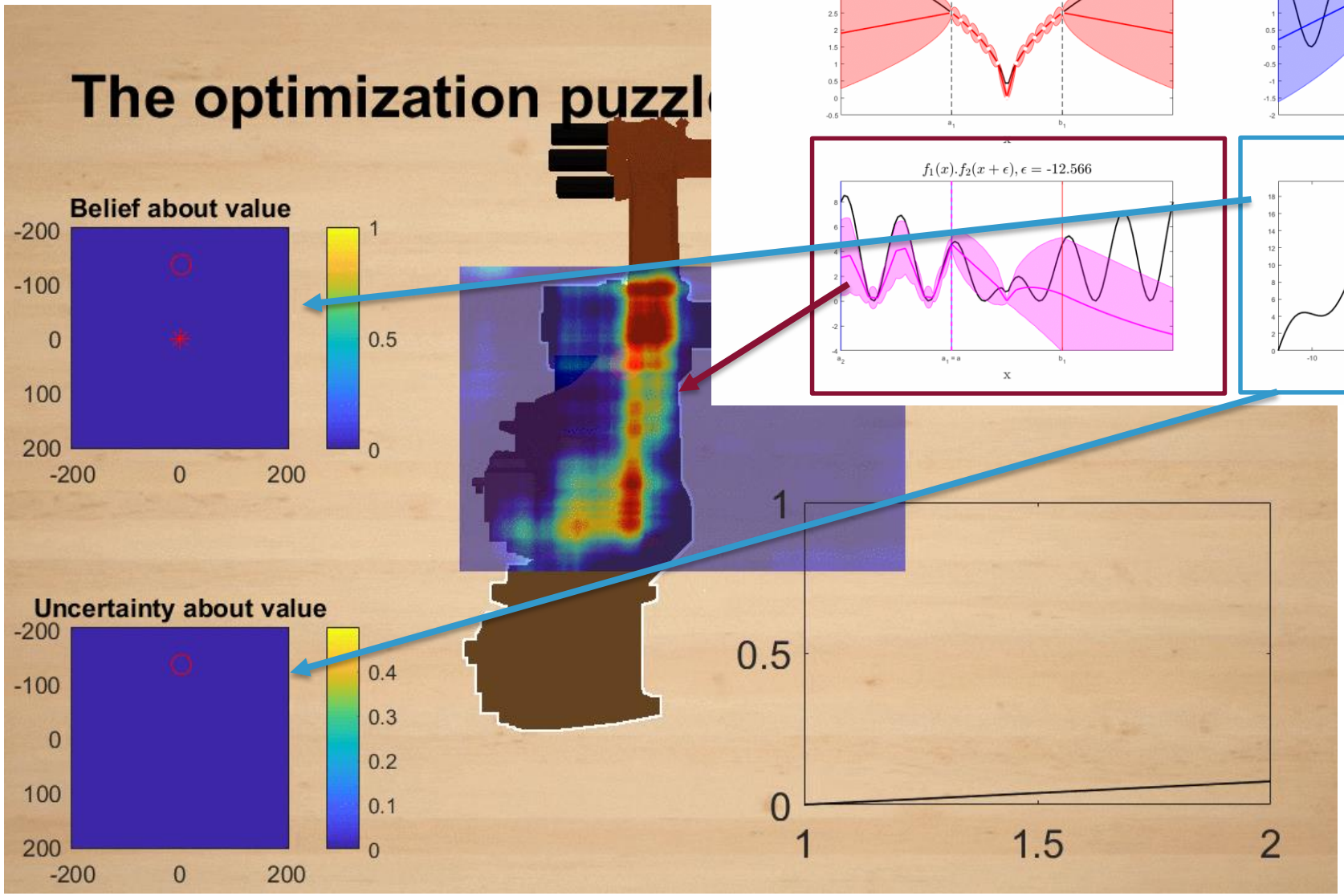
The optimization puzzle (1)

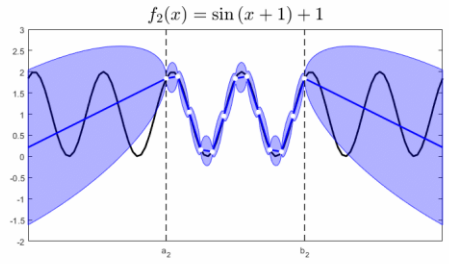


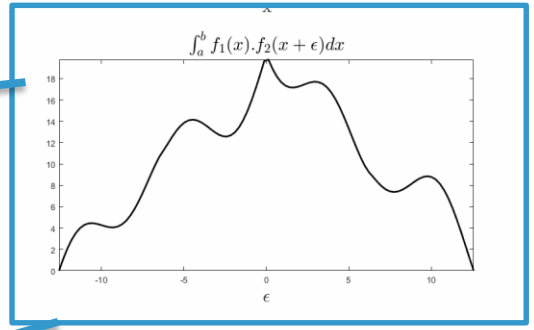
The optimization puzzle (1)



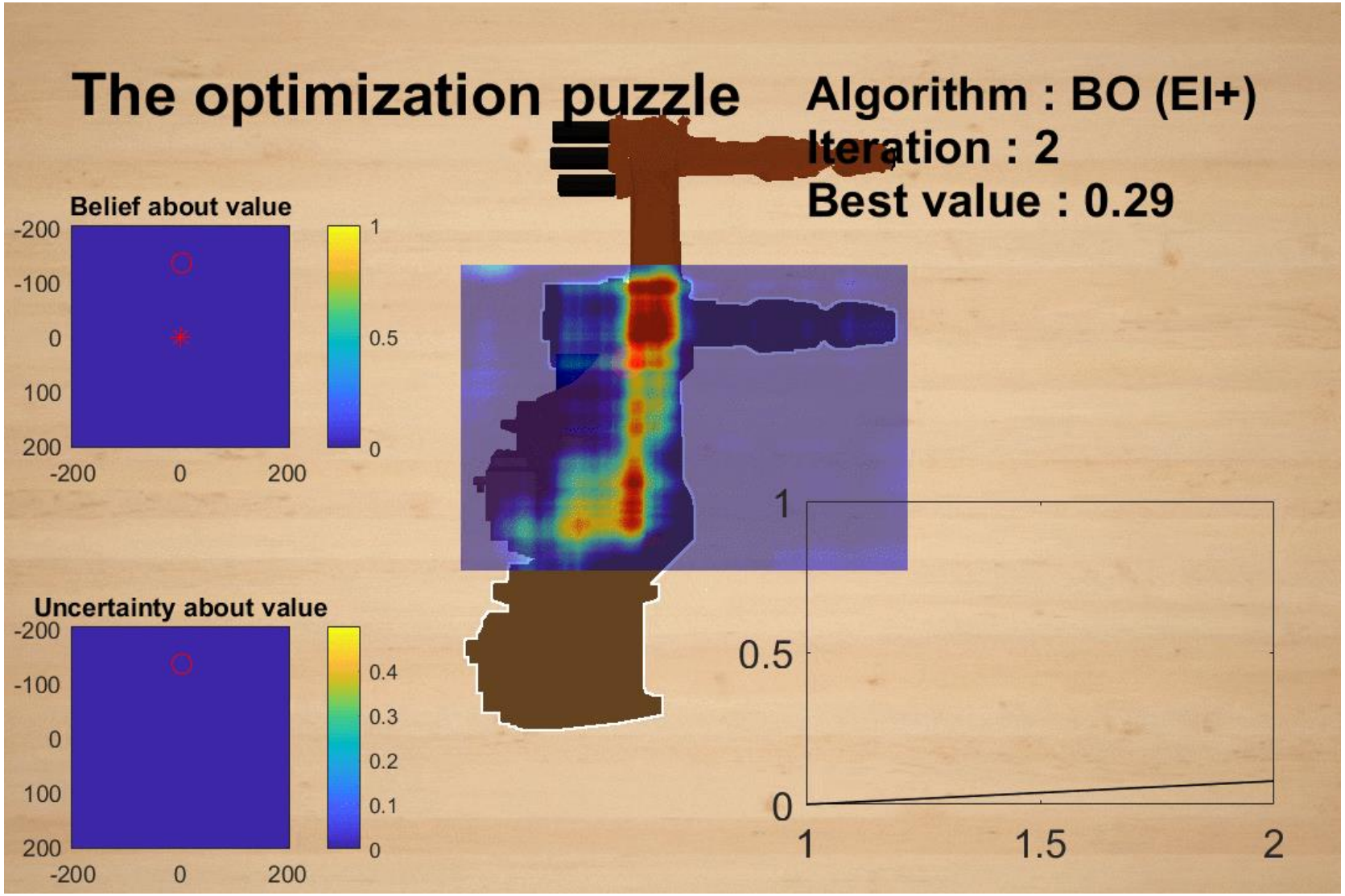
The optimization puzzle (1)



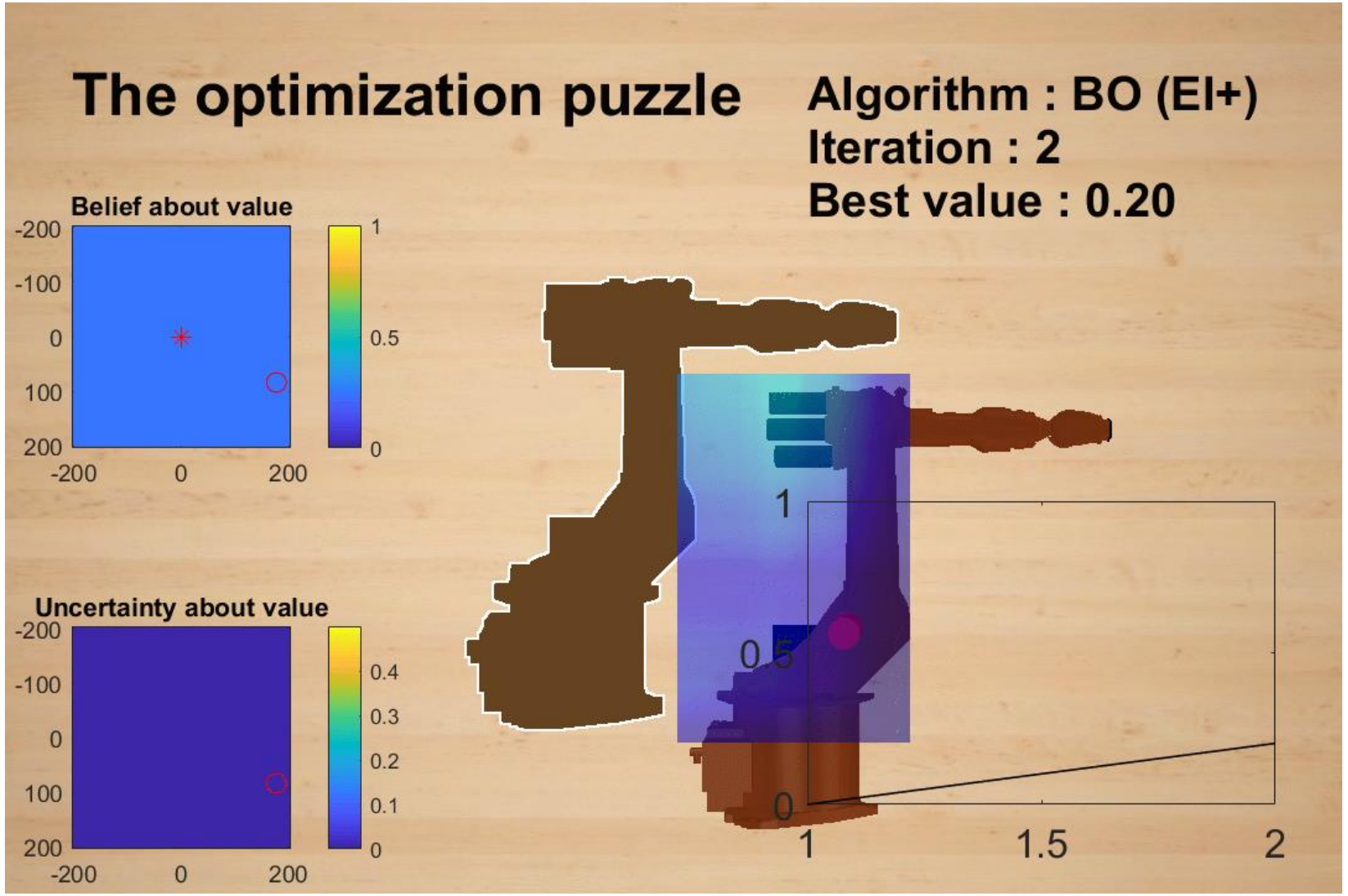




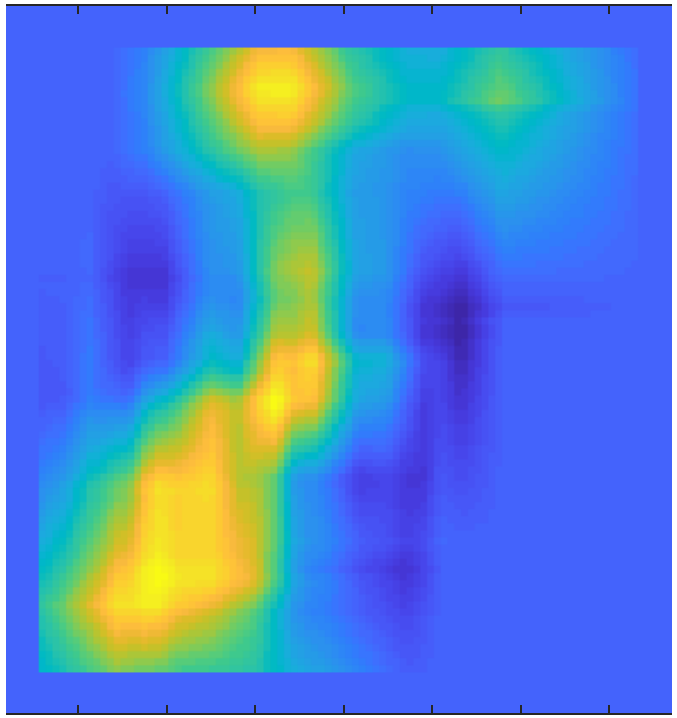
The optimization puzzle (1)



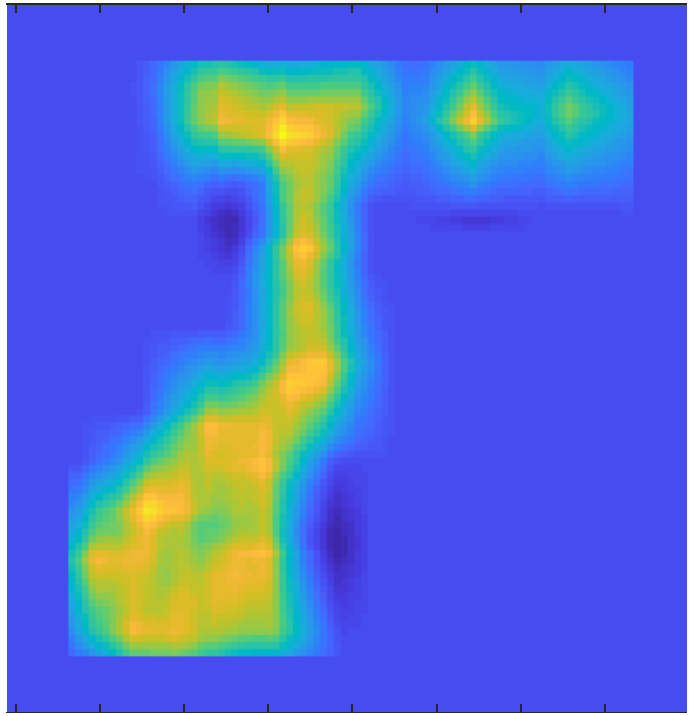
The optimization puzzle (1)



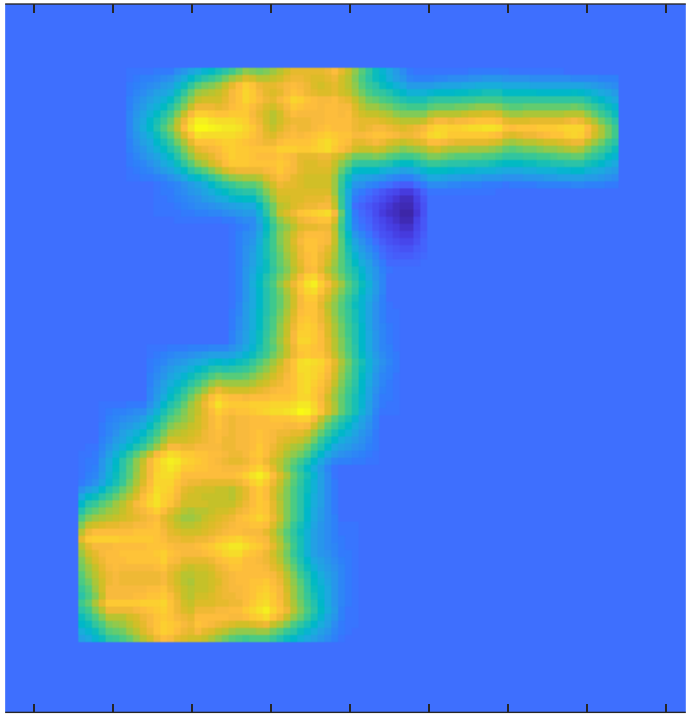
The optimization puzzle: Approximate puzzle piece



20 pseudo-inputs

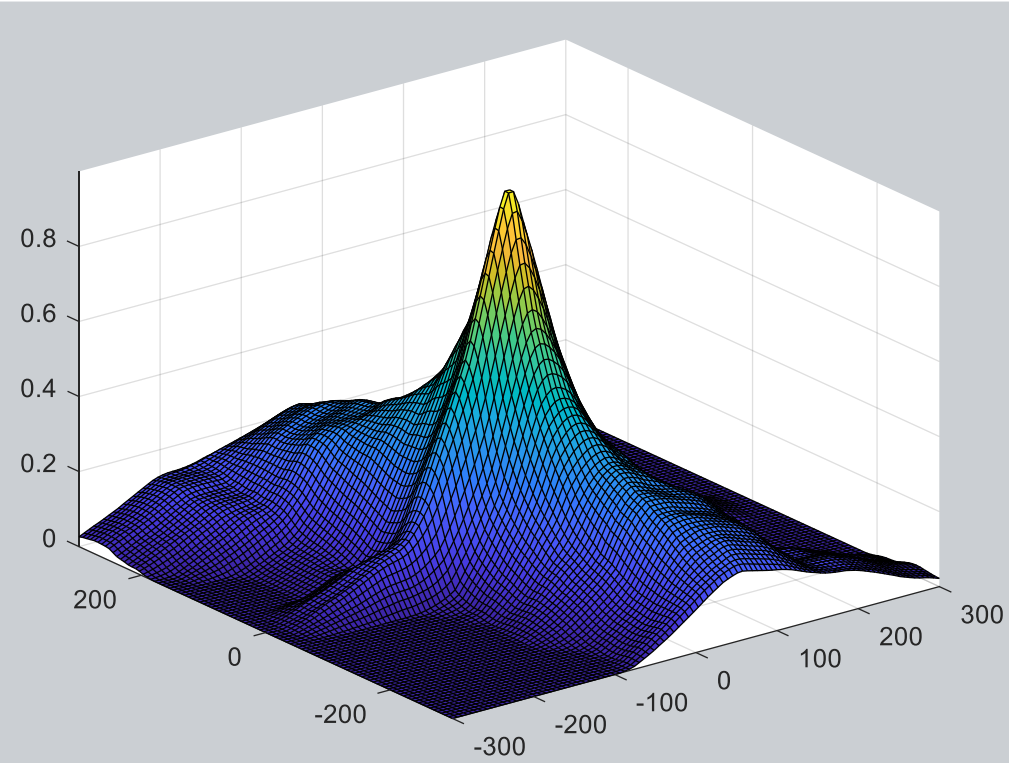
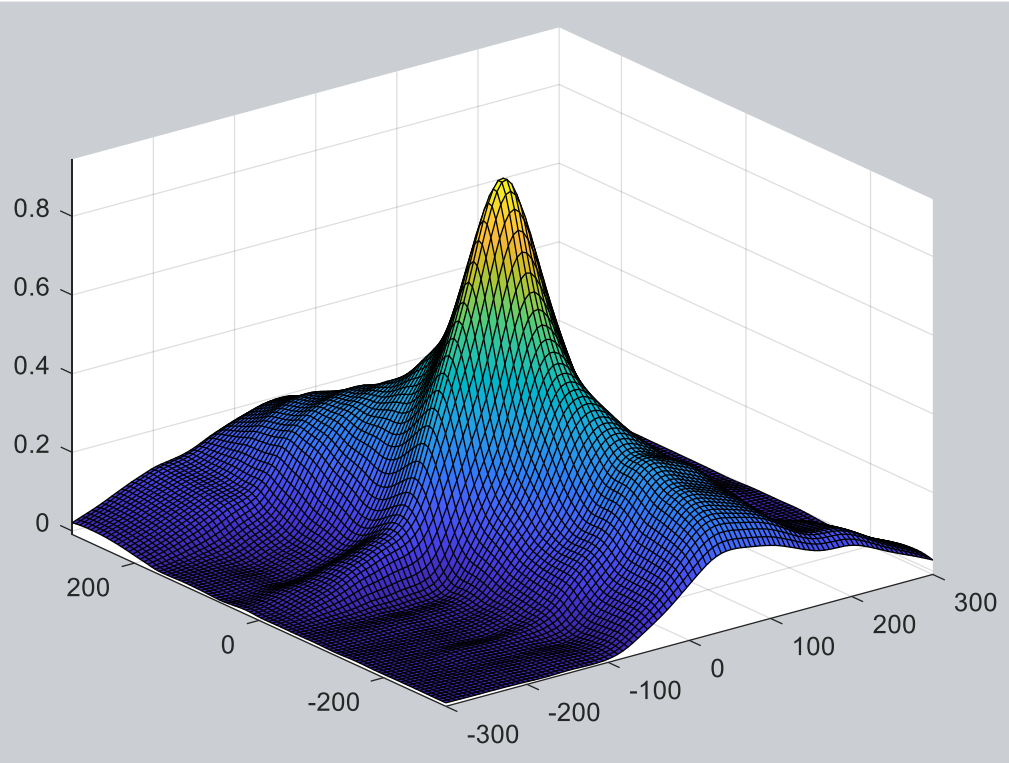


30 pseudo-inputs

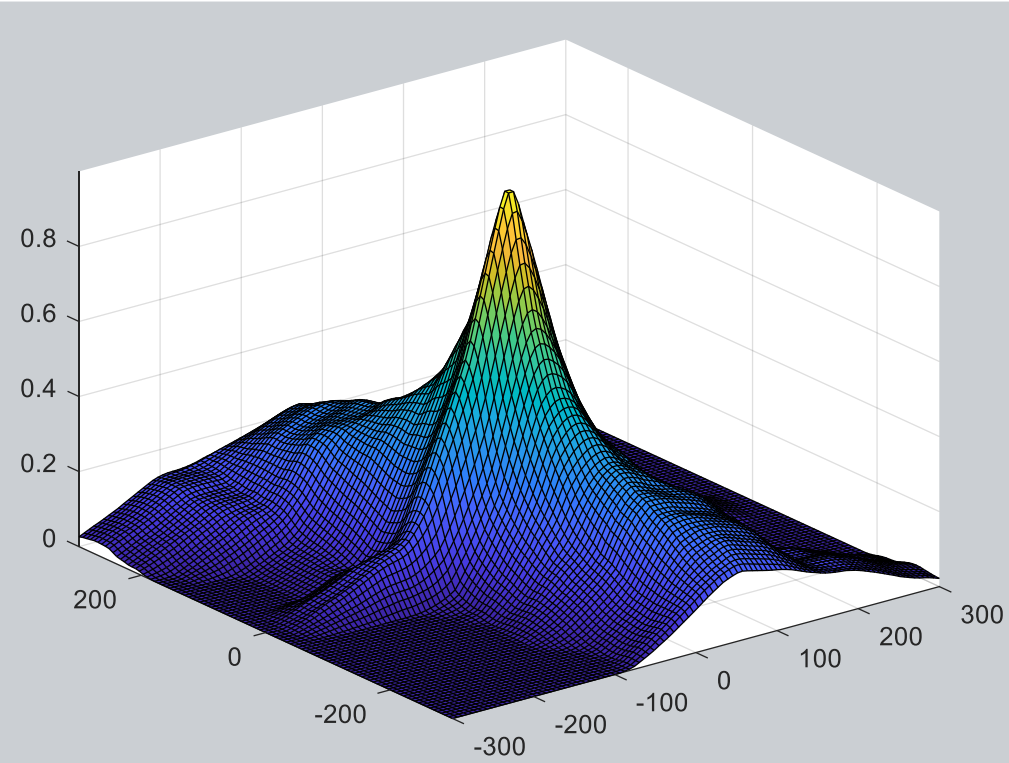
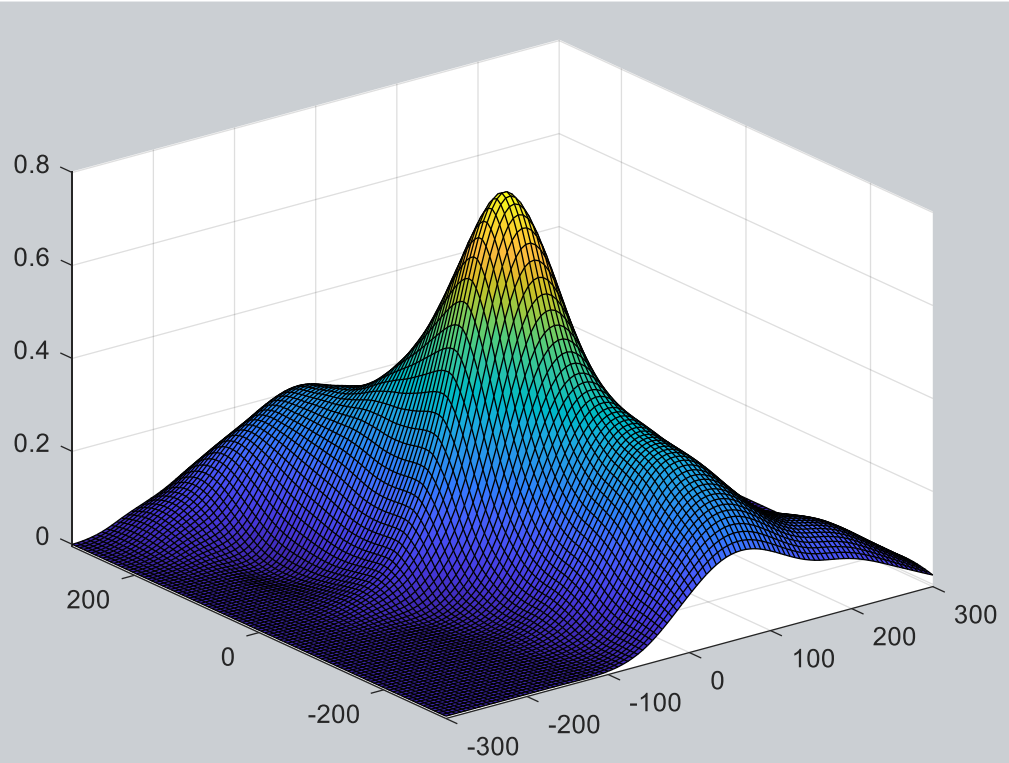


40 pseudo-inputs

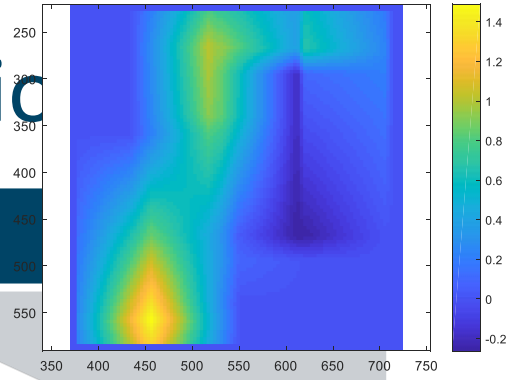
The optimization puzzle: Approximate cost function

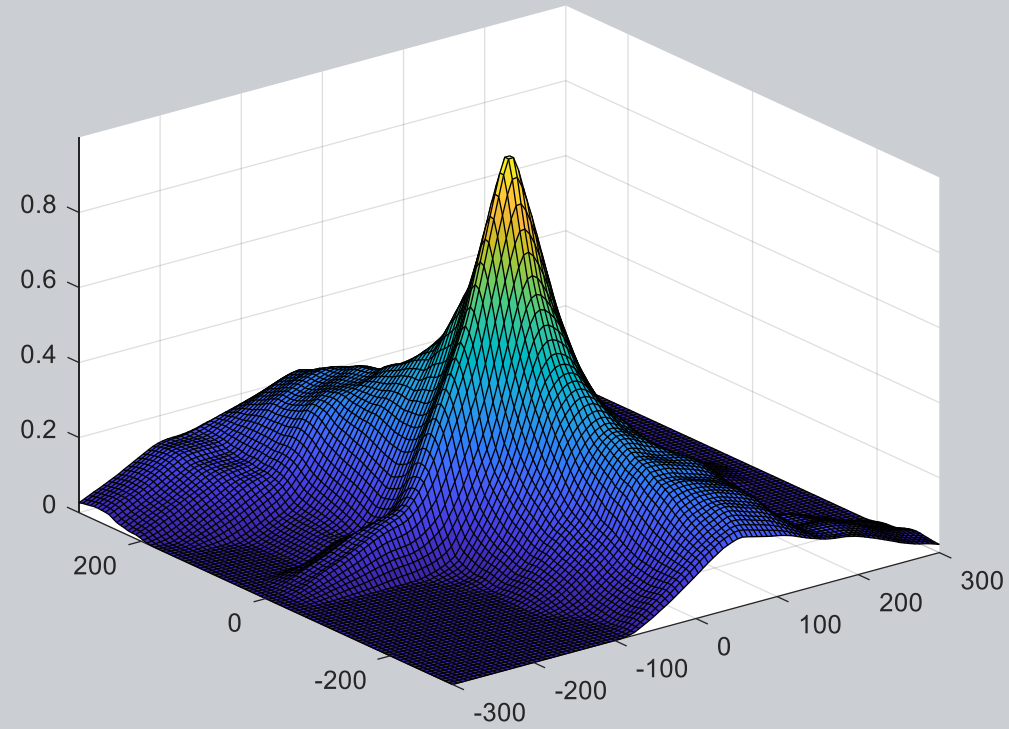
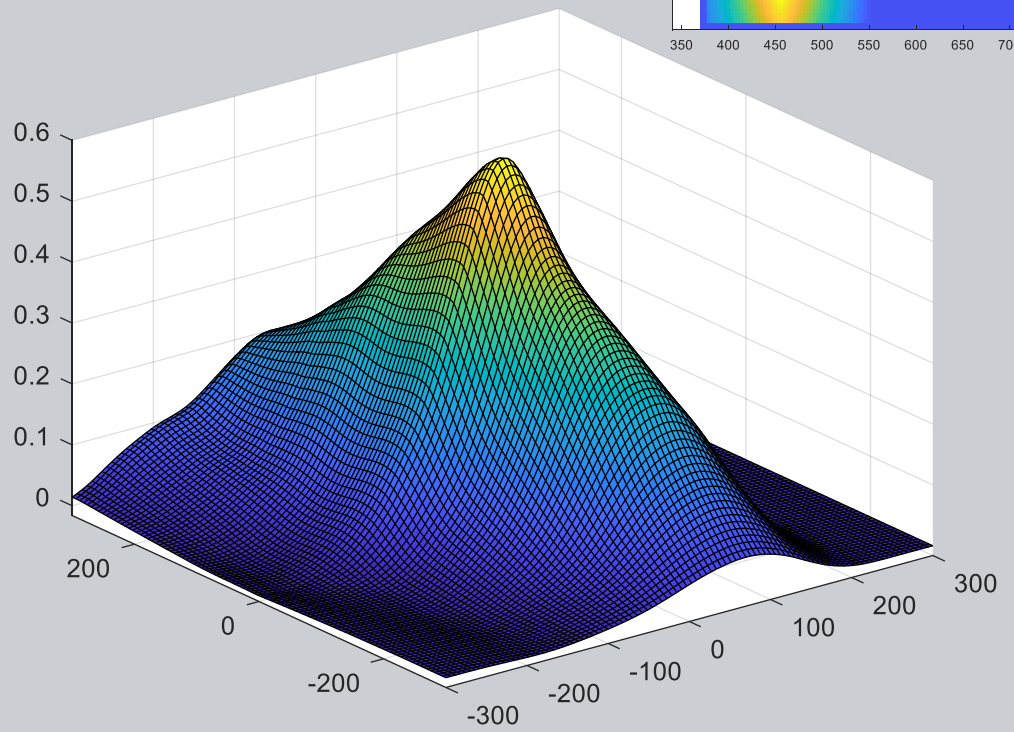
	Exact	100 pseudo-inputs
Cost function		
Distance optimum	**0**	**<1**

The optimization puzzle: Approximate cost function

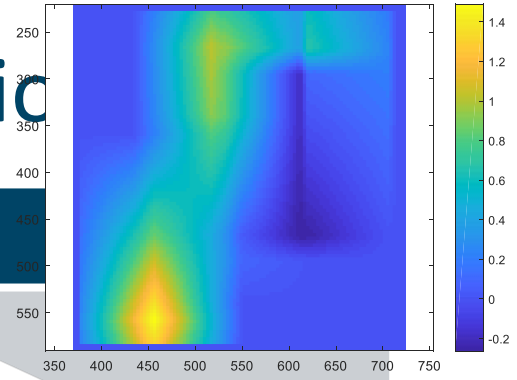
	Exact	20 pseudo-inputs
Cost function		
Distance optimum	**0**	**<1**

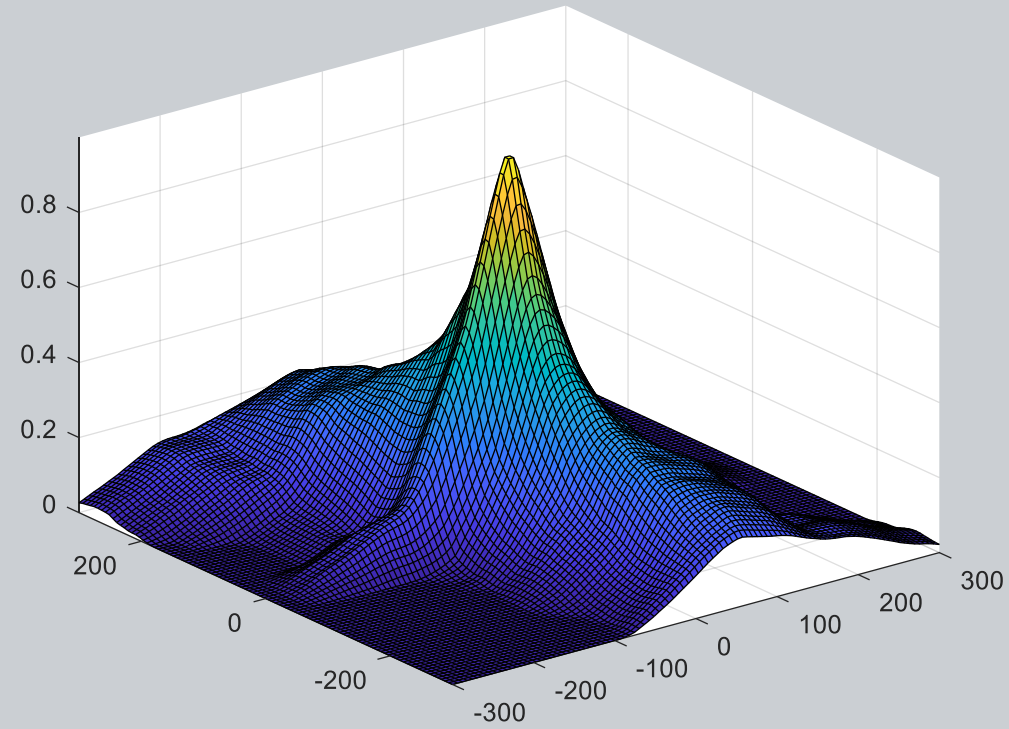
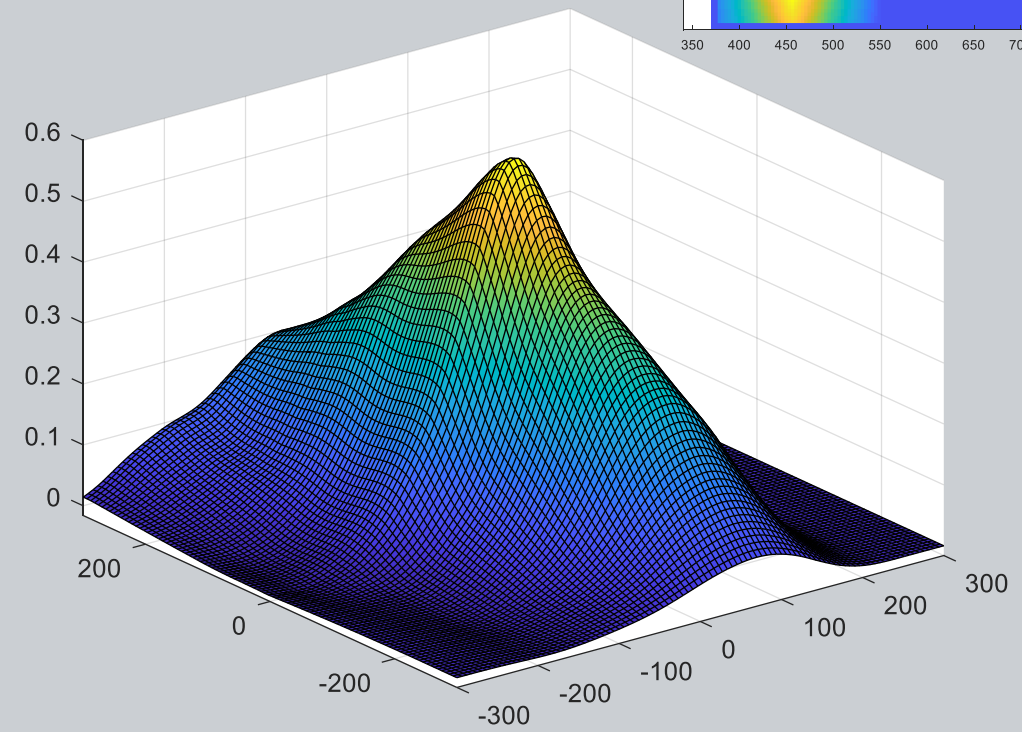
The optimization puzzle: Approximate cost functio



	Exact	5 pseudo-inputs !!!
Cost function		
Distance optimum	**0**	**<1**

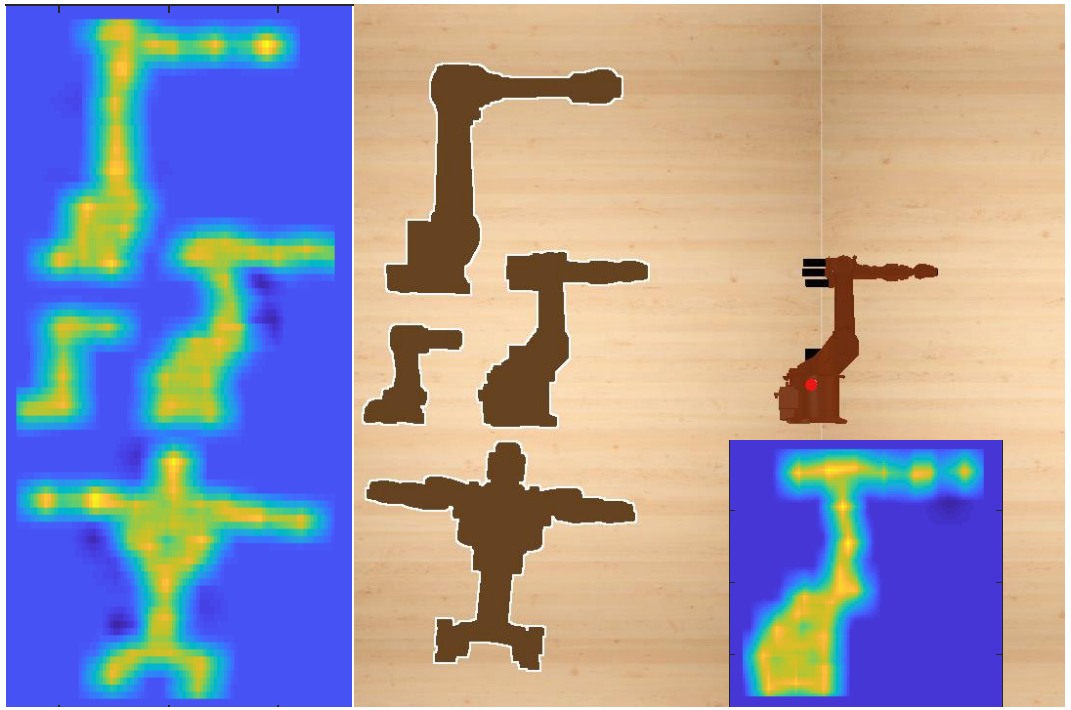
The optimization puzzle: Approximate cost functio



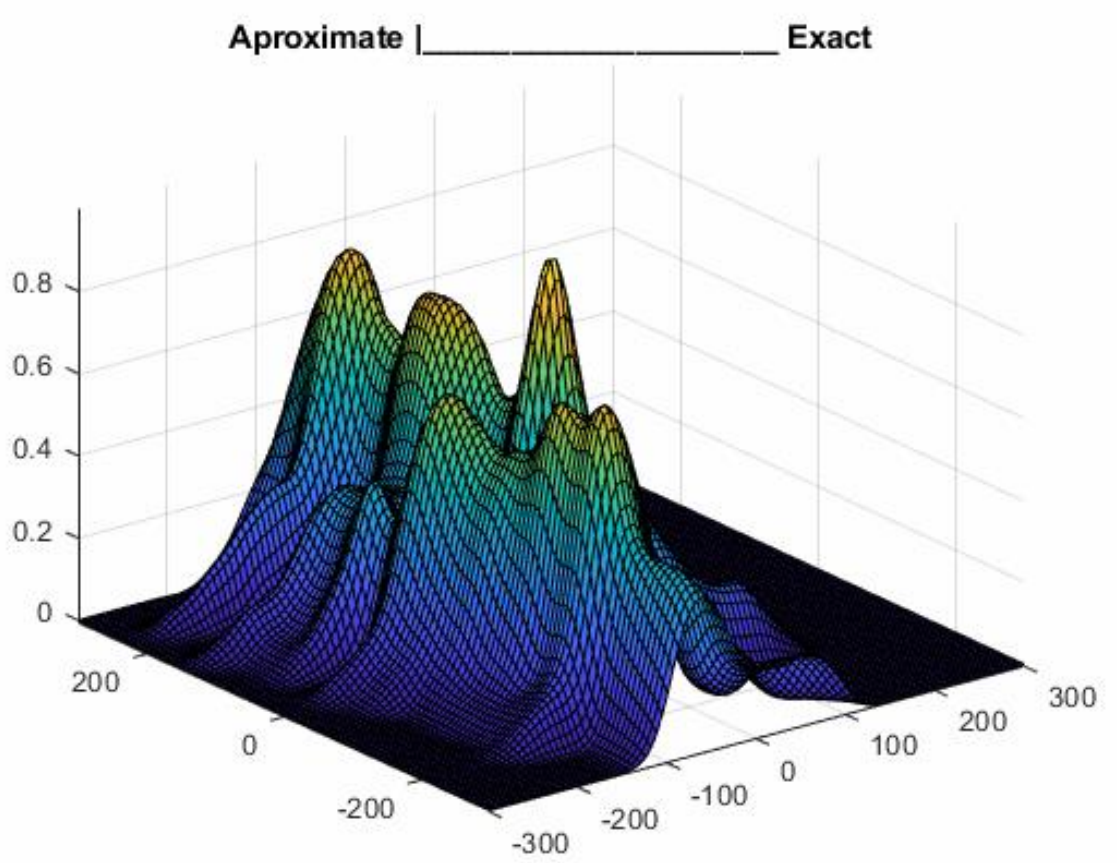
	Exact	5 pseudo-inputs !!!
Cost function		
Distance optimum	**0**	**<1**

Observation: Cost function becomes simpler/nicer !!!

The optimization puzzle: Approximate cost function

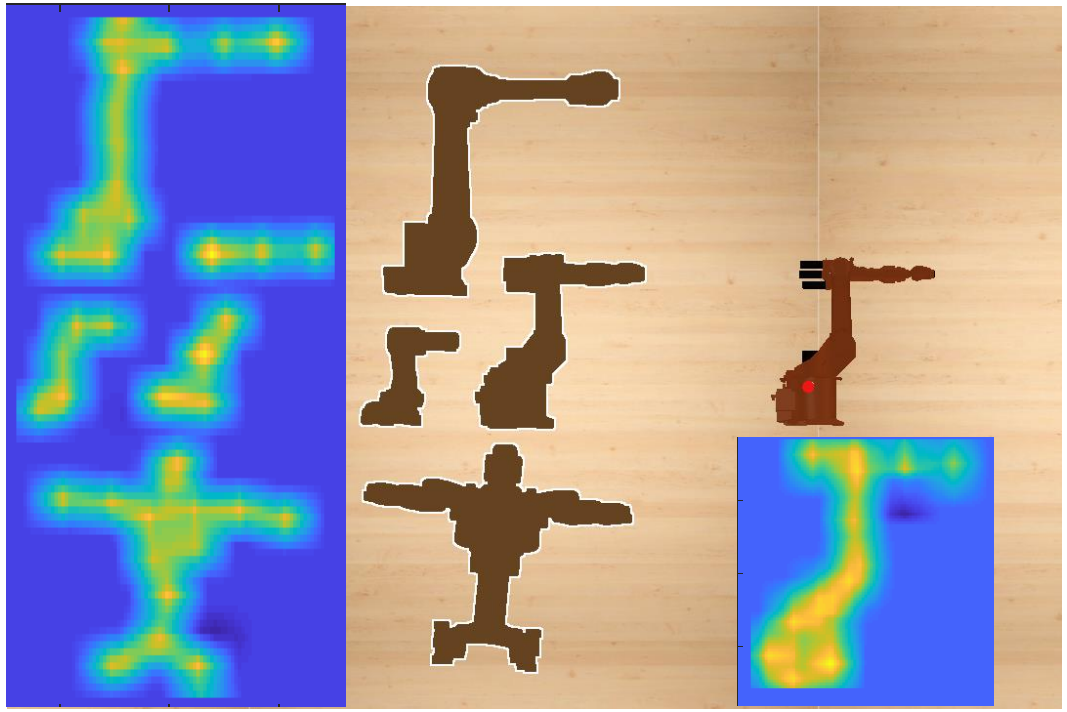


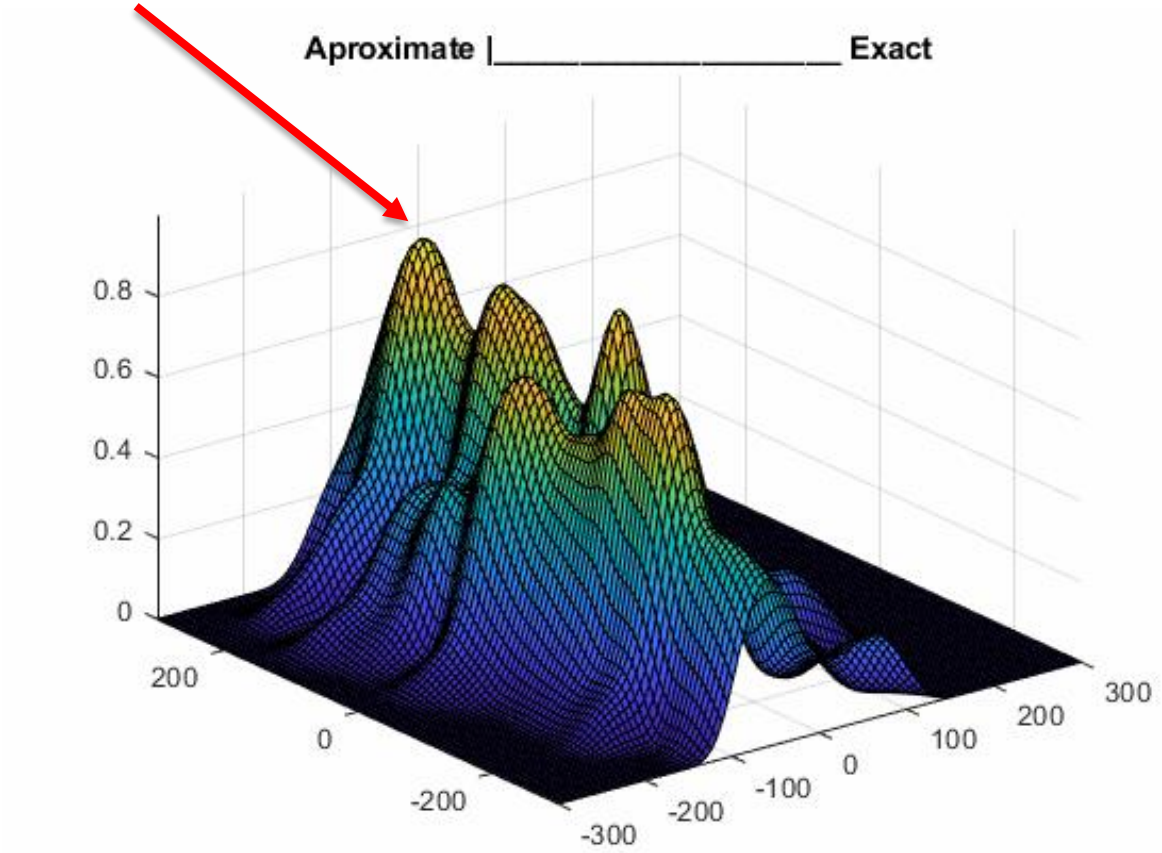
Puzzle	Piece
100 pseudo-inputs	30 pseudo-inputs



The optimization puzzle: Approximate cost function

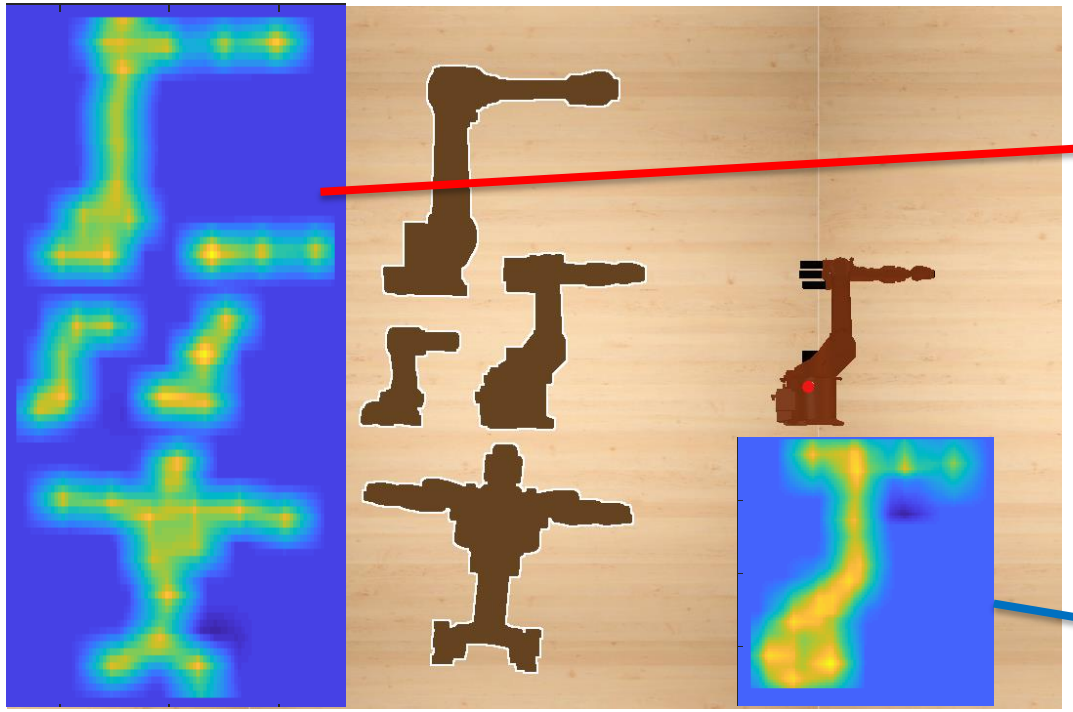
Fail, optimum changes

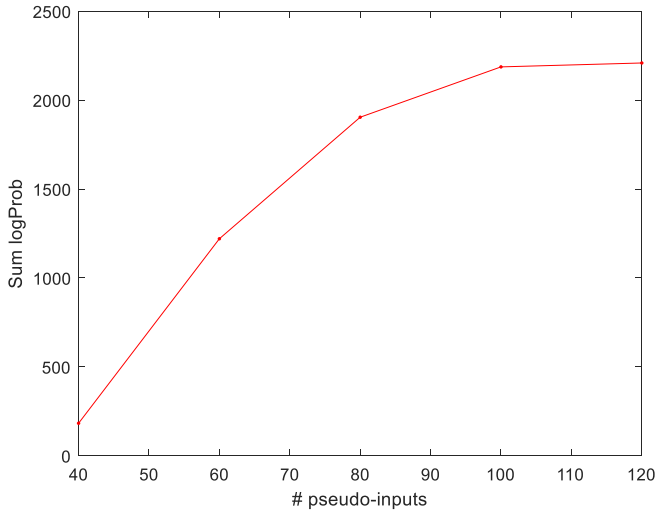


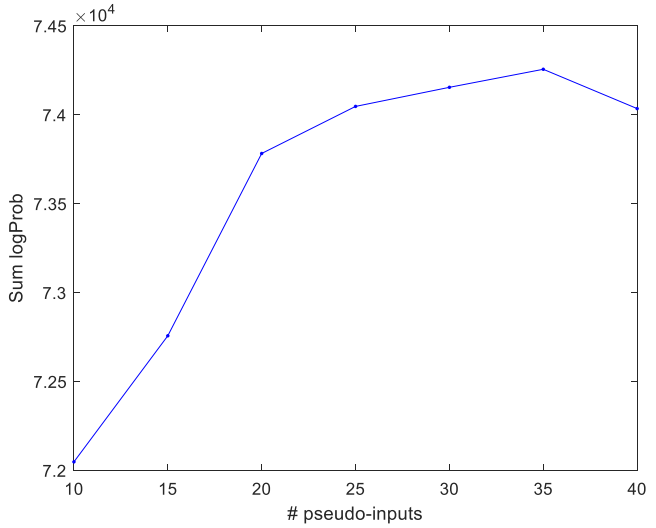


Puzzle	Piece
50 pseudo-inputs	15 pseudo-inputs

The optimization puzzle: Approximate cost function

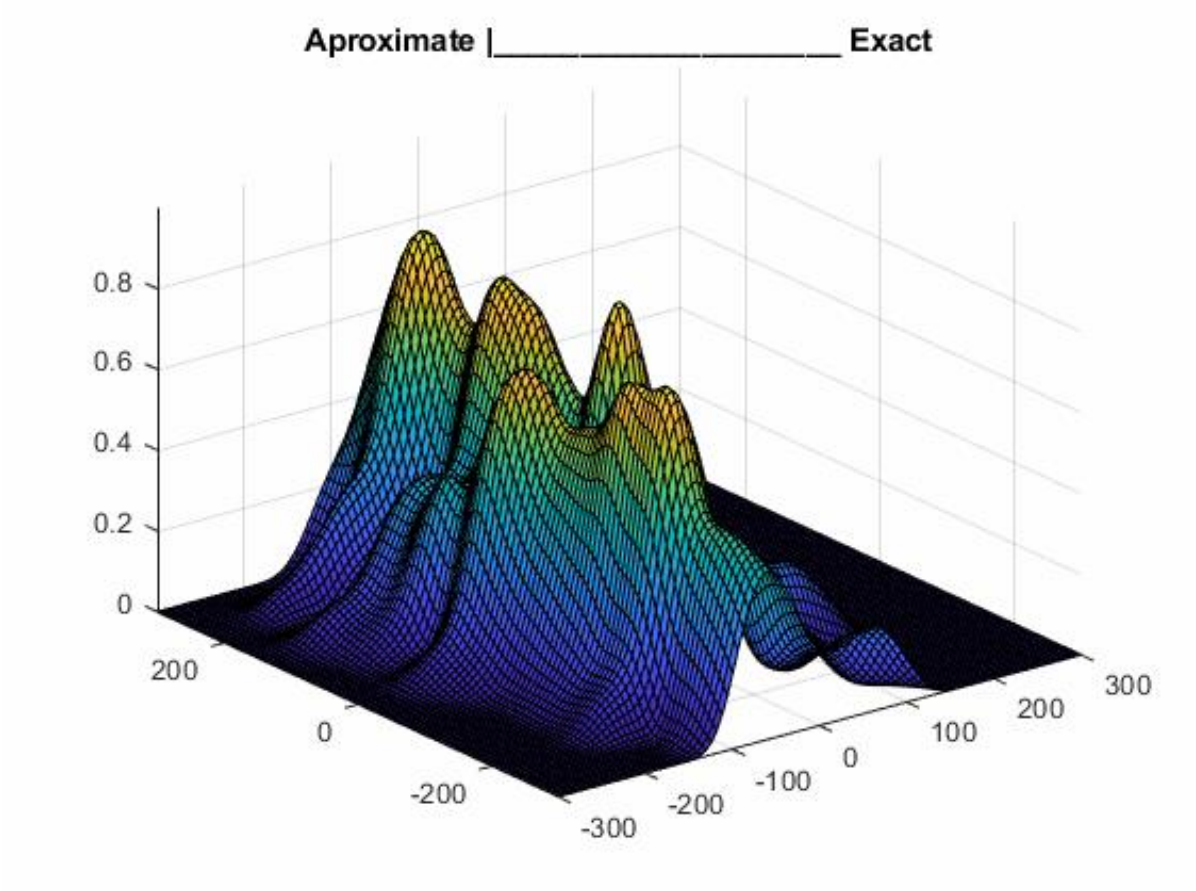


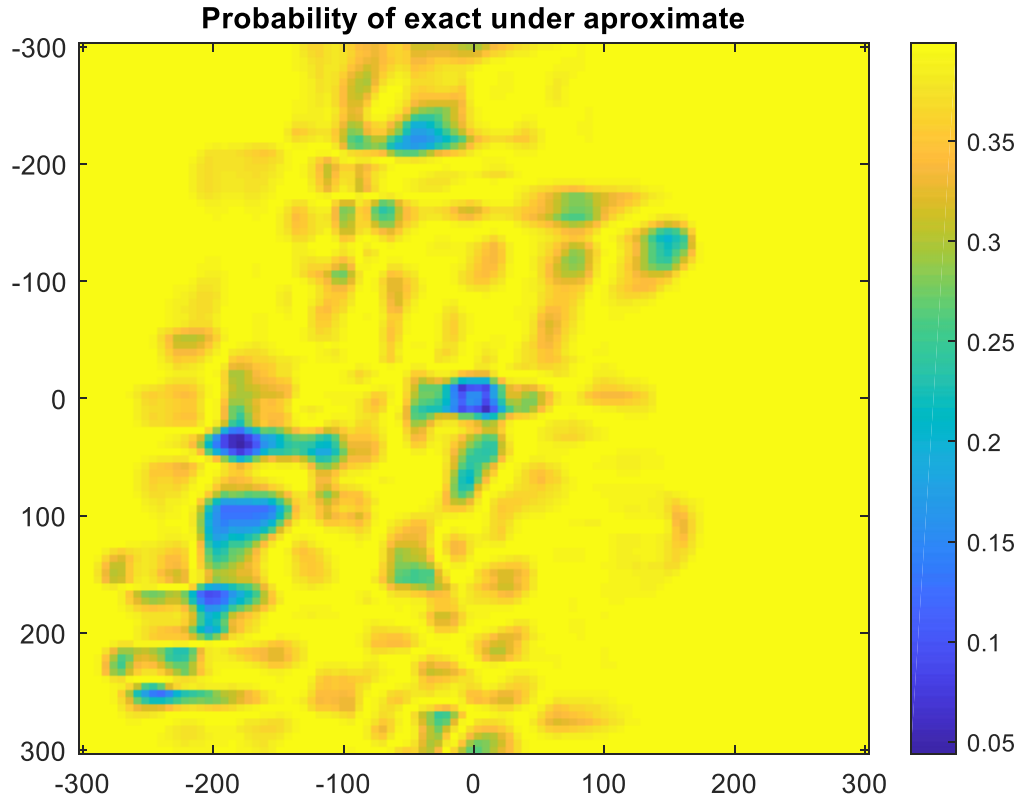




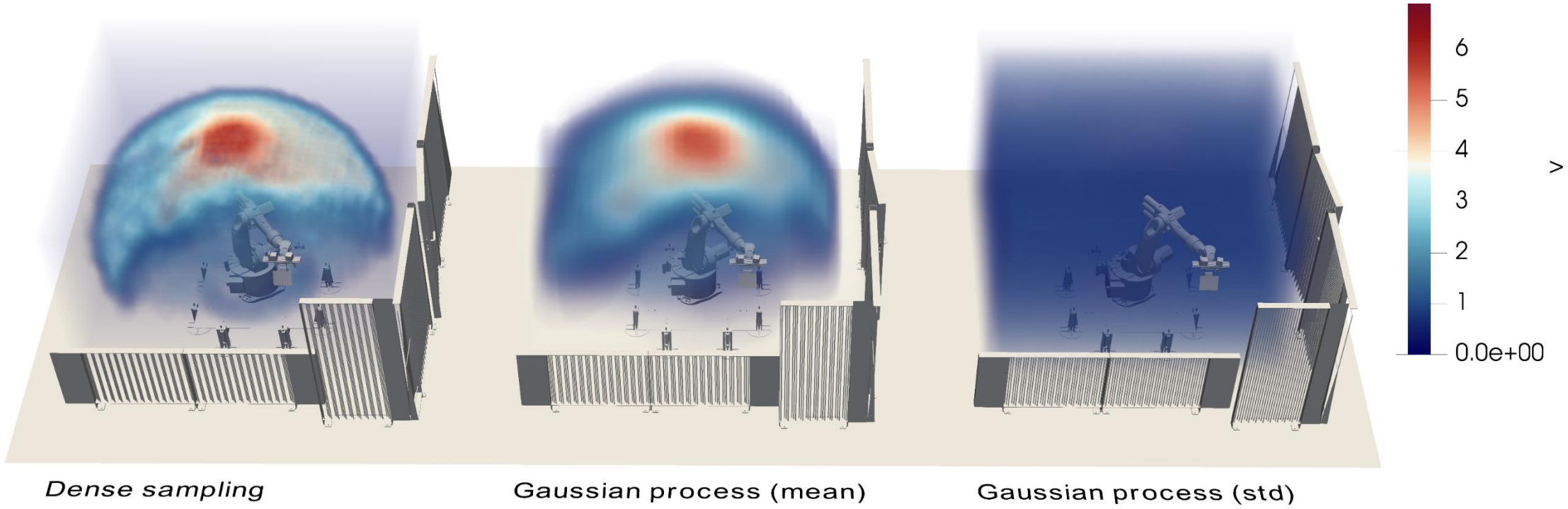
Puzzle	Piece
50 pseudo-inputs	15 pseudo-inputs

The optimization puzzle: Approximate cost function





Interesting partial results



*GP compressed (from 31 000 points) to only 50 pseudo-input points

Free VR demo if you visit Antwerp

Preliminary conclusion

- Application
- Machinery
- Theory

References

Briol, F.-X., Oates, C. J., Girolami, M. A., Osborne, M. A., & Sejdinovic, D. (2015). Probabilistic Integration. CoRR, abs/1512.00933.

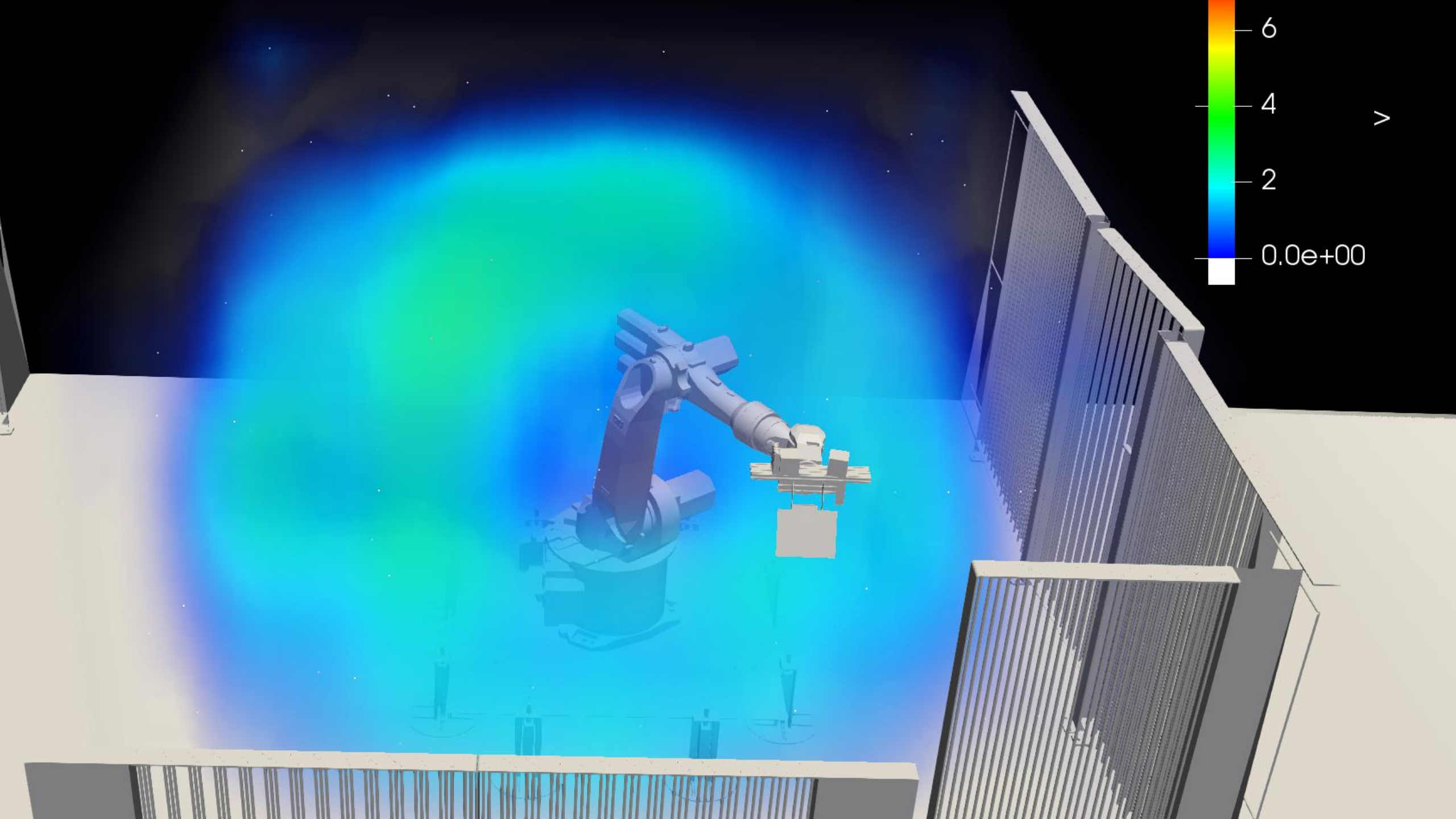
Halcon, B. (n.d.). Robot-enabled LDV basketball modal data collection trial run. Retrieved from https://www.youtube.com/watch?v=xx2Jl7fGf_E%0A

Karimi, M., Lucic, M., Hassani, H., & Krause, A. (2017). Stochastic submodular maximization: The case of coverage functions. In Advances in Neural Information Processing Systems (pp. 6856–6866).

Quiñonero-candela, J., Rasmussen, C. E., & Herbrich, R. (2005). A unifying view of sparse approximate Gaussian process regression. Journal of Machine Learning Research. https://doi.org/10.1163/016918609X12529286896877

Snelson, E., & Ghahramani, Z. (2006). Sparse Gaussian Processes using Pseudo-inputs. Advances in Neural Information Processing Systems 18. https://doi.org/10.1.1.60.2209

Zhang, H., & Vorobeychik, Y. (2016). Submodular Optimization with Routing Constraints. Proceedings of the 30th Conference on Artificial Intelligence (AAAI 2016).


6
4
2
0.0e+00
>