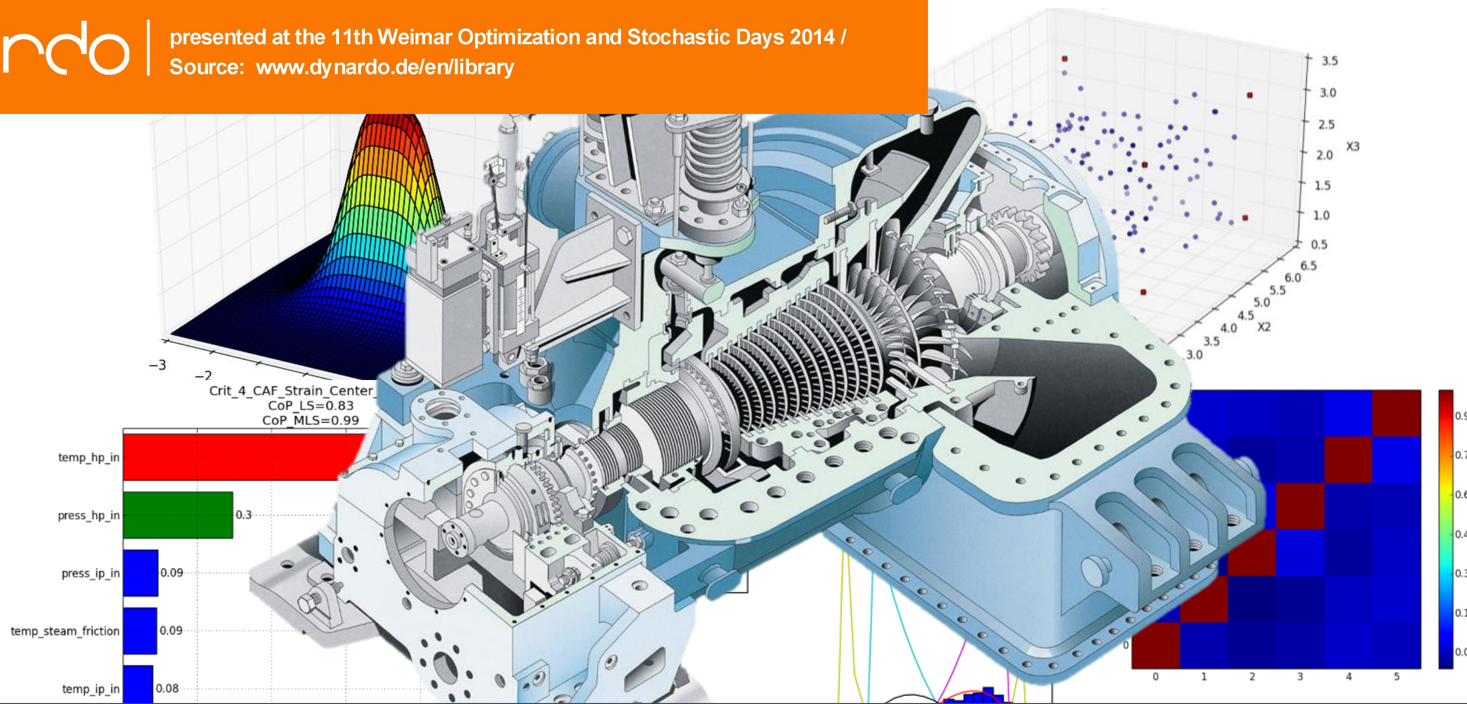


Requirements and new approaches of probabilistic optimal design from a practical point of view considering steam turbines

K.Cremanns*, D. Roos*, R. Voß**
*Institute of Modelling and High-Performance Computing, Niederrhein University of Applied Sciences
**Siemens AG

Outline

1. Introduction
   - Motivation
   - Objective
   - Planned application

2. Previous results
   - Optimized latin hypercube sampling
   - Advanced moving least square approximation

3. Outlook
   - Integration into optiSLang
   - Further developments

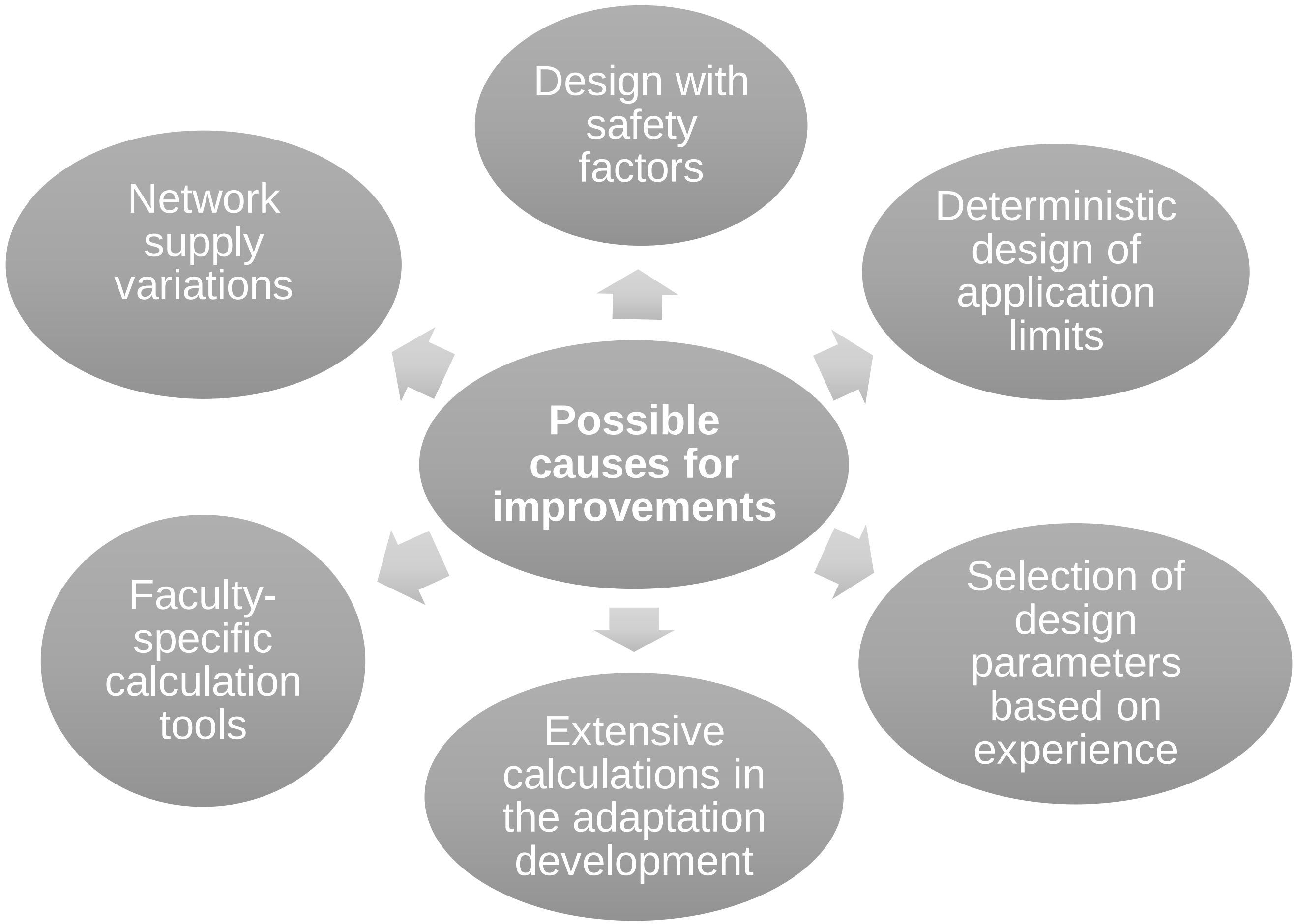
Outline

1. **Introduction**
   - Motivation
   - Objective
   - Planned application

2. Previous results
   - Optimized latin hypercube sampling
   - Advanced moving least square approximation

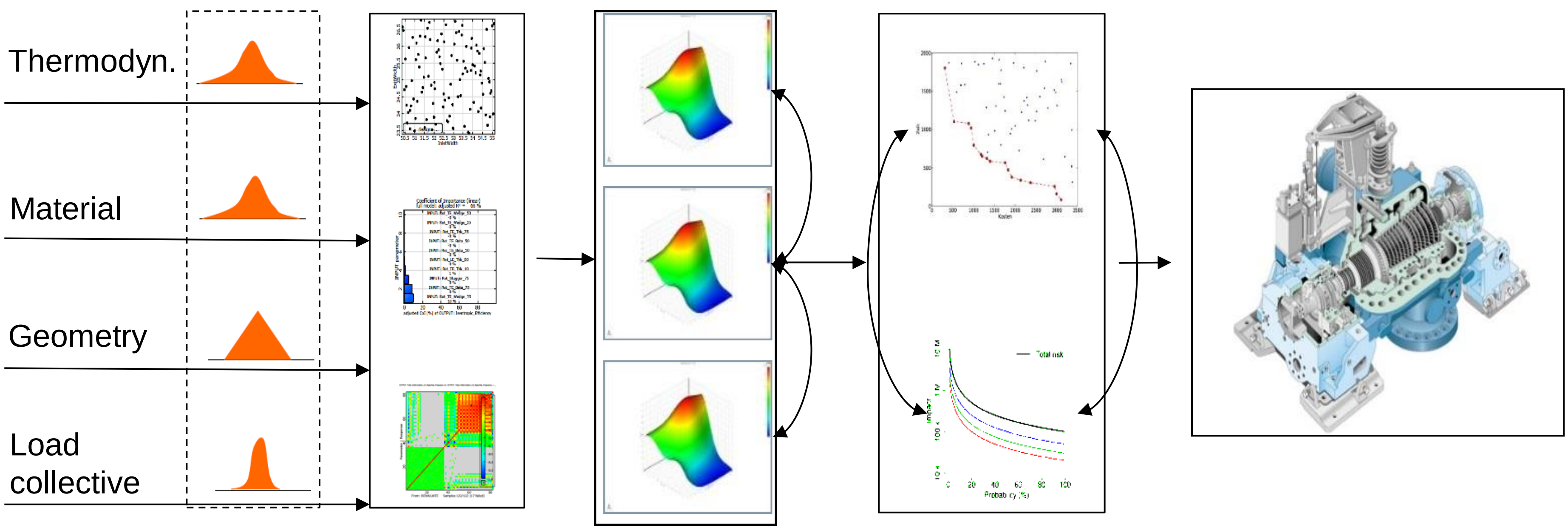
3. Outlook
   - Integration into optiSLang
   - Further developments

1. Introduction - Motivation



1. Introduction - Objective

Input parameters | **Design-sampling** | **Faculty-specific metamodels** | **Reliability-based multi-objective optimization** | **Optimal robust design taking into account the probability of failure**

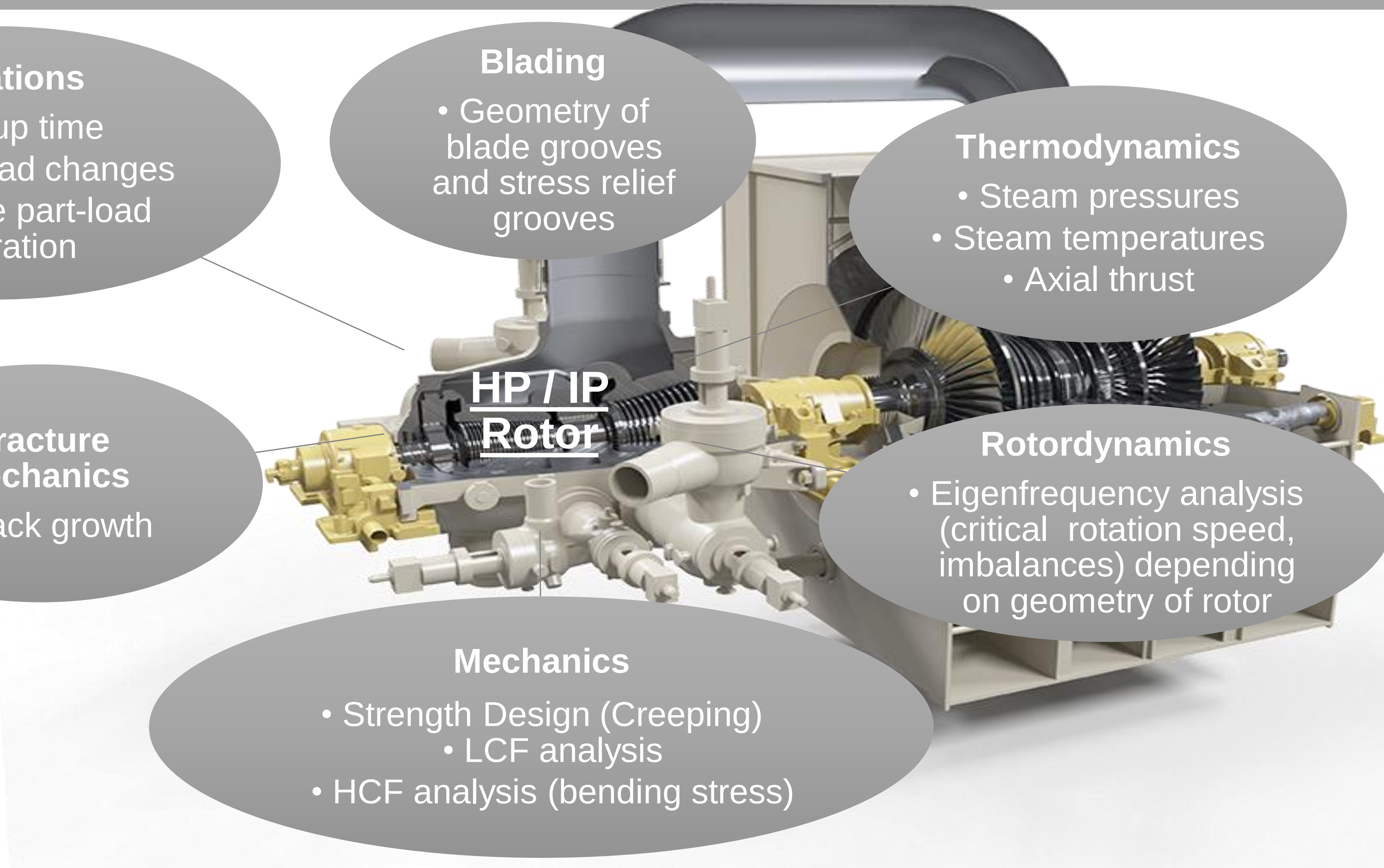


1. Introduction – Planned application

Operations
- Start-up time
- Time for load changes
- Allowable part-load operation

Blading
- Geometry of blade grooves and stress relief grooves

Thermodynamics
- Steam pressures
- Steam temperatures
- Axial thrust



Fracture mechanics
- Crack growth

Rotordynamics
- Eigenfrequency analysis (critical rotation speed, imbalances) depending on geometry of rotor

Mechanics
- Strength Design (Creeping)
- LCF analysis
- HCF analysis (bending stress)

1. Introduction – Planned application

„External" constraints:

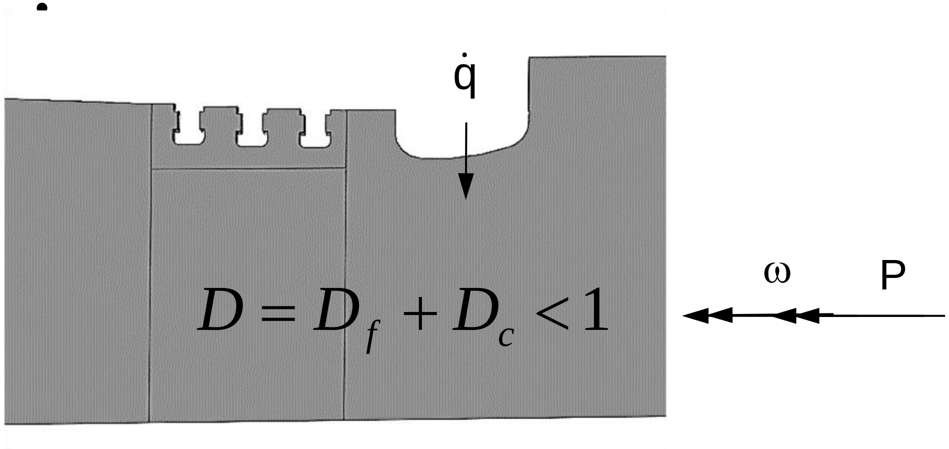
- Steam temperature (THD)
- Start times
- Required lifetime
- Number of starts
- Number of load cycles
- …

„Internal“ constraints :

- Fulfillment mechanical integrity rotor + blading
- Manufacturability
- Compliance criteria rotor dynamics
- …

Objectives:

- Long lifetime (h)
    - ➢ High number of starts (N)
    - ➢ High start speed (MW / min)
    - ➢ High number of load cycles (N)
- High performance (MW)
- High efficiency (mu)
- Eigenfrequencies outside critical regions (50Hz, 60 Hz, 100Hz, 120Hz)



Outline

1. Introduction
   - Motivation
   - Objective
   - Planned application

2. **Previous results**
   - Optimized latin hypercube sampling
   - Advanced moving least square approximation

3. Outlook
   - Integration into optiSLang
   - Further developments

2. Previous results - Optimized latin hypercube sampling

Base : **standard latin hypercube sampling**

- Parameter space is divided into N (samples or classes) with the same probability of 1 / N, and in each of these classes a random point is selected.

Cons:

➢It can cause unwanted input parameter correlations.

➢It does not guarantee a "filling" coverage of the parameter space.

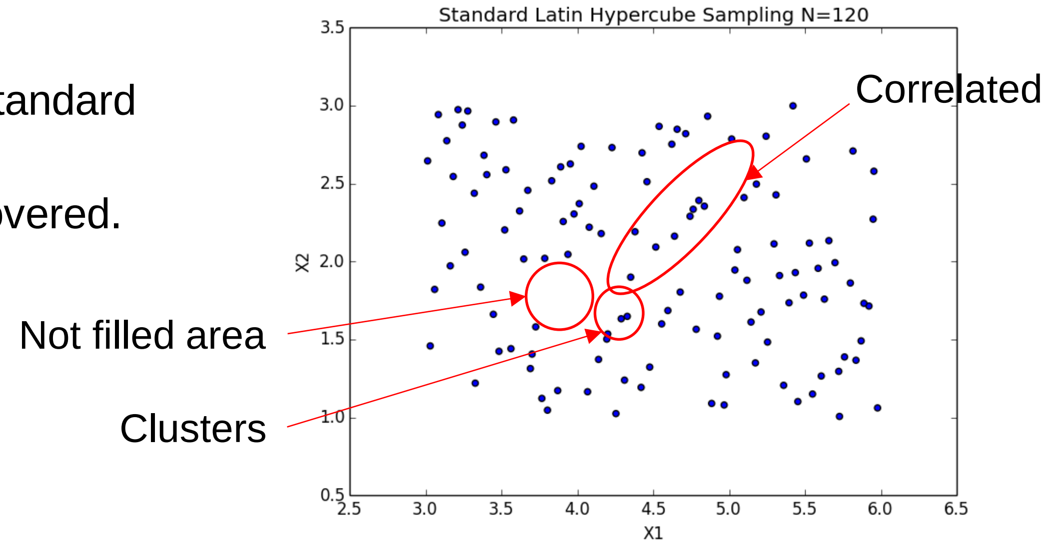
Advantages:

➢Low computation time to generate.

➢It has a lower variance compared to standard Monte Carlo sampling.

➢The value range of each variable is covered.

➢Non – collapsing.



2. Previous results - Optimized latin hypercube sampling

Optimized latin hypercube sampling (OHLS):

- The standard latin hypercube sampling gets improved by optimization.

- Optimization criteria are:
  - Mean value of the linear correlation coefficients ρ_{ij} (modified criterion of Owen [1]):

$$\rho_{avg} = \frac{\sum_{i=2}^{n} \sum_{j=1}^{i-1} |\rho_{ij}|}{n(n-1)/2}; 0 \leq \rho_{avg} \leq 1$$

  - Euclidean Maximin [2][3] design coefficient (scaled to [0,1]):

$$l_2 = d(x_i, x_j) = min - \sqrt{\sum_{k=1}^{n} |x_{ik} - x_{jk}|^2}; 0 \leq l_2 \leq 1$$

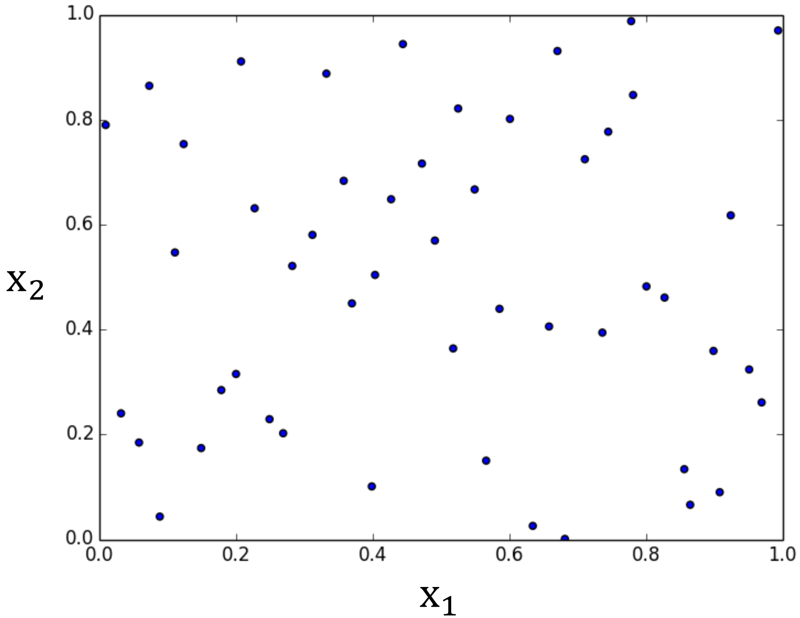
- Overall criterion: $f(\rho_{avg}, l_2) = w_1 \rho_{avg} + w_2 l_2 = \Psi$; with $w_1 + w_2 = 1; w_1 = 0.5, w_2 = 0.5$

- Optimization method: **Simulated Annealing (SA)** [4]

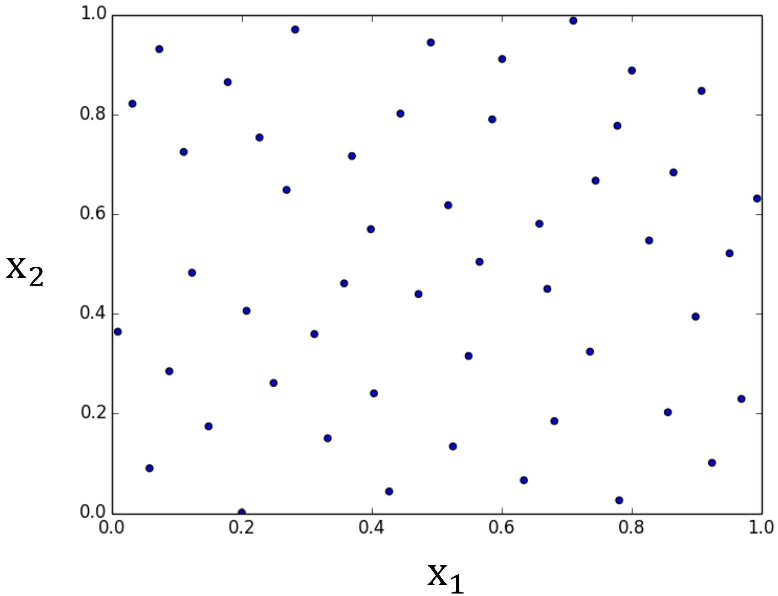
2. Previous results - Optimized latin hypercube sampling

Comparison for N=50 n=2:

Standard LHS



Own OLHS

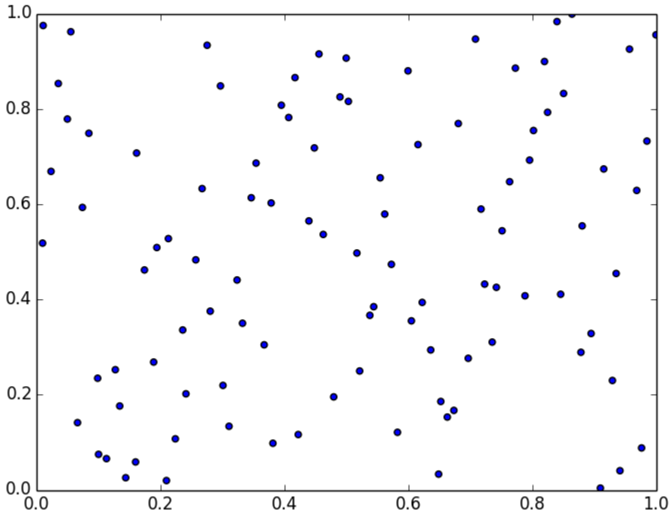


$\psi = -0.0117$ → after optimization → $\psi = -0.0394$

2. Previous results - Optimized latin hypercube sampling

Comparison for Matlab lhsdesign N=100 n=10: criterion maximin

Matlab lhsdesign criterion: maximin
1.000.000 iterations

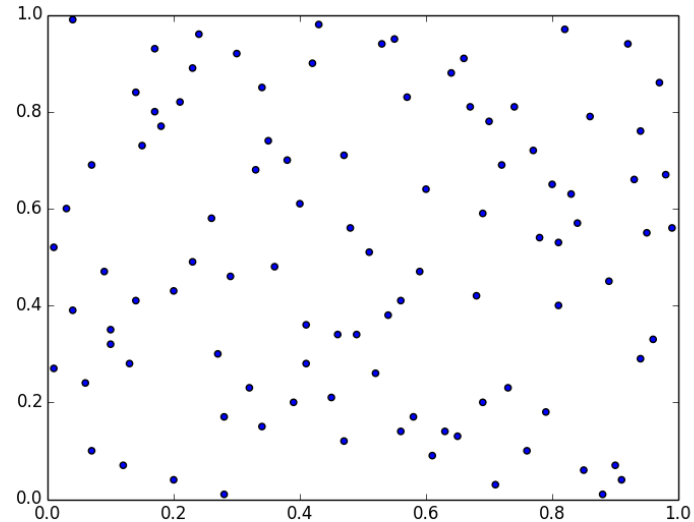


x_2

x_1

$l_2 = 0.191$

Own OLHS 10.000 iterations



x_2

x_1

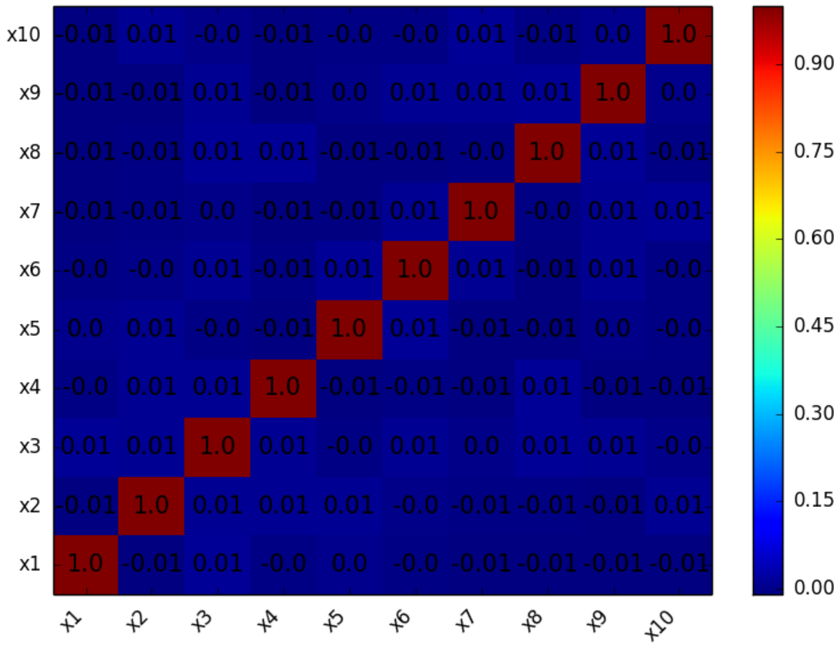
$l_2 = 0.212$

scaled minimal distance between design points

2. Previous results - Optimized latin hypercube sampling

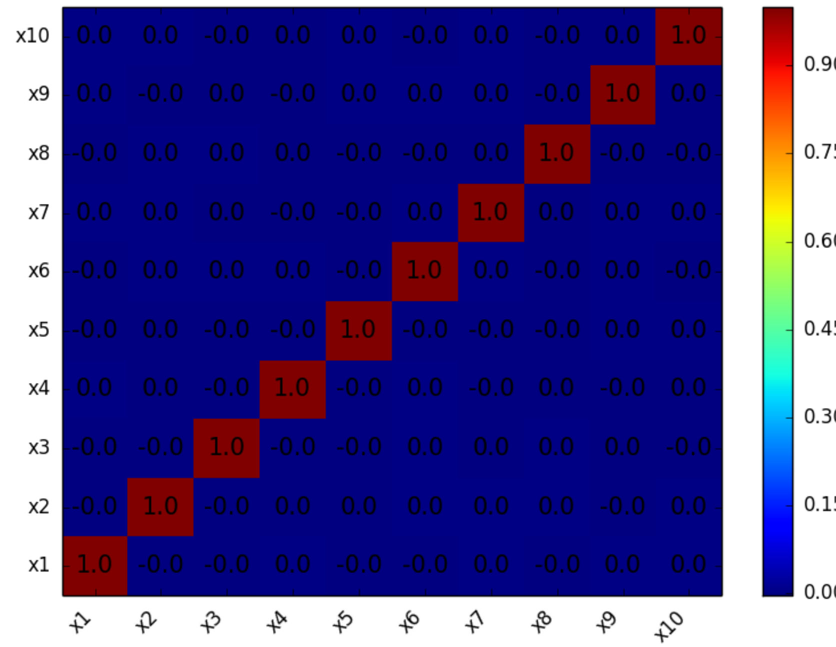
Comparison for Matlab lhsdesign N=100 n=10: criterion: orthogonality

Matlab lhsdesign criterion: orthogonality
1.000.000 iterations



$\rho_{avg} = 0.692\text{E} - 2$

Own OLHS 10.000 iterations

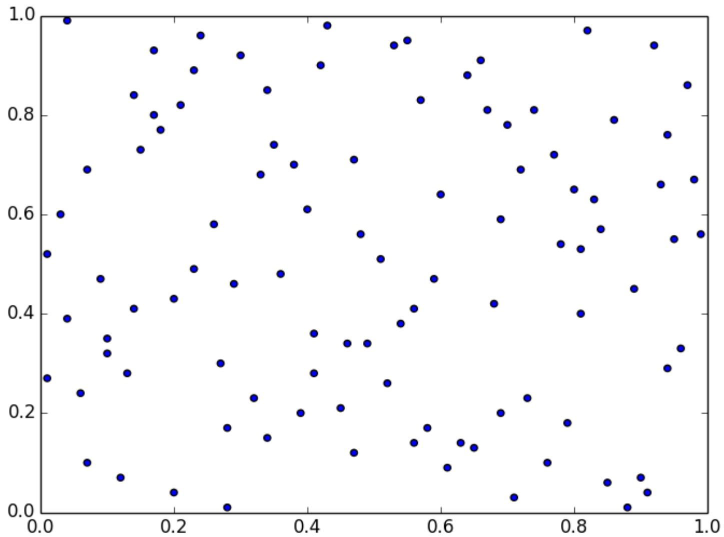


ρ_{avg} =0.694E-3

Comparison for optiSLang N=100 n=10: Advanced latin hyper cube sampling (ALHS)

optiSLang ALHS (optimized correlation)

x_2

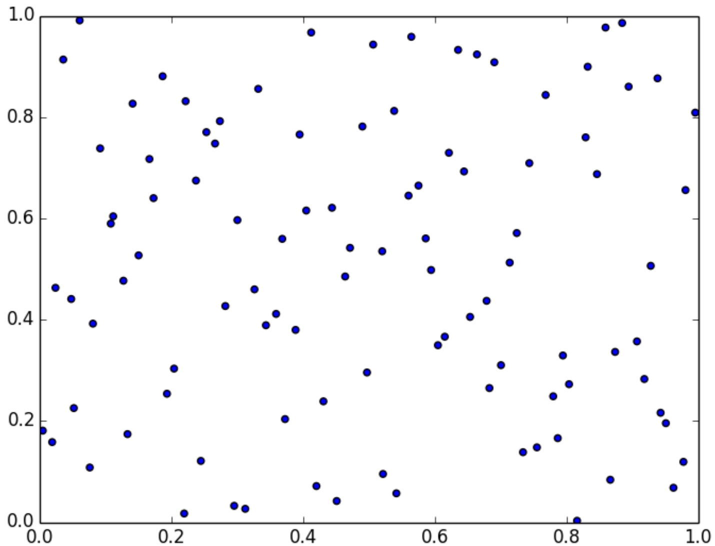


x_1

$l_2 = 0.152$

Own OLHS 10.000 iterations

x_2



x_1

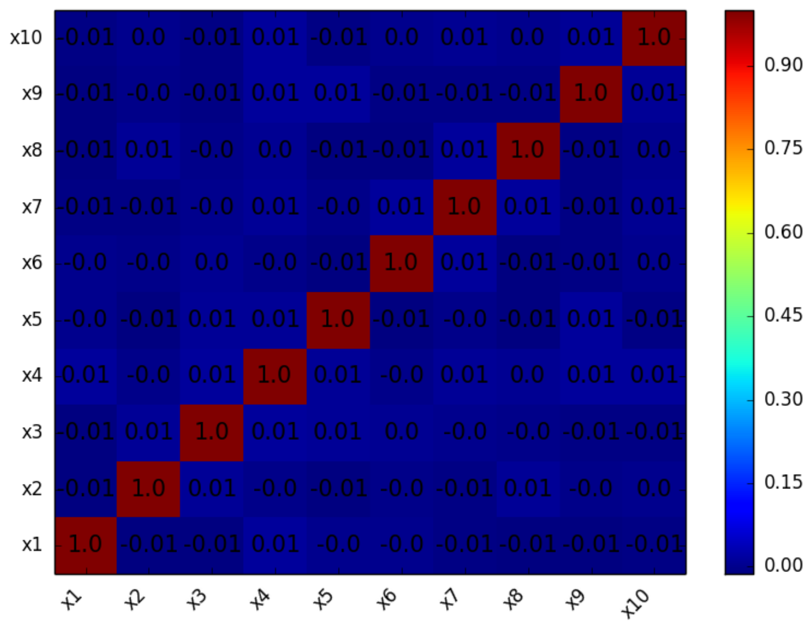
$l_2 = 0.212$

scaled minimal distance between design points

2. Previous results - Optimized latin hypercube sampling

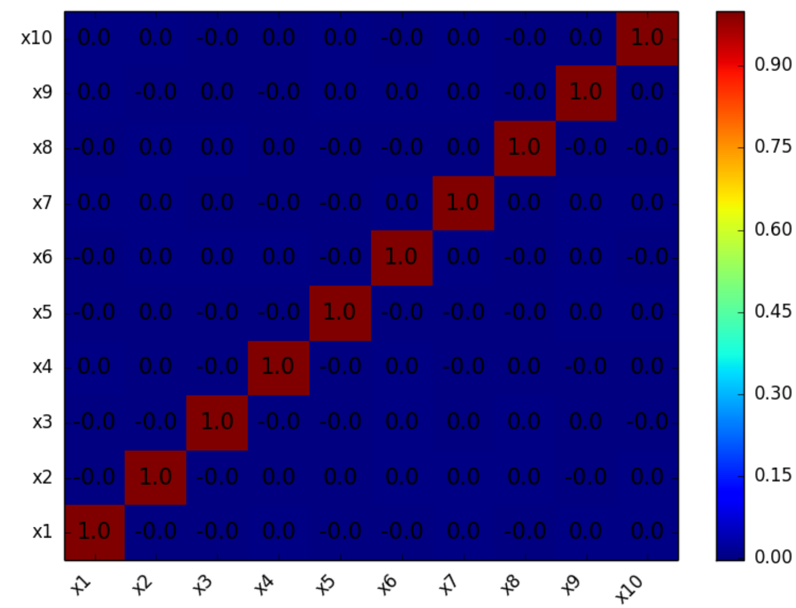
Comparison for optiSLang N=100 n=10: Advanced latin hyper cube sampling (ALHS)

optiSLang ALHS (optimized correlation)



ρ_{avg} =0.72E-2

Own OLHS 10.000 iterations

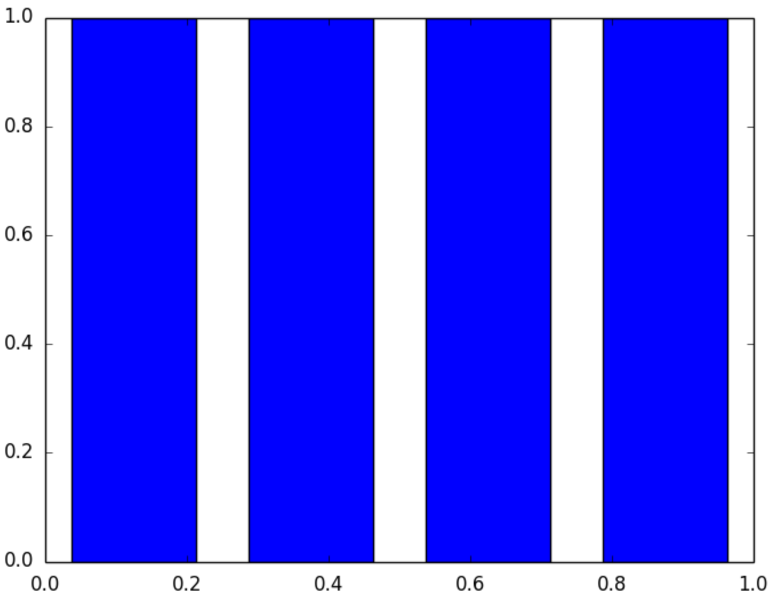


$\rho_{avg} = 0.694\text{E}{-3}$

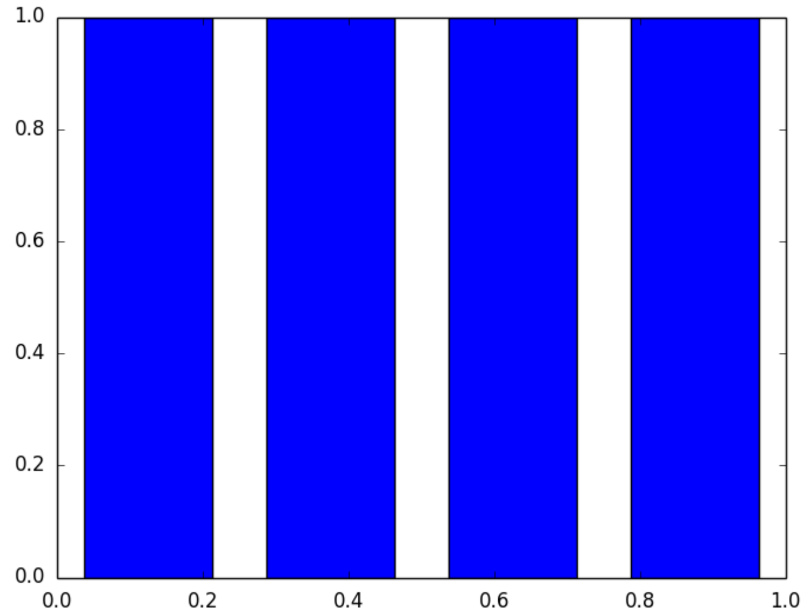
2. Previous results - Optimized latin hypercube sampling

Uniformity

Own OLHS 10.000 iterations



X1



X2

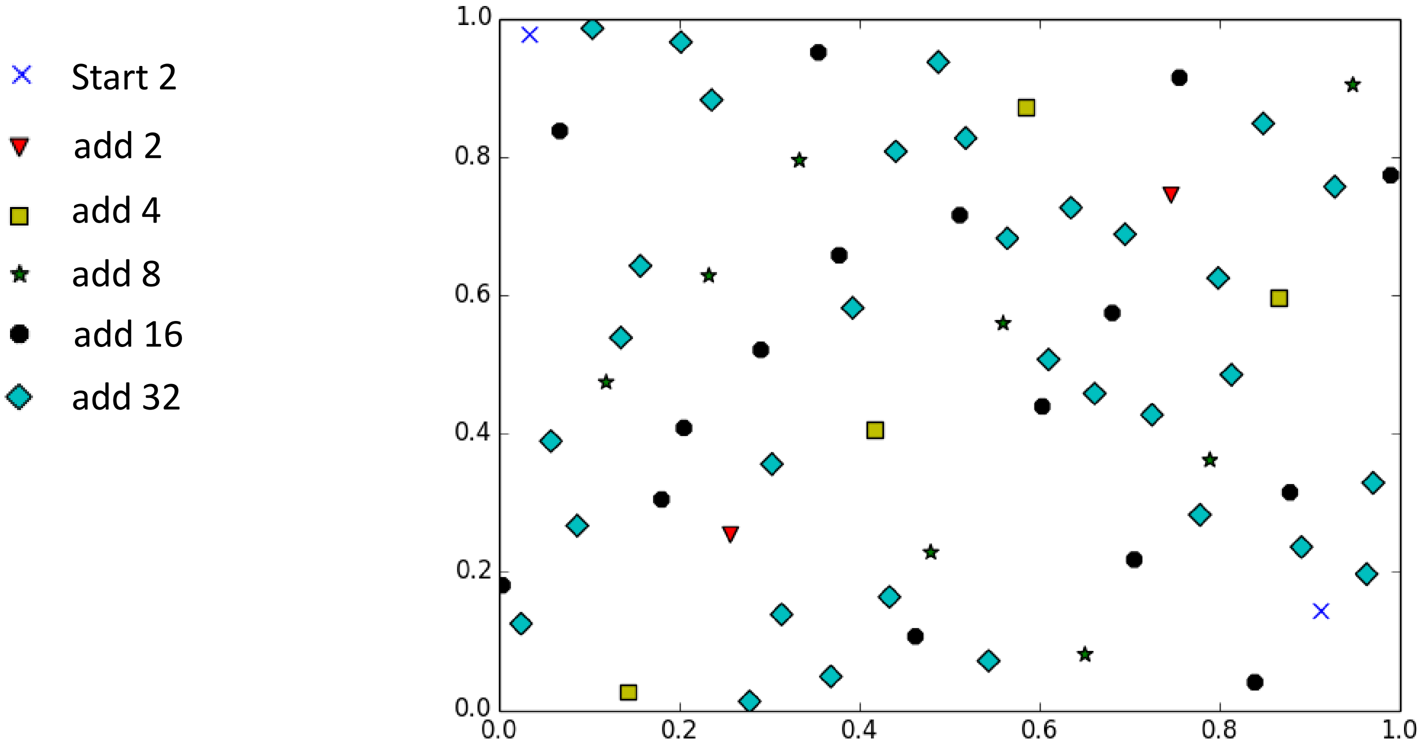
2. Previous results - Optimized latin hypercube sampling

- **Sequentiell OLHS (SOLHS):**
  - Start with a OLHS with presented method with e.g. 2 samples.
  - Add sample points in that way, that each step the number of samples double (necessary to keep **uniformity**).
  - The new points are also optimized with a combination of random picks and SA regarding ψ of the whole new design matrix (existing points + additional points).
  - Continue until metamodel is convergent (further steps are adding 4, 8, 16… samples).

- Advantages:
  - This method can result in a lower number of needed calculated samples, but still in a oversampling, because the fixed number of additional sets.
  - Over the iterations it‘s possible to check the convergence of the metamodel.
  - Every new created LHS is still a OLHS, with optimal settings for maximin and correlation.

2. Previous results - Optimized latin hypercube sampling

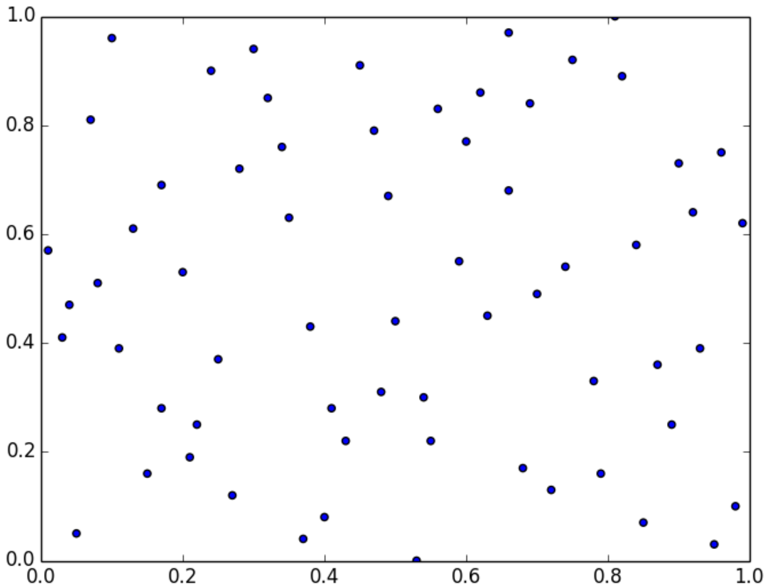
- Example for 64 samples with SOLHS (steps: adding 2, 4, 8, 16, 32 samples)



2. Previous results - Optimized latin hypercube sampling

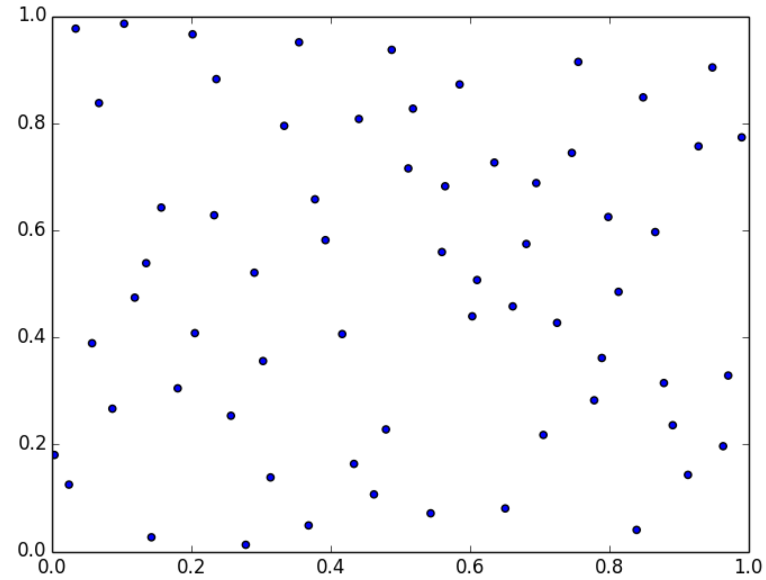
Comparison for Matlab N=64 n=2: maximin and SOHLS 64 samples

Matlab OLHS criterion: maximin
100.000 iterations



$l_2 = 0.035$

Own SOLHS 10.000 iterations

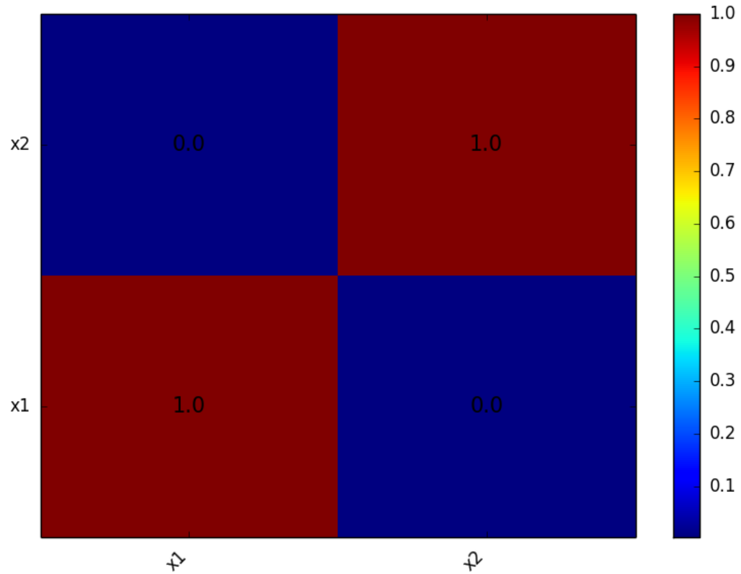


$l_2 = 0.041$

2. Previous results - Optimized latin hypercube sampling

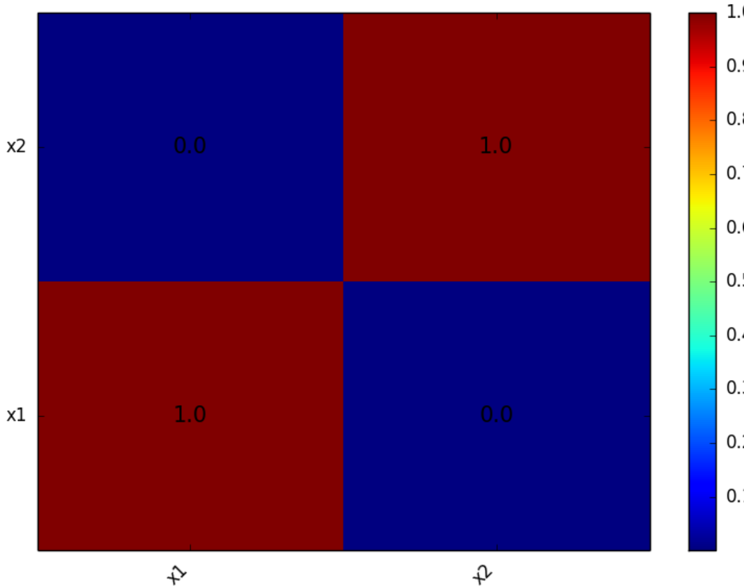
Comparison for Matlab N=64 n=2: orthogonality and layer 64 samples

Matlab OLHS criterion: orthogonality MCS (100.000) Iterations



ρ_{avg} =0.3E-2

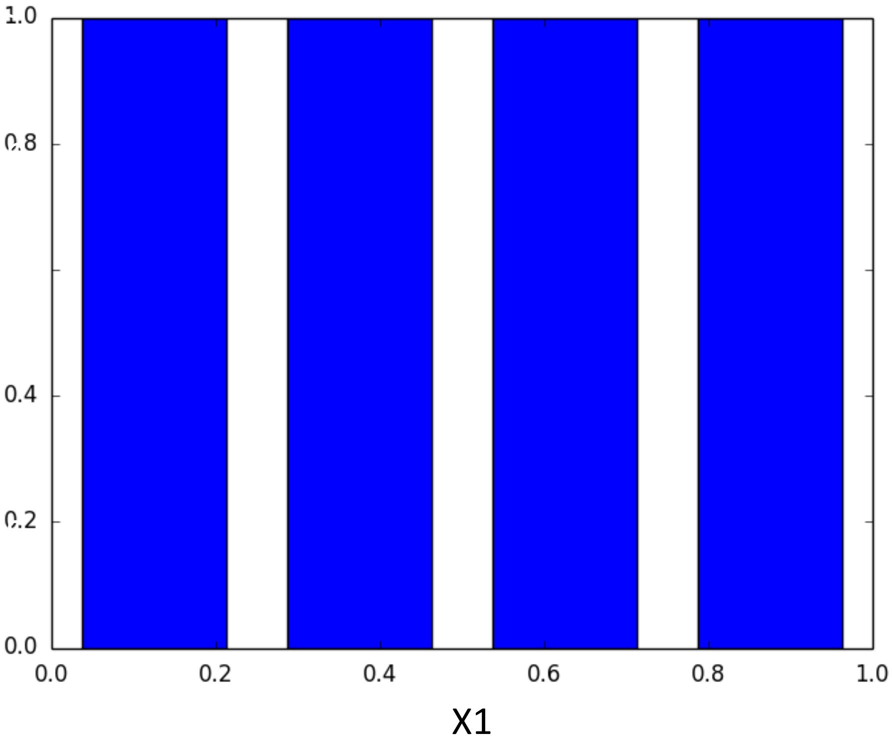
Own OLHS 10.000 iterations layer 64 samples

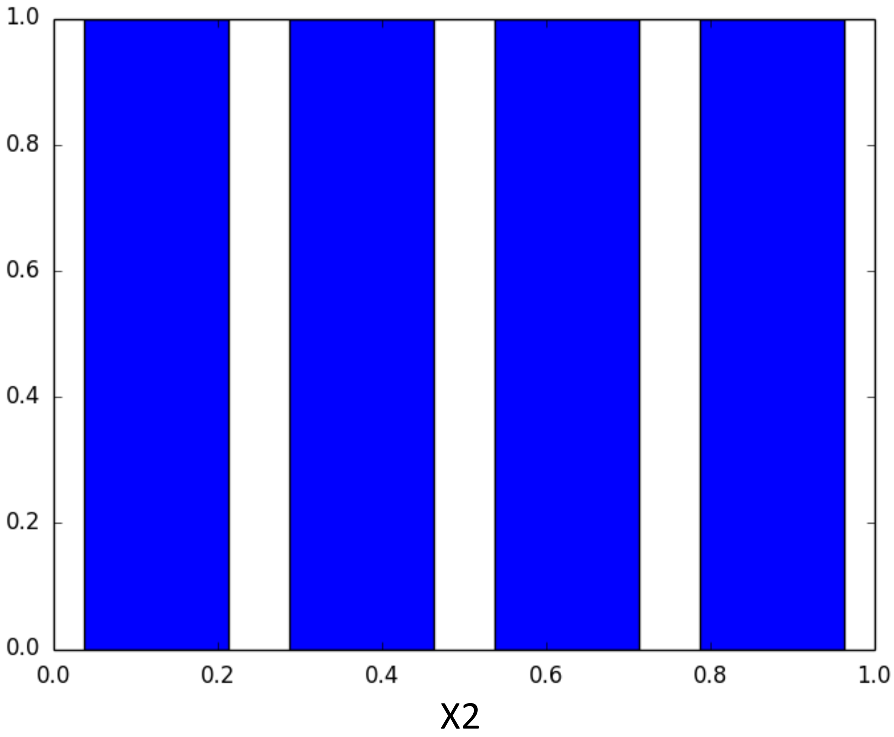


ρ_{avg} =0.12E-2

2. Previous results - Optimized latin hypercube sampling

Own SOLHS (10.000 Iterations)
64 samples uniformity check





2. Previous results - Advanced moving least square approximation

Standard moving least square approximation (MLS) [6] :

$\widetilde{y}(x) = p^T(x)a(x)$; Approximation of the results of the test points

$p(x) = [1, x_1, x_2, ..., x_n, x_1^2, x_2^2, ..., x_n^2, x_1x_2, ...x_nx_m]^T$; Polynomial basisfunction

$a(x) = (P^TW(x)P)^{-1}P^TW(x)y(x)$; Moving coefficients depending on testpoint x

$P = [p^T(x_1)p^T(x_2)...p^T(x_N)]$; Contains all polynomial basisfunction of the supportpoints

$y(x) = [y(x_1)y(x_2)...y(x_N)]$; Contains the results of the objective function for all support points

$W(x) = diag[w_1(x - x_1)w_2(x - x_2)...w_N(x - x_N)]$; Overall weighting matrix (diagonal matrix) contains for each testpoint a seperate weighting

$$w_N(\|x - x_N\|) = \begin{cases} e^{-(\frac{\|x - x_N\|}{D\alpha})^2} & \|x - x_N\| \leq D \\ 0 & \|x - x_N\| > D \end{cases}$$; Gaussian weighting function

$\alpha = \frac{1}{\sqrt{-log_{10}(0.001)}}$; Constant

$D =$ Self- selected parameter affecting model accuracy

2. Previous results - Advanced moving least square approximation

Advanced moving least square approximation (AMLS):

- A new concept of weighting, which not only includes a weighting for each test point, but also per variable. This means there exist not only one weighting matrix and D for the whole approximation but a weighting matrix and D per variable and if present for the crossterms.
- The D's are chosen through optimization with a particle swarm optimization algorithm.
- The optimization objective is the generalized coefficient of determination:

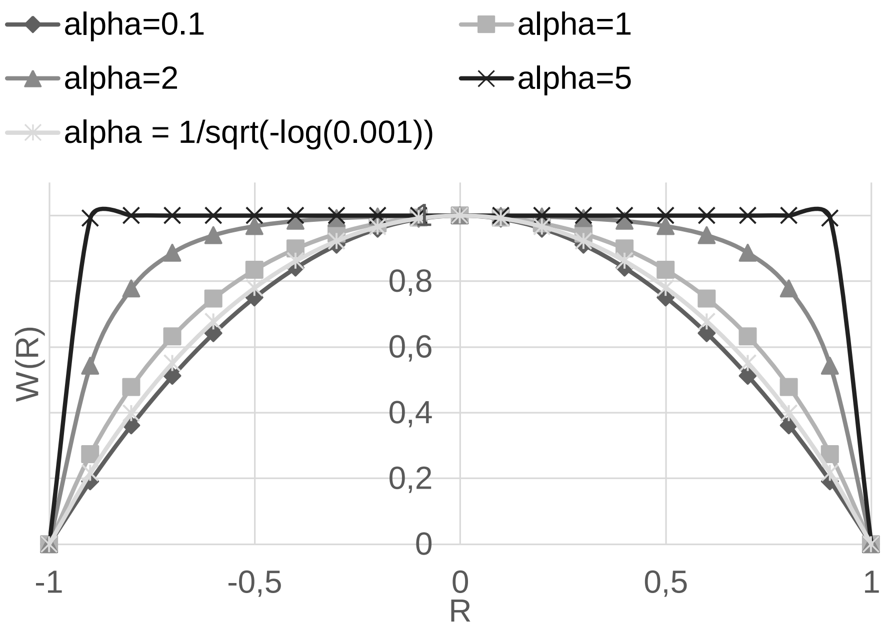
$$R^2 = \left(\frac{\sum_{N=1}^{k}(y^k - \mu_y)(\hat{y}^k - \mu_{\hat{y}})}{(N-1)\sigma_y \sigma_{\hat{y}}}\right)^2; 0 \leq R^2 \leq 1$$

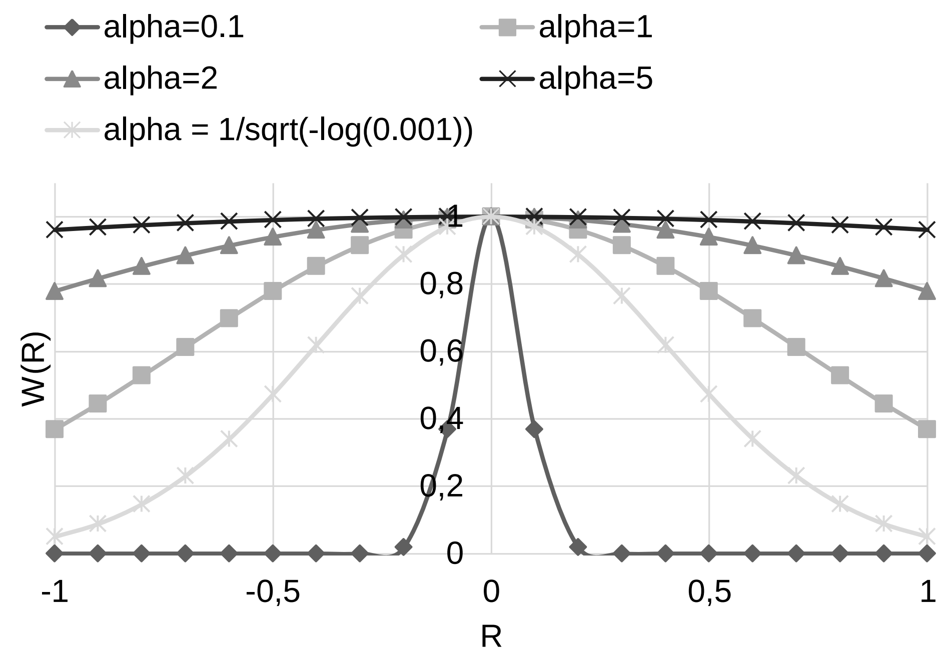
- Whereby the sample points are divided in equal subsets (0.2*N), so that every sample point is support- and testpoint, then the average R^2 is calculated (cross validation).
- α is not longer a constant, but a further optimization variable, in order to better fit the problem, rather than to change the weighting function. Therefore is also used a new formulation of the Gaussian weighting function:

$$w_N(\|x - x_N\|) = \begin{cases} \frac{e^{-(\alpha\|x - x_N\|/D)^2} - e^{-\alpha^2}}{(1 - e^{-\alpha^2})} \\ 0, \end{cases}$$

2. Previous results - Advanced moving least square approximation

$$w_N(\|x-x_N\|) = \begin{cases} \frac{e^{-(\alpha\|x-x_N\|/D)^2} - e^{-\alpha^2}}{(1-e^{-\alpha^2})} \\ 0, \end{cases} \qquad w_N(\|x-x_N\|) = \begin{cases} e^{-(\frac{\|x-x_N\|}{D\alpha})^2} \\ 0 \end{cases}$$





Shape of the Gaussian weight functions with different control parameter α with $R = \|x-x_N\|/D$

2. Previous results - Advanced moving least square approximation

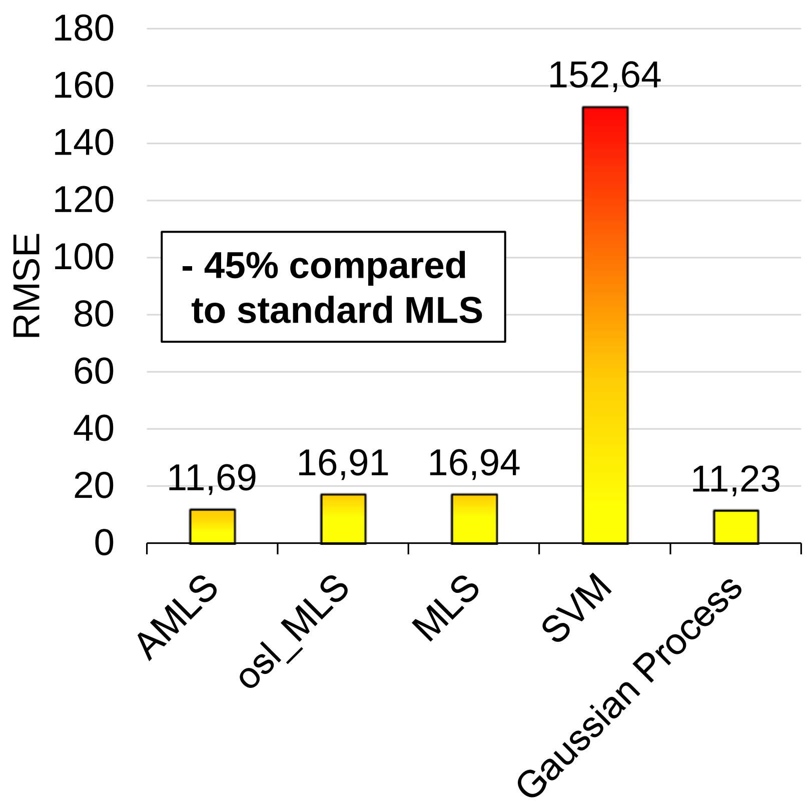
Benchmark of the AMLS:

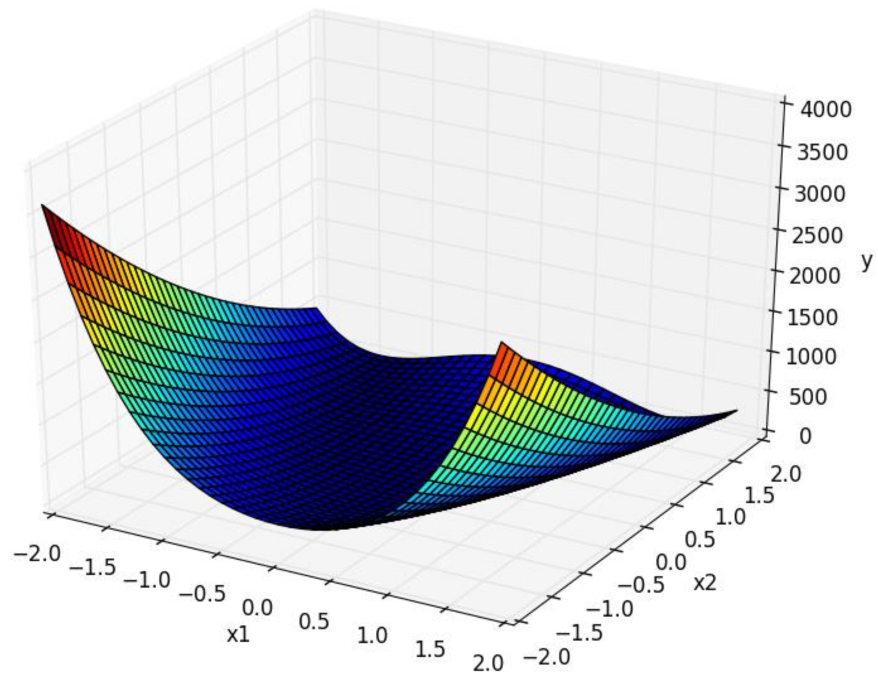
- Used Metamodels [9]: **MLS** (D is through optimization), **AMLS**, **super vector machine** [10], **Gaussian-process-regression** [11], **optiSLang MLS** (gaussaian weighting function, same D like MLS, all parameters have a quadratic basis polynom)

- Evaluation criterion: $RMSE = \sqrt{\frac{1}{N_{test}} \sum_{k=1}^{N_{test}} (y(x_k) - \hat{y}_e(x_k))^2}$
- $N_{Support} = 120;\ N_{test} = 500; n = 2-4$ (testpoints are not used, to create the metamodel)
- Testfunctions:
  - Rosenbrock: $y = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2; -2 \leq x_i \leq 2$
  - Normal PDF shape: $y = \frac{1}{1+x_1^4+5x_2^4+2x_2^2}; -3 \leq x_i \leq 3$
  - Sixhump Camelback: $y = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4; -2 \leq x_i \leq 2$
  - Own testfunction 1 $y = x_1 2 + x_2^2 + sin(x_3) + cos(x_4) + x_5^3; -5 \leq x_i \leq 5$
  - Own testfunction 2 $y = x_1 + x_1^2 x_2^2 + sin(x_3)^2 + cos(x_4 x_2) + x_5^3 x_1; -5 \leq x_i \leq 5$

2. Previous results - Advanced moving least square approximation

Rosenbrock: $y = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2; -2 \leq x_i \leq 2$

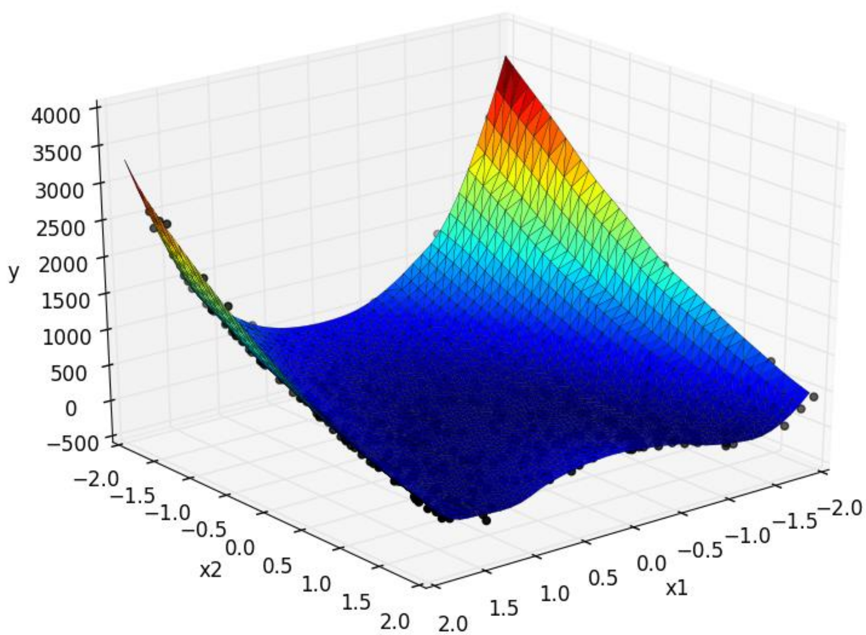




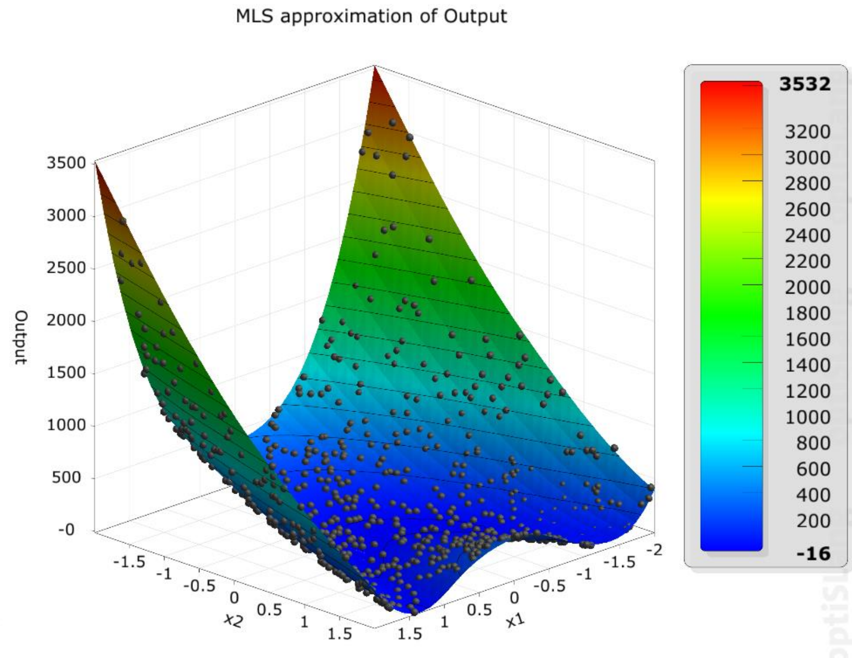
2. Previous results - Advanced moving least square approximation

Rosenbrock: $y = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2; -2 \leq x_i \leq 2$

Response surface AMLS

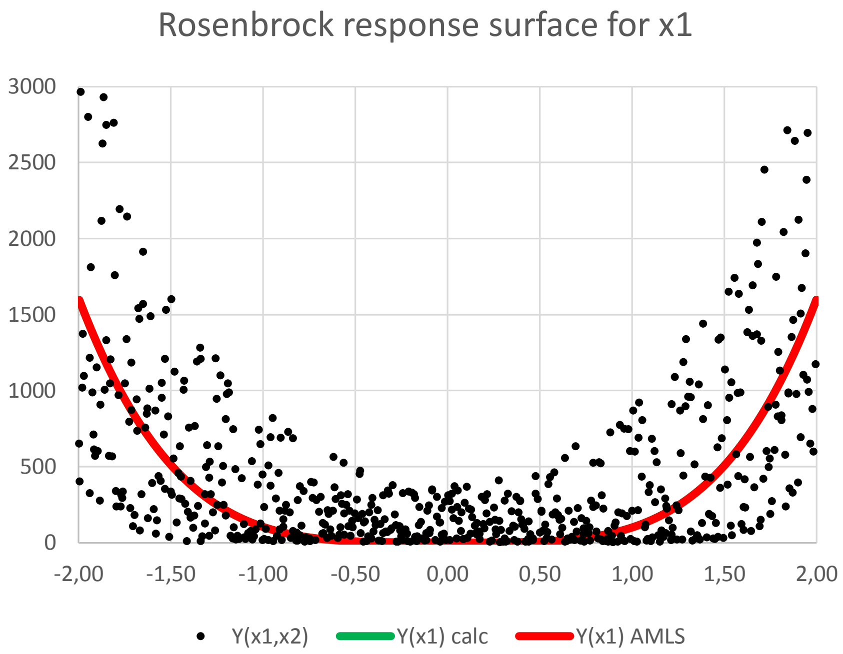


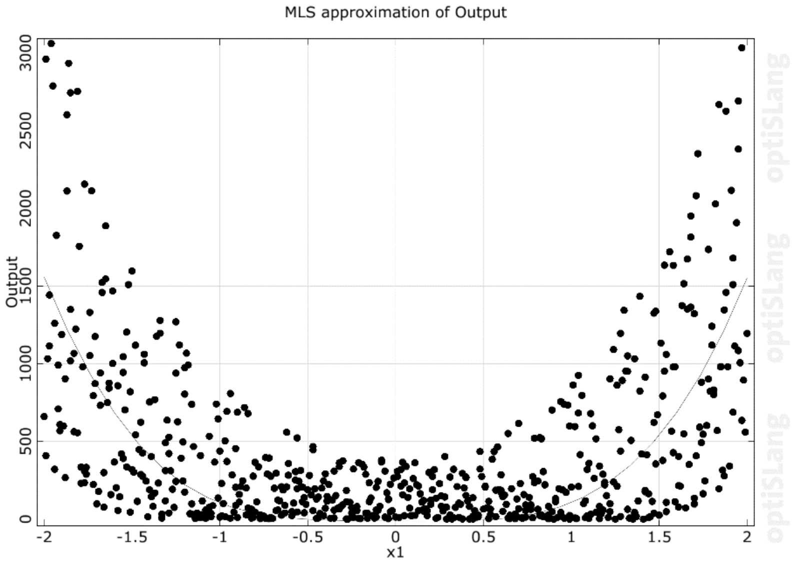
Response surface osl_MLS



2. Previous results - Advanced moving least square approximation

Rosenbrock: $y = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2; -2 \leq x_i \leq 2$

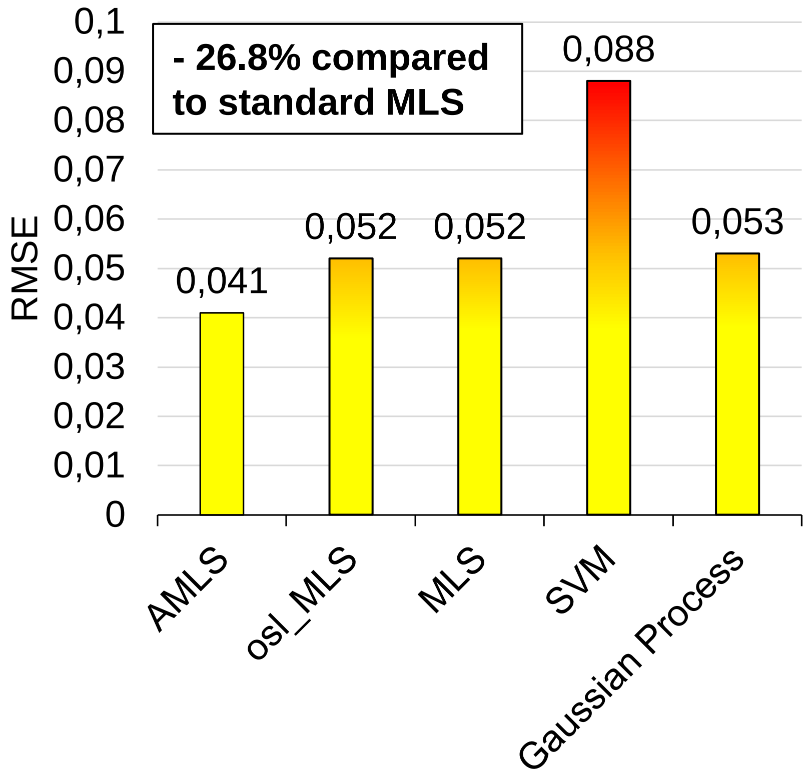


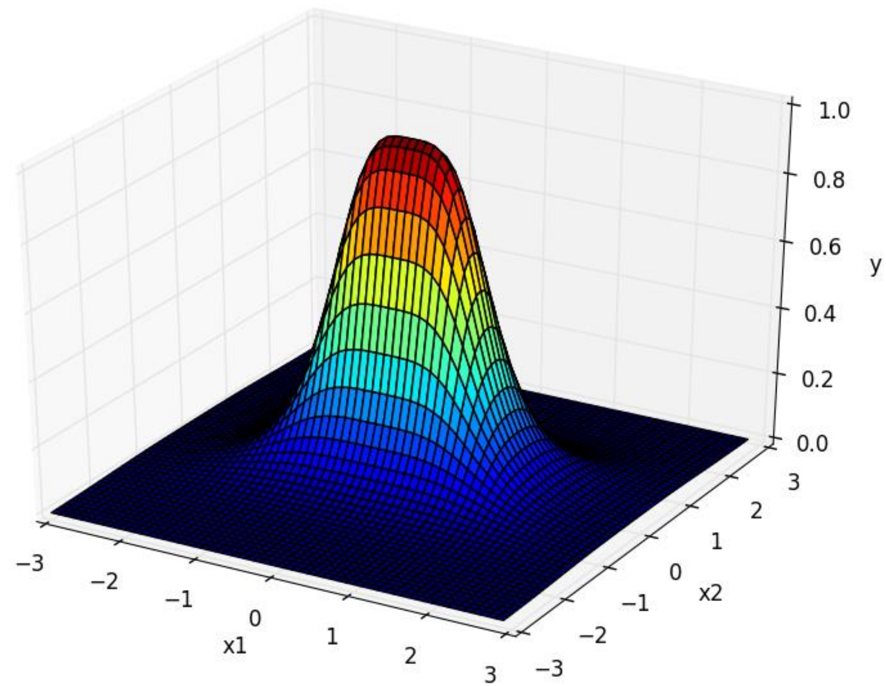


red = Approximation
green = Calculation

2. Previous results - Advanced moving least square approximation

Normal PDF shape: $y = \frac{1}{1+x_1^4+5x_2^4+2x_2^2}; -3 \leq x_i \leq 3$

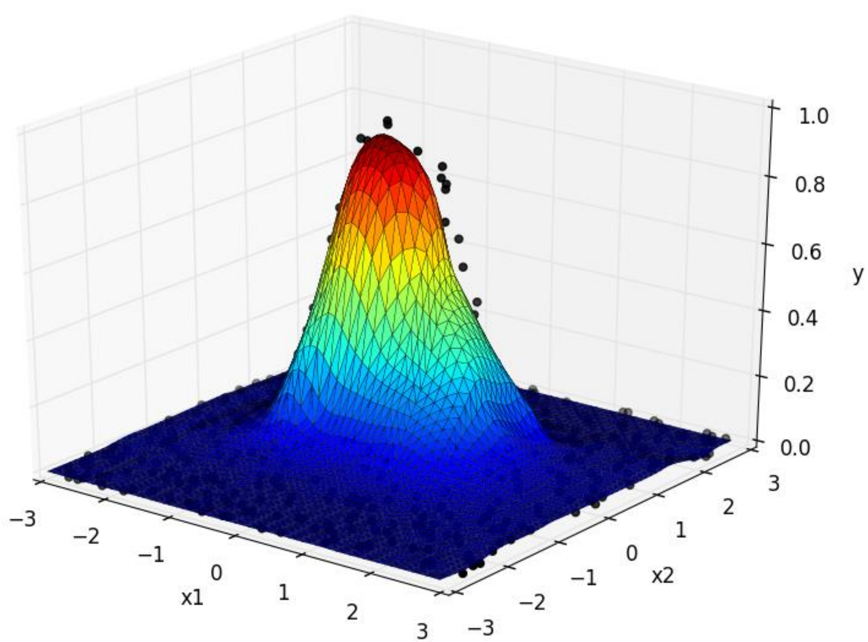




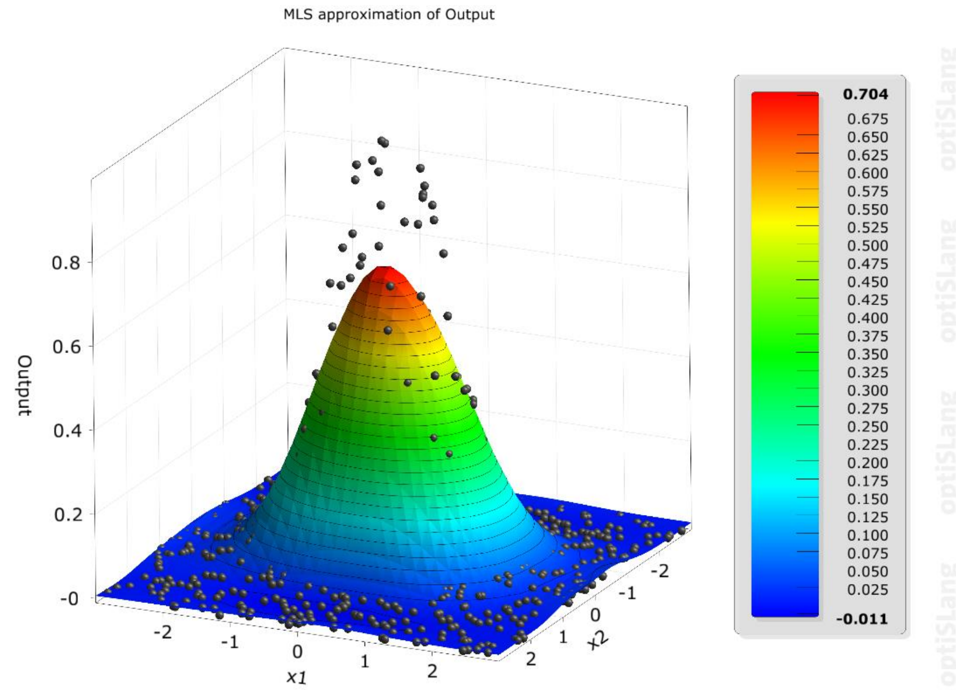
2. Previous results - Advanced moving least square approximation

Normal PDF shape: $y = \frac{1}{1+x_1^4+5x_2^4+2x_2^2}; -3 \leq x_i \leq 3$

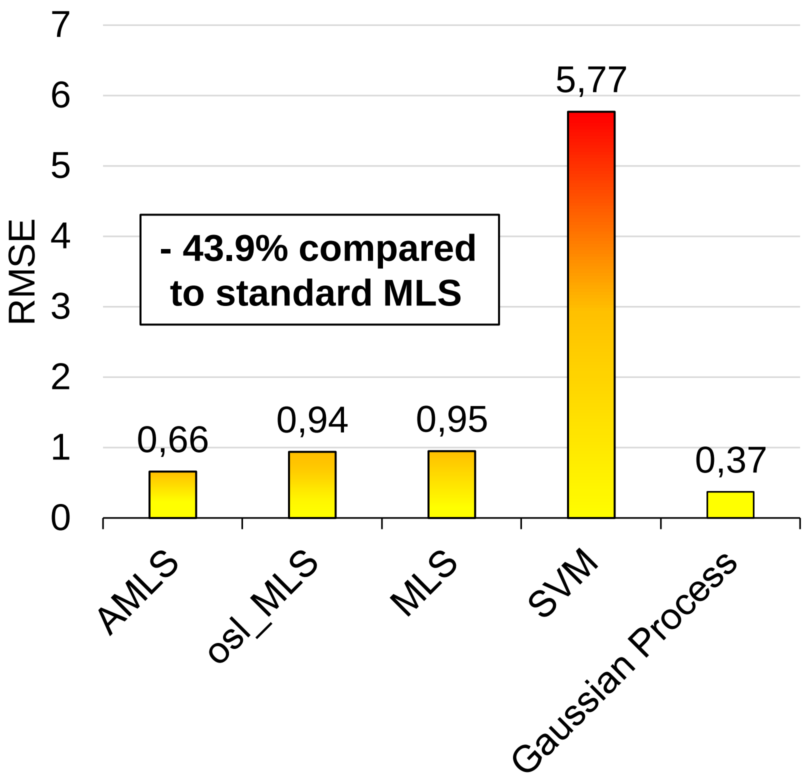
Response surface AMLS

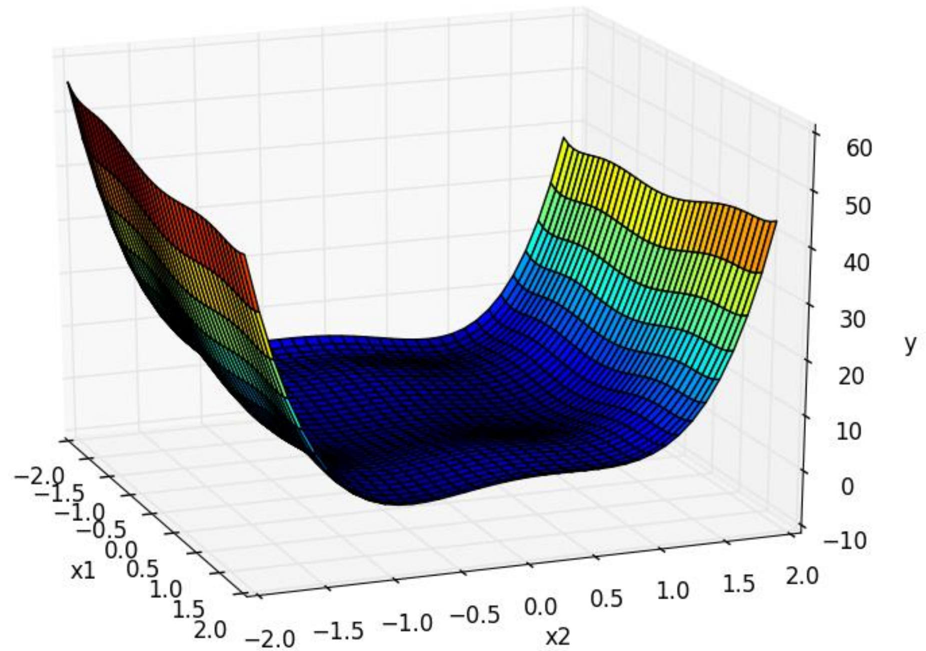


Response surface osl_MLS



Sixhump Camelback: $y = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4; -2 \leq x_i \leq 2$

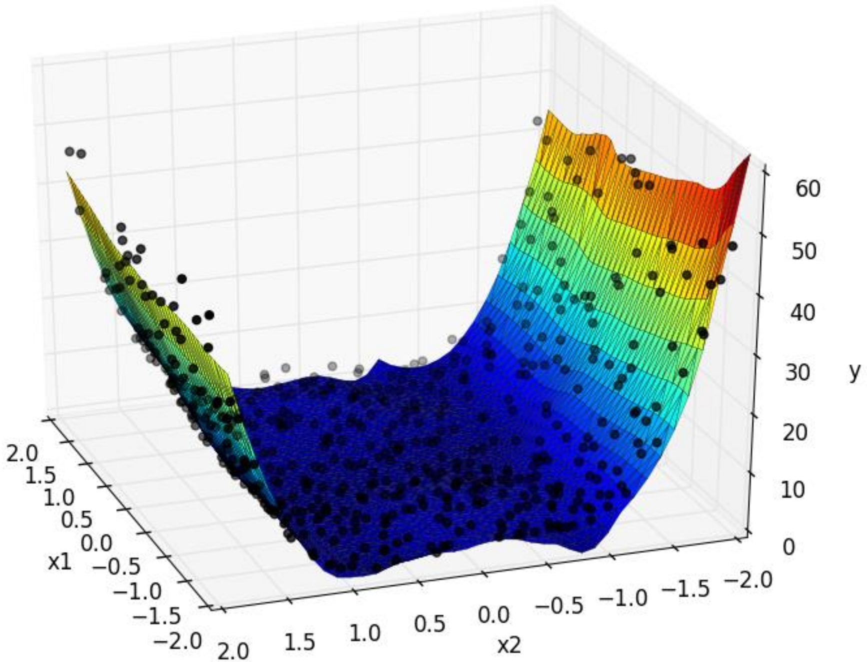




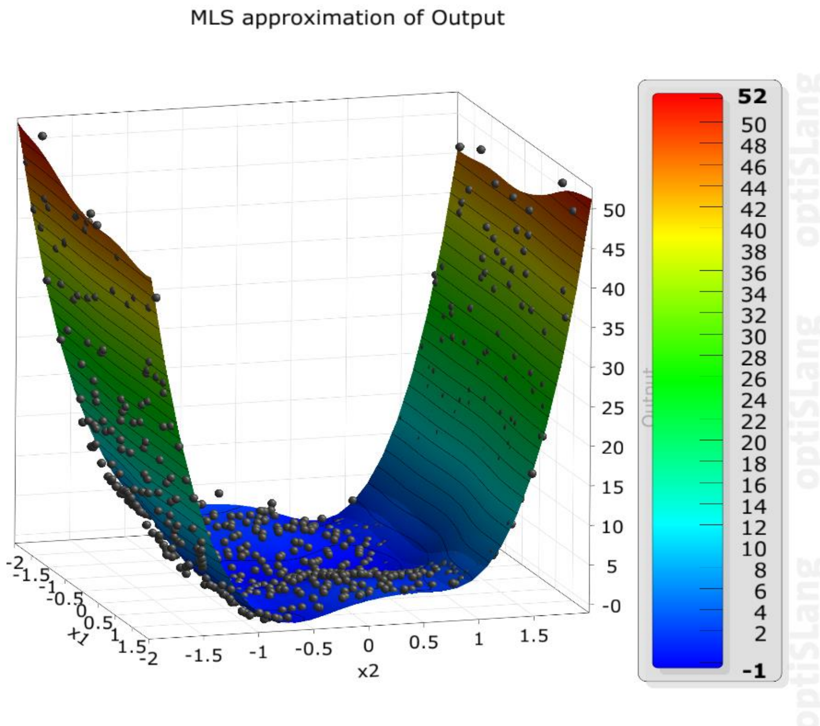
2. Previous results - Advanced moving least square approximation

Sixhump Camelback: $y = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4; -2 \leq x_i \leq 2$

Response surface AMLS

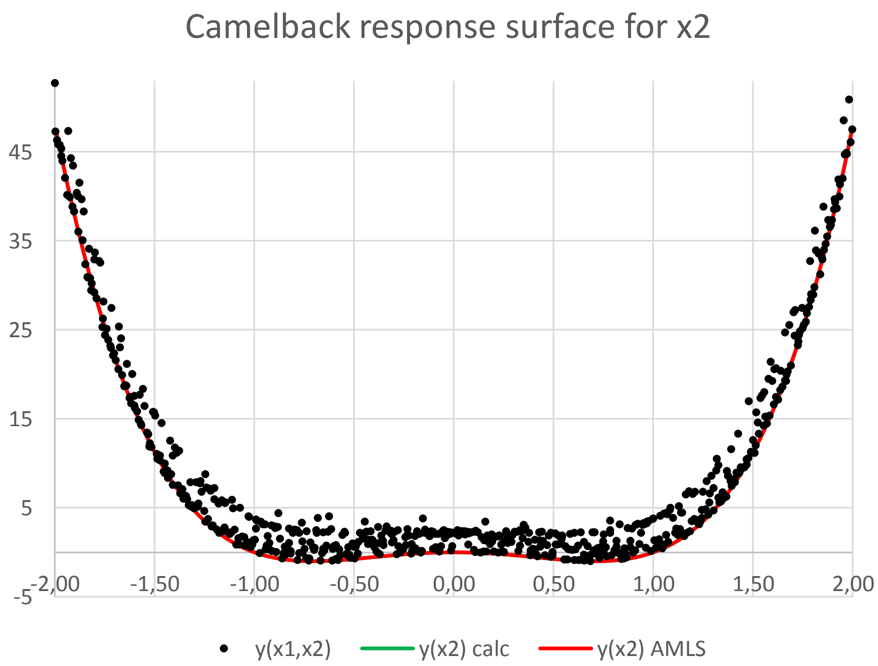


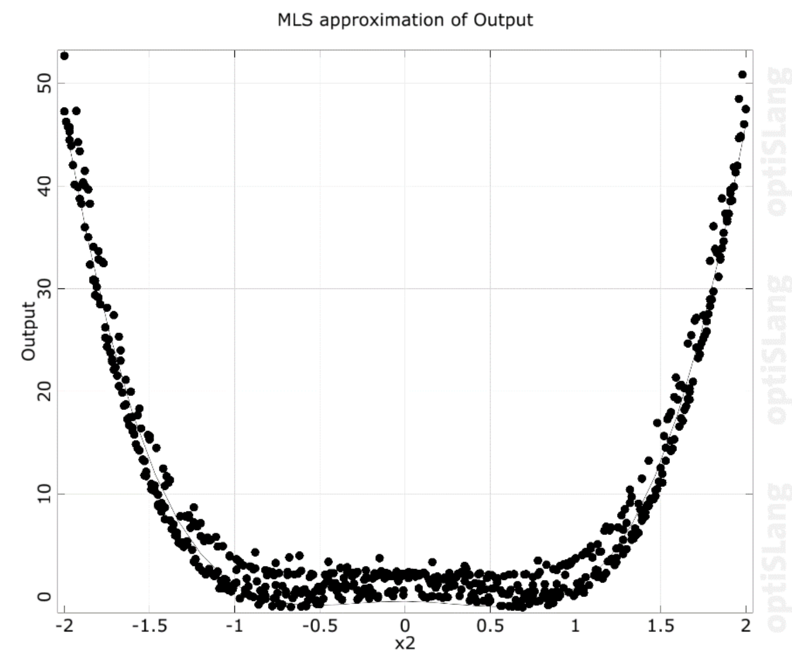
Response surface osl_MLS



2. Previous results - Advanced moving least square approximation

Sixhump Camelback: $y = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4; -2 \leq x_i \leq 2$

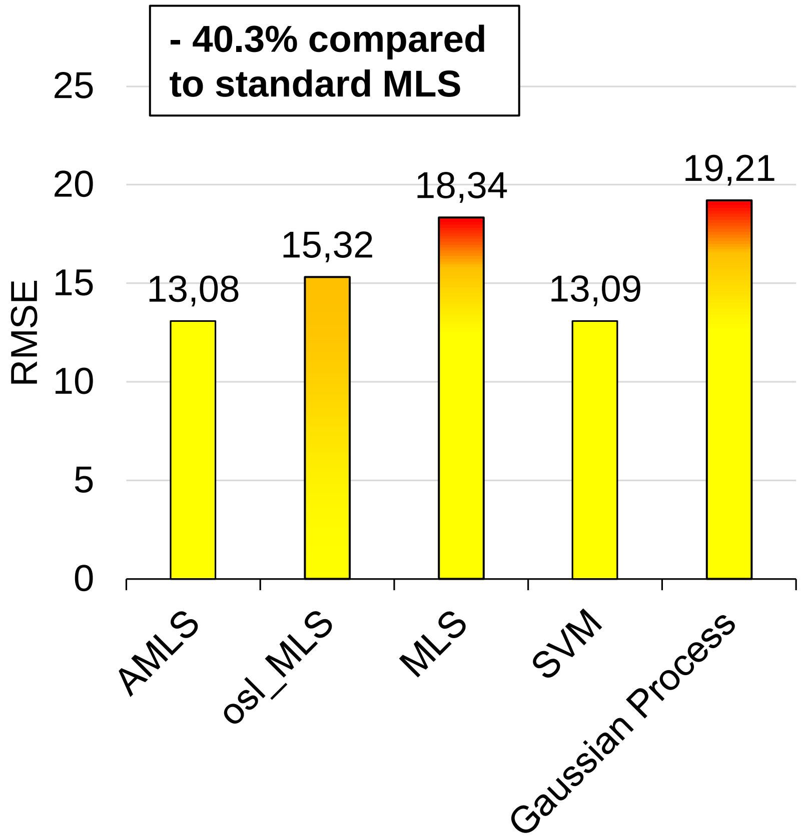


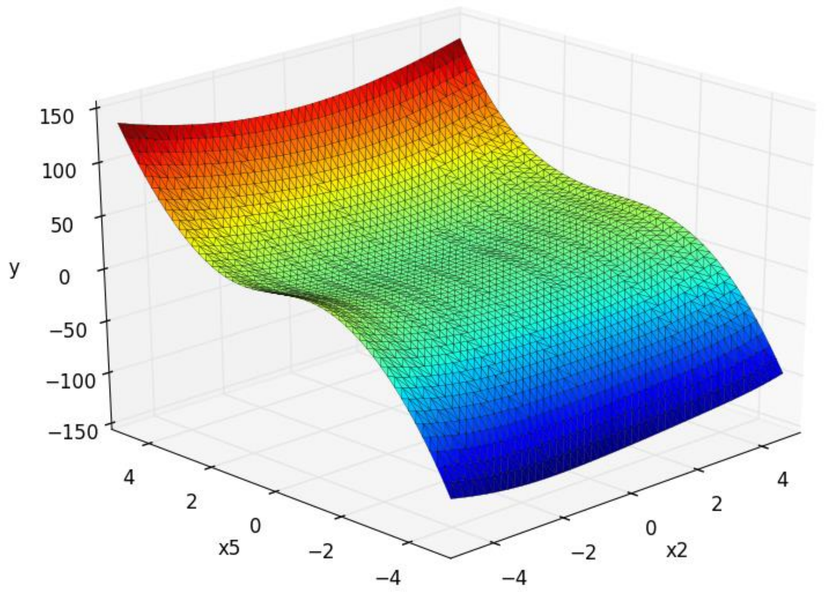


red = Approximation
green = Calculation

2. Previous results - Advanced moving least square approximation

Own testfunction 1: $y = x_1 2 + x_2^2 + sin(x_3) + cos(x_4) + x_5^3; -5 \leq x_i \leq 5$

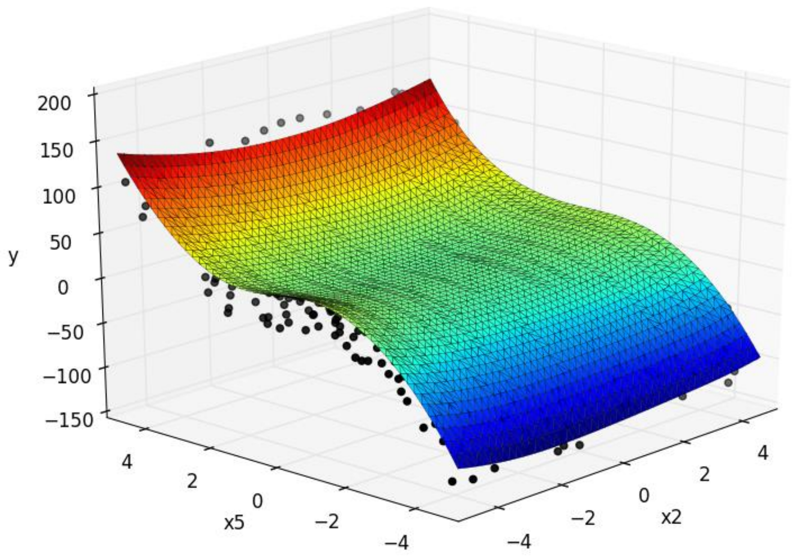




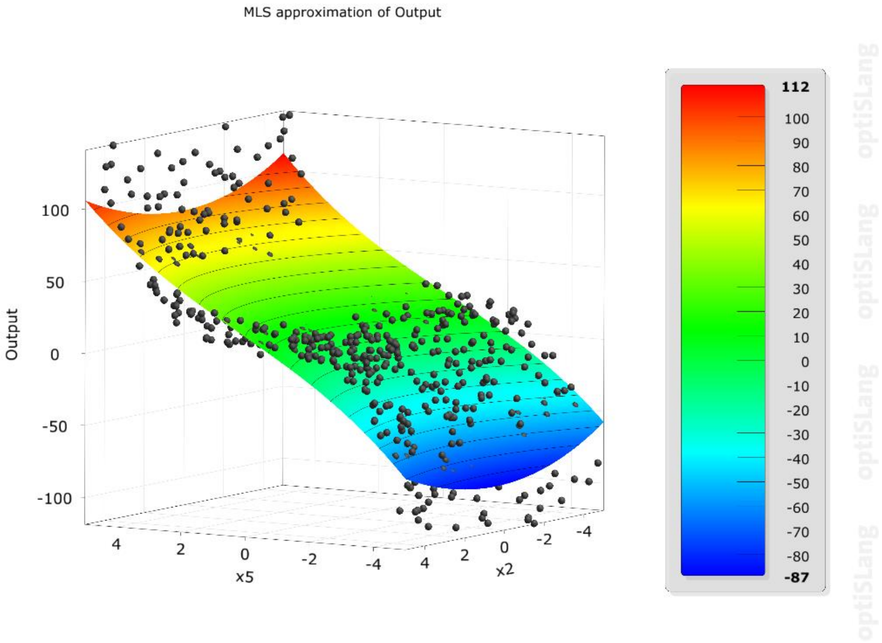
2. Previous results - Advanced moving least square approximation

Own testfunction 1: $y = x_1 2 + x_2^2 + sin(x_3) + cos(x_4) + x_5^3; -5 \leq x_i \leq 5$

Approximationspoints AMLS

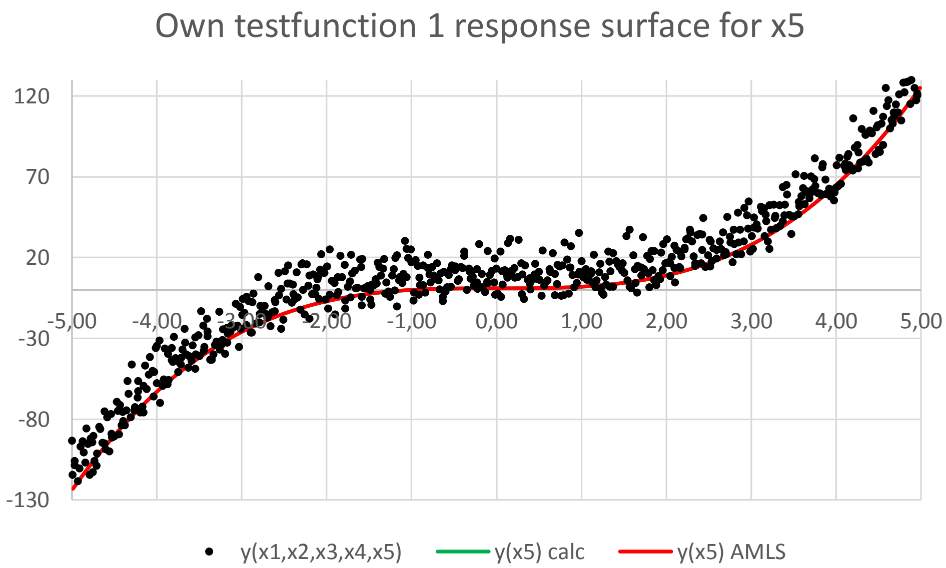


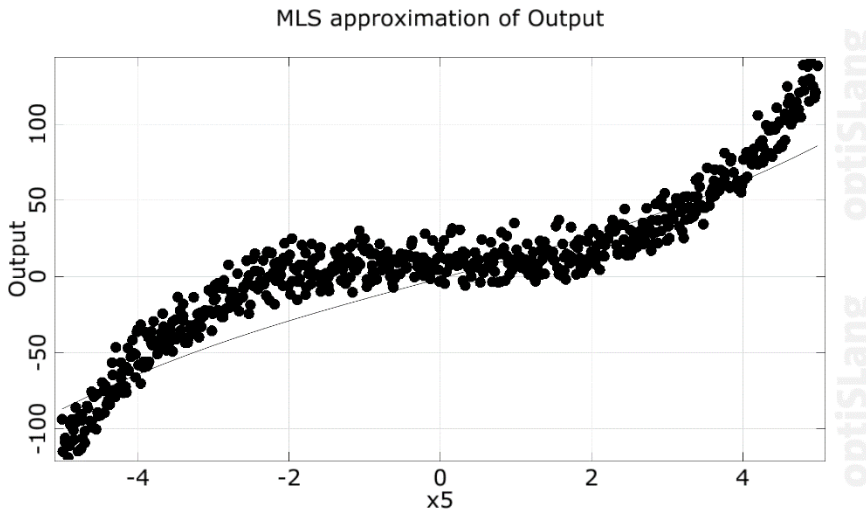
Approximationspoints osl_MLS (**x1, x3, x4 constant**)



2. Previous results - Advanced moving least square approximation

Own testfunction 1: $y = x_1 2 + x_2^2 + sin(x_3) + cos(x_4) + x_5^3; -5 \leq x_i \leq 5$

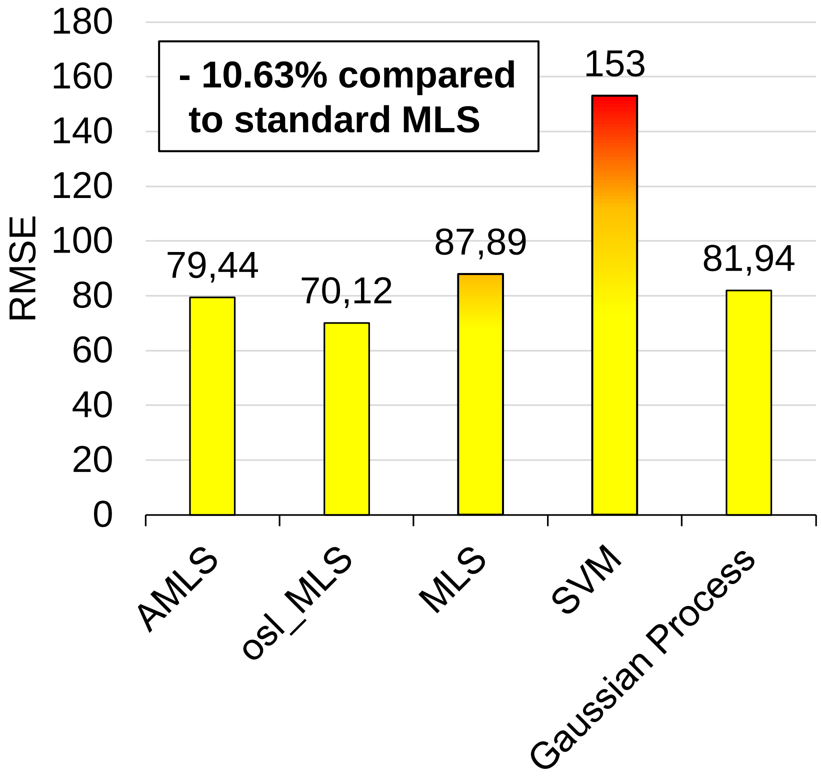


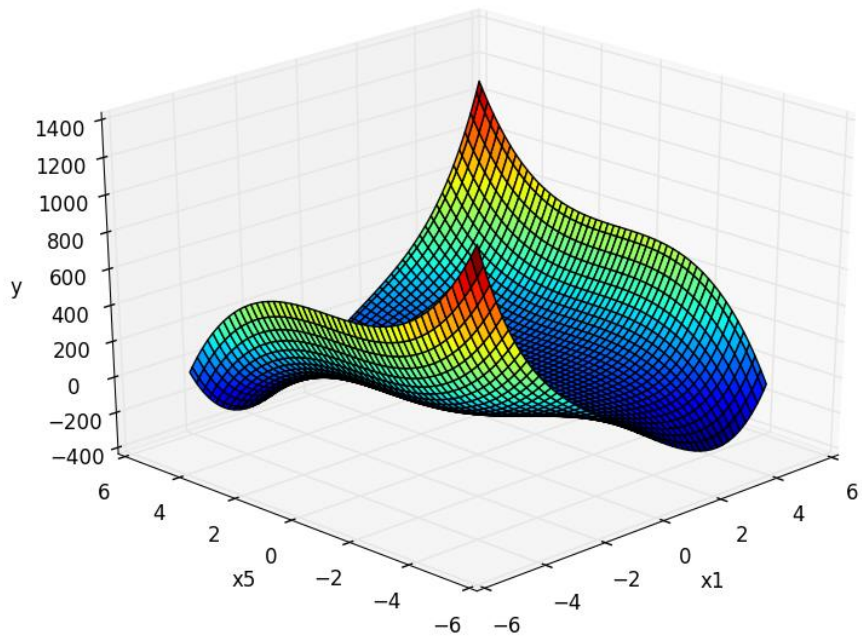


red = Approximation
green = Calculation

2. Previous results - Advanced moving least square approximation

Own testfunction 2: $y = x_1 + x_1^2 x_2^2 + sin(x_3)^2 + cos(x_4 x_2) + x_5^3 x_1; -5 \leq x_i \leq 5$

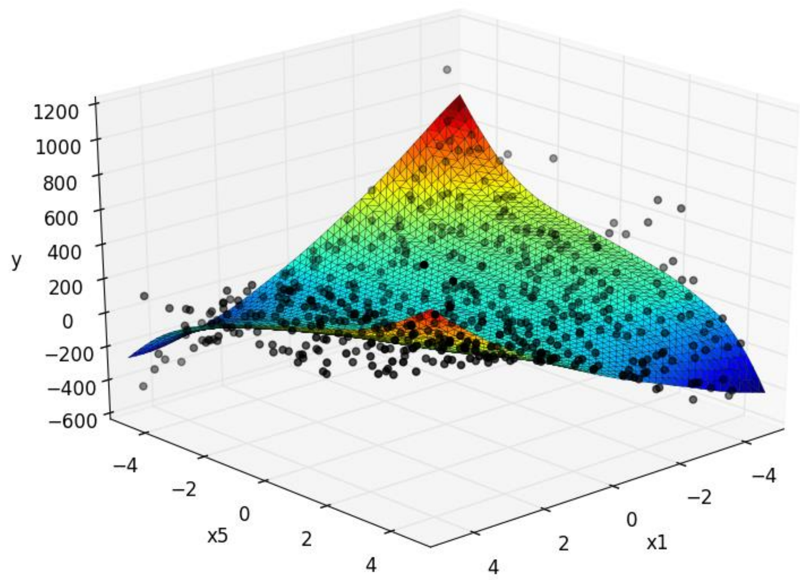




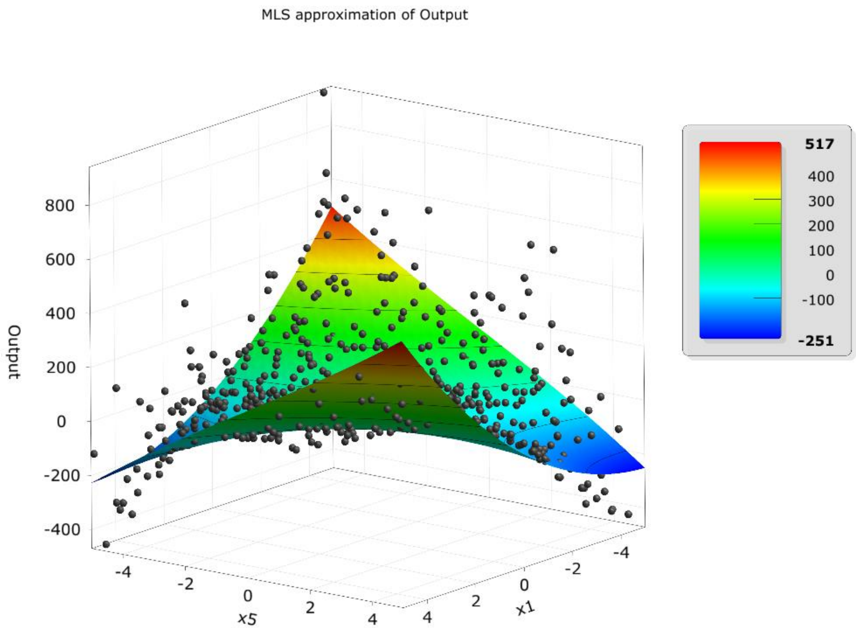
2. Previous results - Advanced moving least square approximation

Own testfunction 2: $y = x_1 + x_1^2 x_2^2 + sin(x_3)^2 + cos(x_4 x_2) + x_5^3 x_1; -5 \leq x_i \leq 5$

Response surface AMLS



Response surface osl_MLS (**x3, x4, x2 constant**)



2. Previous results - Advanced moving least square approximation

Conclusion:

- For problems without variables, which need to be filtered / constant, the AMLS showed better results as optiSLangs MoP with Gaussian Weighting and optimized D = MLS.
- For Problems with more than 2 variables, it occurs a advantage through filtered or constant parameters.
- The AMLS is for every testfunction better than the standard MLS with optimized D.

Plans:

- Our objective is to use metamodells for robust design optimization, therefore very accurate metamodels are necessary.
- It is planned to use optiSLangs variable reduction methods in combination with our metamodells to further improve the metamodels.

Outline

1. Introduction
    - Motivation
    - Objective
    - Planned application

2. Previous results
    - Optimized latin hypercube sampling
    - Advanced moving least square approximation

3. **Outlook**
    - Integration into optiSLang
    - Further developments

3. Outlook - Integration into optiSLang

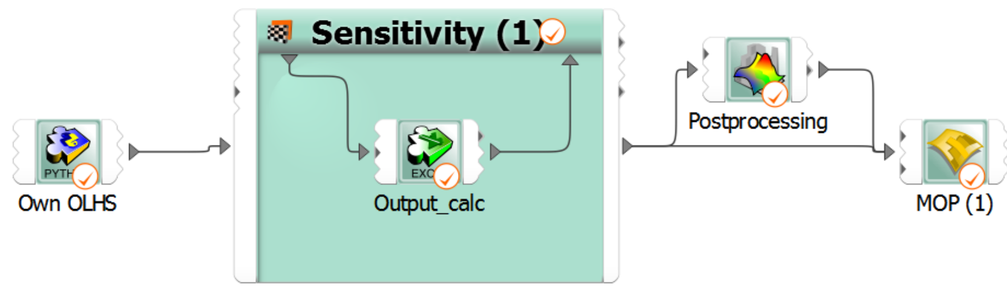
- Own developments in python can easliy be integrated into optiSLang v 4.

Advantages:

- Usage of existings methods / post processing and great process integration possibilities for other softwares, like ANSYS, ABAQUS, Excel.
- Extract data of different kinds of formats to work together. (ETK)
- No need to programm interfaces.
- Variables in Python can be decleared as parameters for optiSLang -> a lot of opportunities to work with own developments and methods / post processing of optiSLang.

```
##################################################
#Uebergabe Optislang
Xbest=np.transpose(Xbest)
parameter_names = ['x1', 'x2','x3','x4']
designs =        Xbest.tolist()

# convert py-List to PyOSDesignContainer
out_designs = PyOSDesignContainer()
for i in range(0, len(designs)):
    design = PyOSDesign()
    designpoint = PyOSDesignPoint()
    counter = 0
    for j in designs[i]:
        designpoint.add(parameter_names[counter],j)
        counter += 1
    design.SetParameters(designpoint)
    out_designs.push_back(design)
```



3. Outlook - Further developments

Developement of:

- AMLS (tests of different optimization algorithms for a high number of variables).
- Changing polynom coefficients regarding the variables (thanks to T. Most).
- Combination of sampling and estimination of the prognosis quality of the metamodell -> convergence analysis of the prognosis quality to sample the minimum number of designs.
- Chance-constraints stochastic multiobjective optimization on metamodels (optimization taking into account the probability of failure of the constraints).
- Simulationmodel for the different fractions of the hp/ip rotor developement.

Literature

- [1] Owen, A. (1994). Controlling correlations in Latin hypercube samples. J. Amer. Statist. Assoc. 89, 1517-1522.
- [2] Johnson, M., Moore, L. and Ylvisaker, D. (1990) Minimax and maximin distance designs. Journal of Statistical Planning and Inference 26 (2): 131-148
- [3] Morris, M. D., Mitchell, T. J. (1995), Exploratory designs for computer experiments. Journal of Statistical Planning and Inference 43 (3): 381-402
- [4] Kirkpatrick, S.; Gelatt Jr, C. D.; Vecchi, M. P. (1983). "Optimization by Simulated Annealing". Science 220 (4598): 671–680
- [5] web.stanford.edu
- [6] Lancaster P. and Salkauskas K. (1981). Surfaces Generated by Moving Least Squares Methods. Mathematics of Computation, Vol. 37, 141-158.
- [7] Eberhart, R. C. and J. Kennedy (1995).A new optimizer using particle swarm theory. pp. 39–43. Nagoya, Japan. IEEE Service Center Piscataway NJ: Proceedings of the Sixth International Symposium on Micromachine and Human Science.
- [8] www.quantcom.de/diemorphogenetischefelder.html

Literature

- [9] Pedregosa *et al (2011).*, Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825-2830.
- [10] C. Cortes, V. Vapnik (1995), Support-vector networks, Machine Leaming, 20, 273-297
- [11] C. E. Rasmussen, C. K. I. Williams (Ed. T Diettrich), (2006), Gaussian Processes for Machine Learning, MIT Press.

Appendix

1. Introduction - Objective

The advantages:

- Failure probabilities instead of safety factors
- Better understanding of parameters through sensitivity analysis
- Increased flexibility to perform changes within the parameter space
- The possibility to use extended application limits in contrast to the deterministic design
- Avoid interface conflicts and no need for expert knowledge in all areas / tools
- Optimal compromise solutions for the requirements

2. Previous results - Optimized latin hypercube sampling

Simulated Annealing (SA):

- Start is a random LHS with a design matrix D, where each column stands for a design point.

- By interchanging p (p < n) elements of two randomly selected columns -> D_{try} is a new design matrix.

- If $\psi(D_{try})$ is better than $\psi(D)$ D gets D_{try}. If $\psi(D_{try})$ is worse than $\psi(D)$ a random decision is made, if still D gets D_{try} or whether D_{try} is discarded. That D gets D_{try} will happen with the probability:

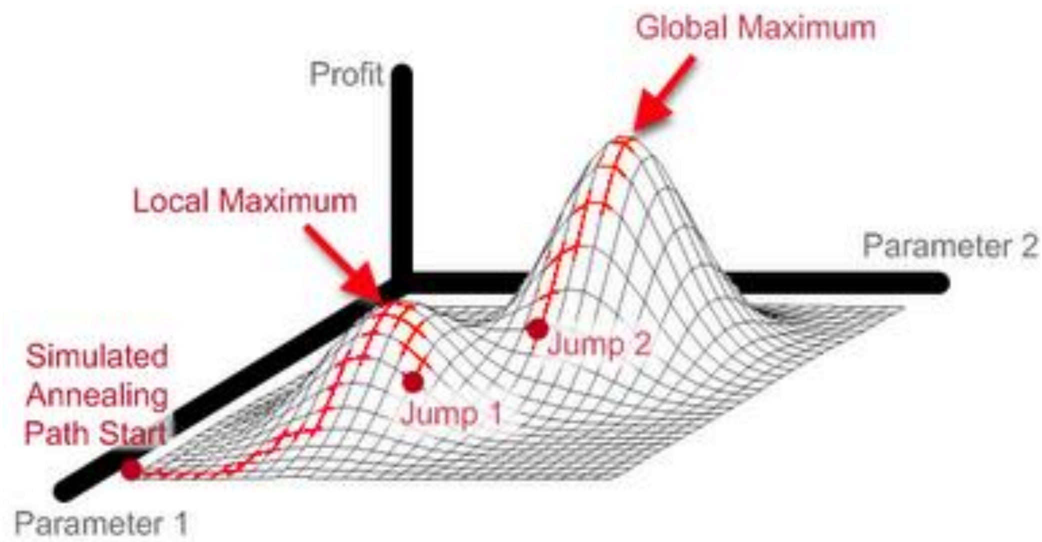
  $\pi = exp(-[\Psi(D_try) - \Psi(D)]/t)$, t=self-selected parameter.

- This random decision prevents that only a local minimum is found.

2. Previous results - Optimized latin hypercube sampling

- The result of the optimization is $\mathrm{D}_{\mathrm{best}}$.

- Start design matrix D is the best sampling regarding the overall ciriterion ψ out of 500 generated standard latin hypercube samplings

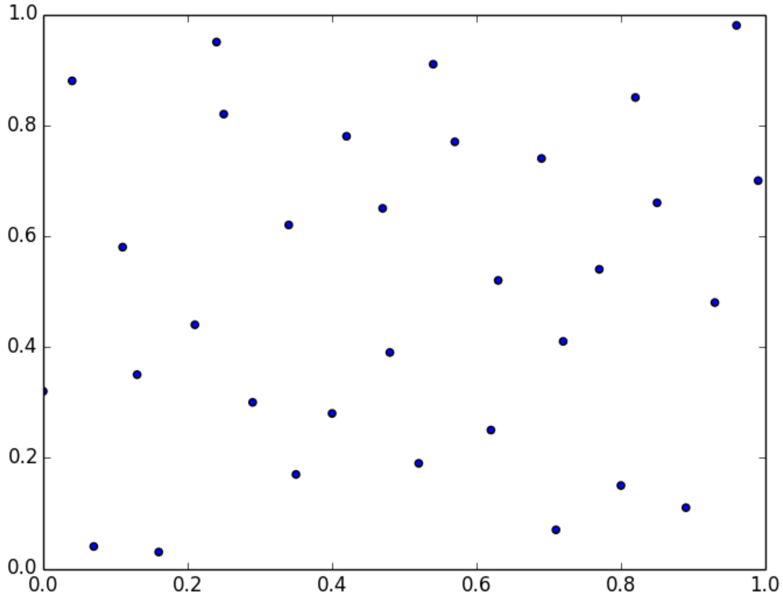


(source: [5])

2. Previous results - Optimized latin hypercube sampling

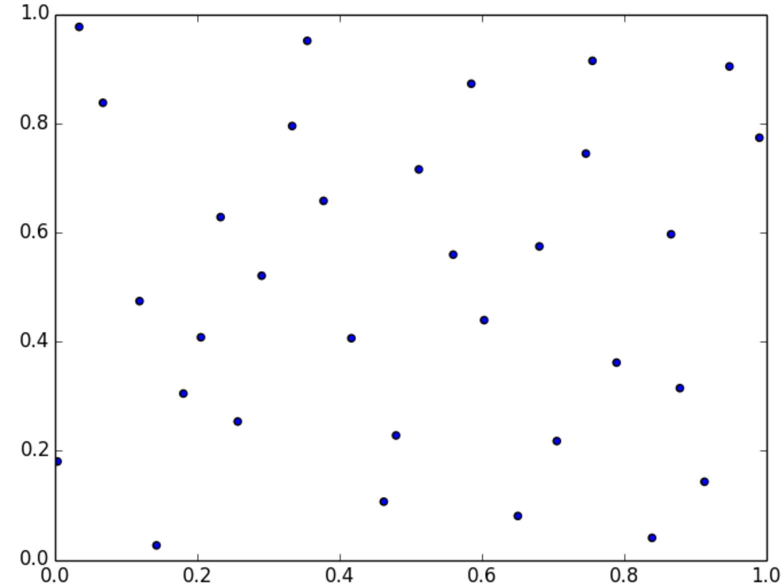
Comparison for Matlab N=32 n=2: maximin and SOHLS 32 samples

Matlab OLHS criterion: maximin
100.000 iterations



$l_2 = 0.064$

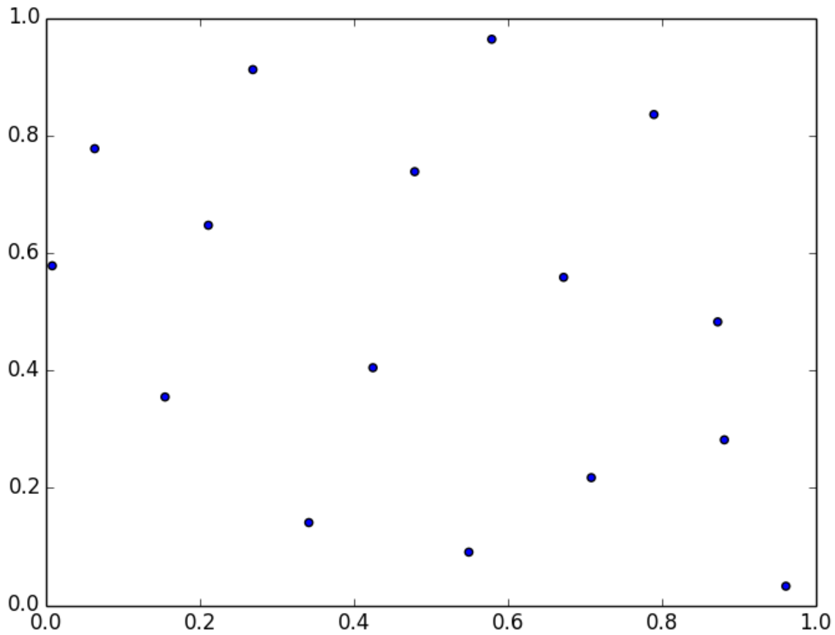
Own SOLHS 10.000 iterations



$l_2 = 0.065$

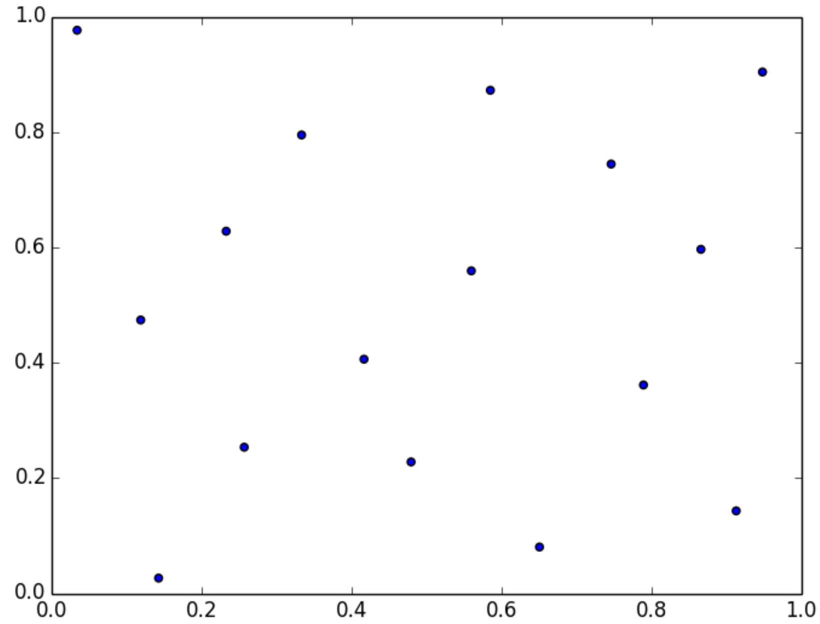
Comparison for Matlab N=16 n=2: maximin and SOHLS 16 samples

Matlab OLHS criterion: maximin
100.000 iterations



$l_2 = 0.130$

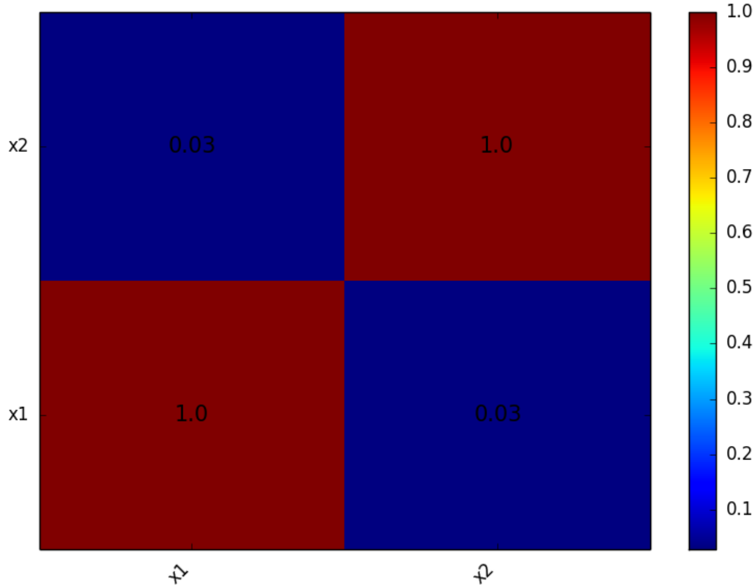
Own SOLHS 10.000 iterations



$l_2 = 0.133$

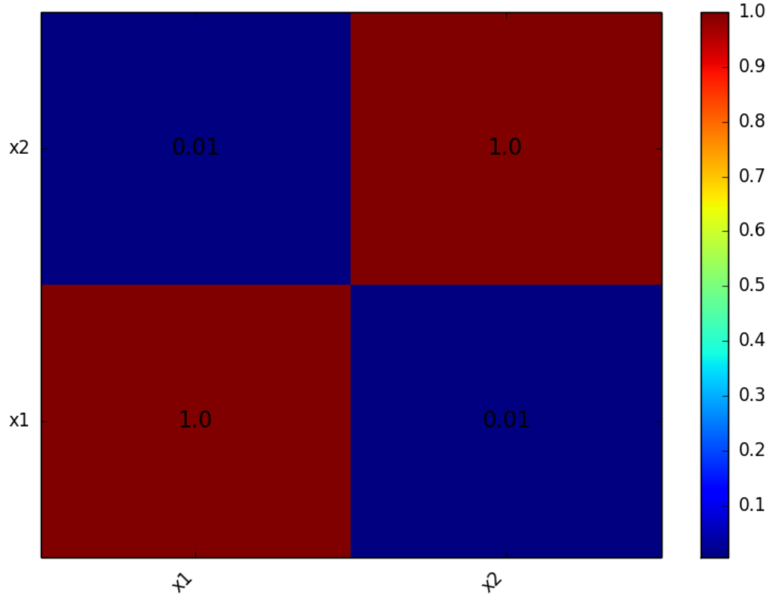
Comparison for Matlab N=16 n=2: orthogonality and layer 16 samples

Matlab OLHS criterion: orthogonality
100.000 iterations



$$\rho_{avg} = 0.0283$$

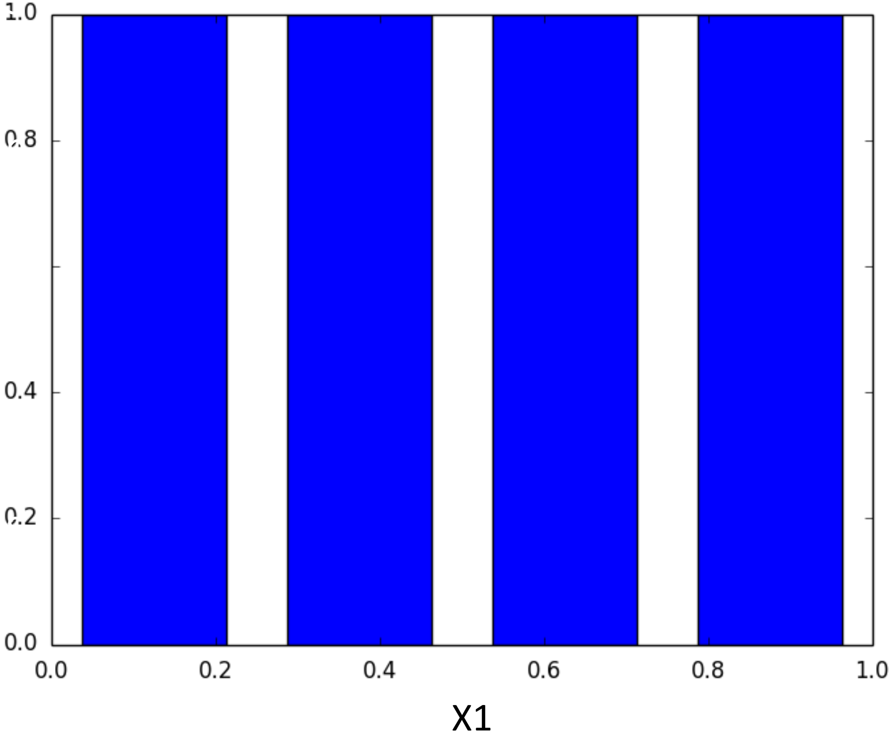
Own SOLHS 10.000 iterations

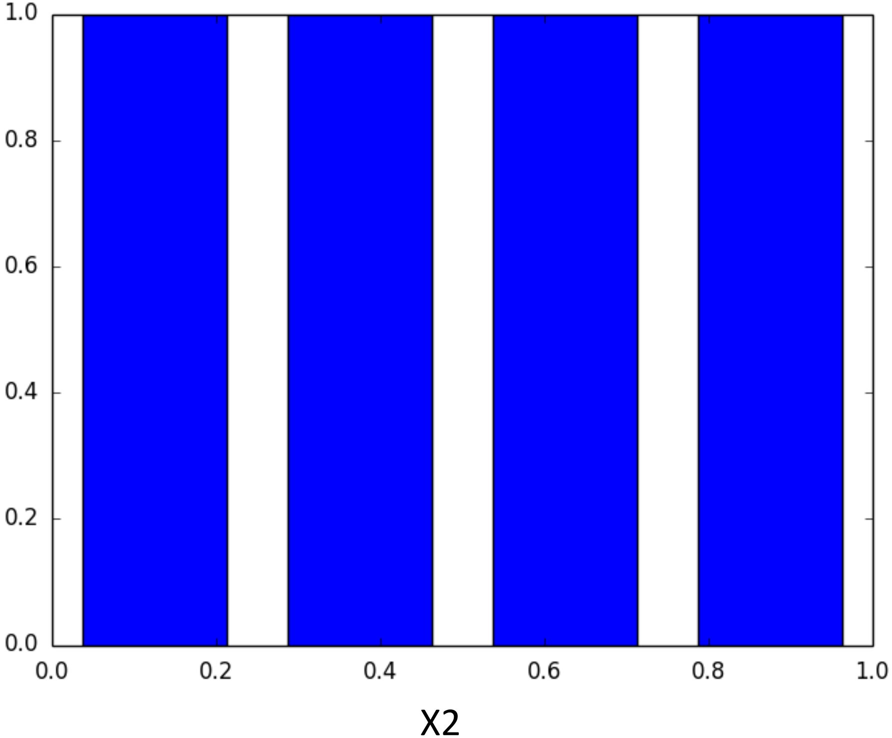


$$\rho_{avg} = 0.58\text{E} - 2$$

2. Previous results - Optimized latin hypercube sampling

Own SOLHS 10.000 iterations
16 samples uniformity check

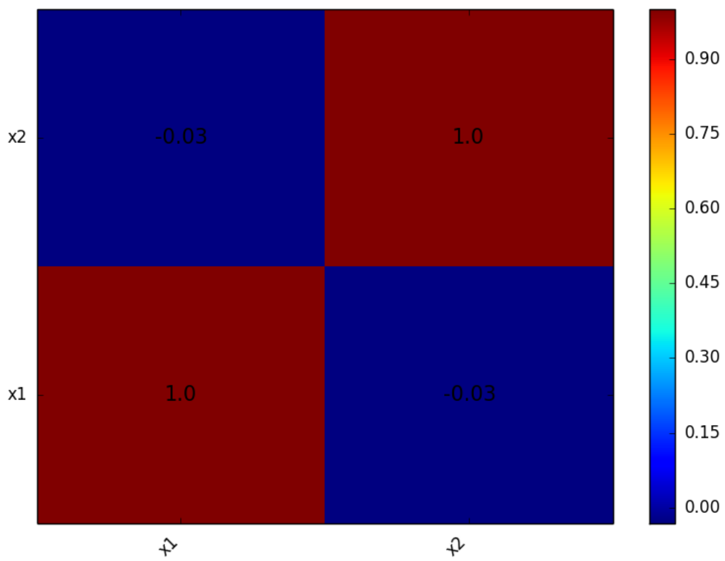




2. Previous results - Optimized latin hypercube sampling

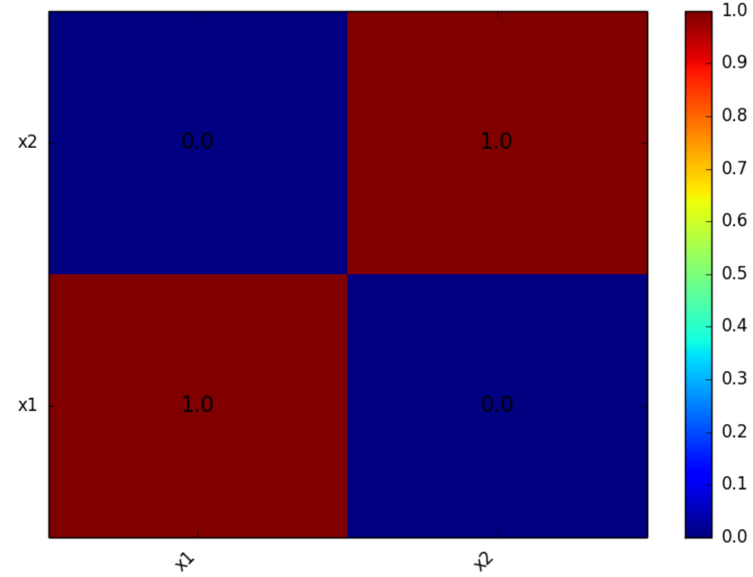
Comparison for Matlab N=32 n=2: orthogonality and layer 32 samples

Matlab OLHS criterion: orthogonality
100.000 iterations



$\rho_{avg} = 0.0315$

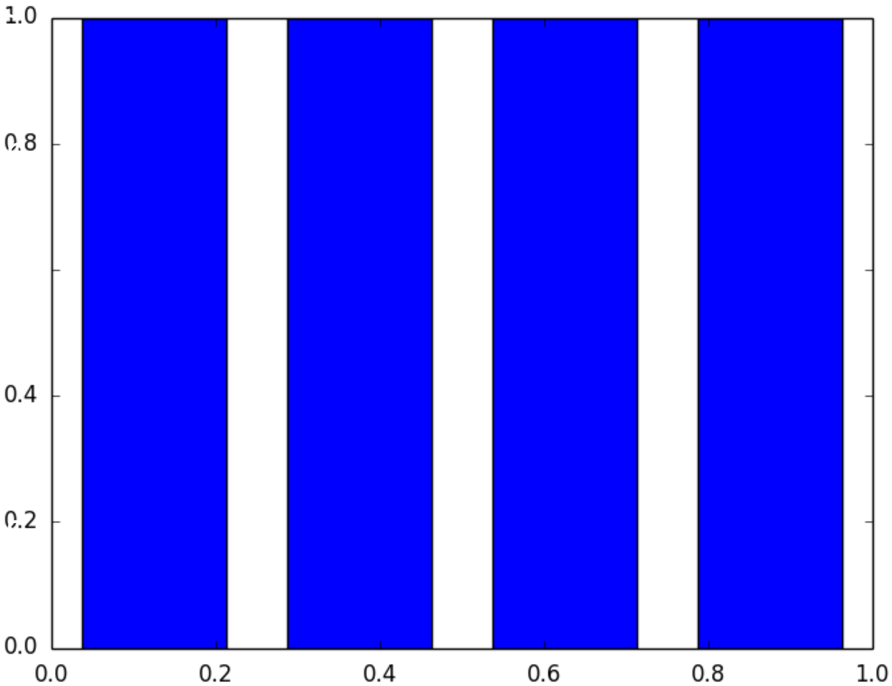
Own SOLHS 10.000 iterations



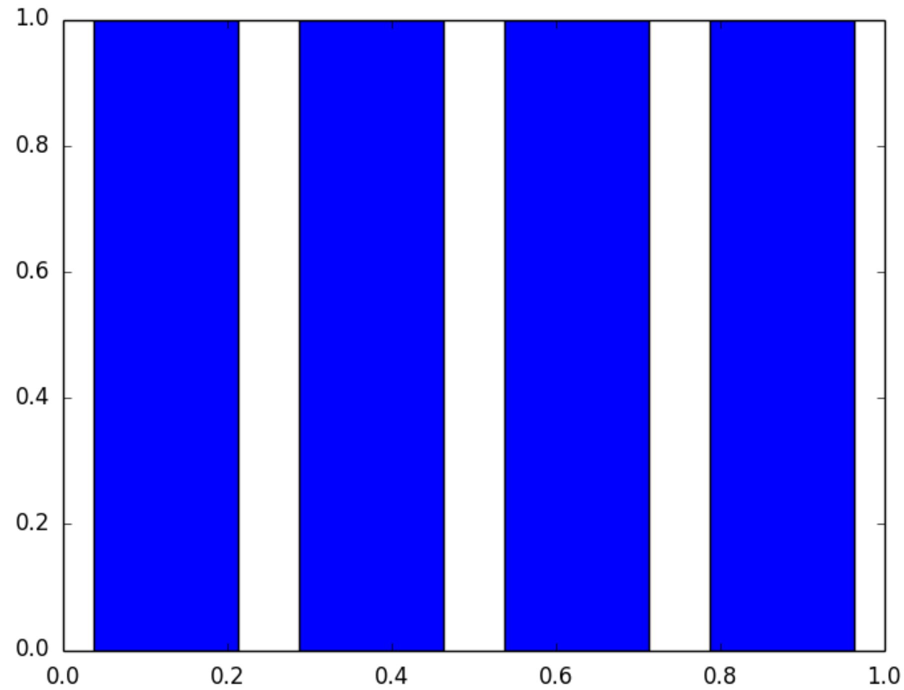
$\rho_{avg} = 0.6\mathrm{E}-3$

2. Previous results - Optimized latin hypercube sampling

Own SOLHS (10.000 Iterations)
32 samples uniformity check



X1



X2

2. Previous results - Advanced moving least square approximation

Particle Swarm algorithm [7]:

- An initial population ("swarm") of possible candidates (“particle") move through the parameter space .
- The direction of the "particle" is guided by the knowledge of its best position (local optimum), and the best known position of the “swarm leader“ (global optimum).
- If new better positions are discovered, they are used to steer the “swarm".
- This process is repeated a certain number of iterations until an optimal solution is found.

Advantages:

- Suitable for multiobjective optimization.
- High number of input variables possible.

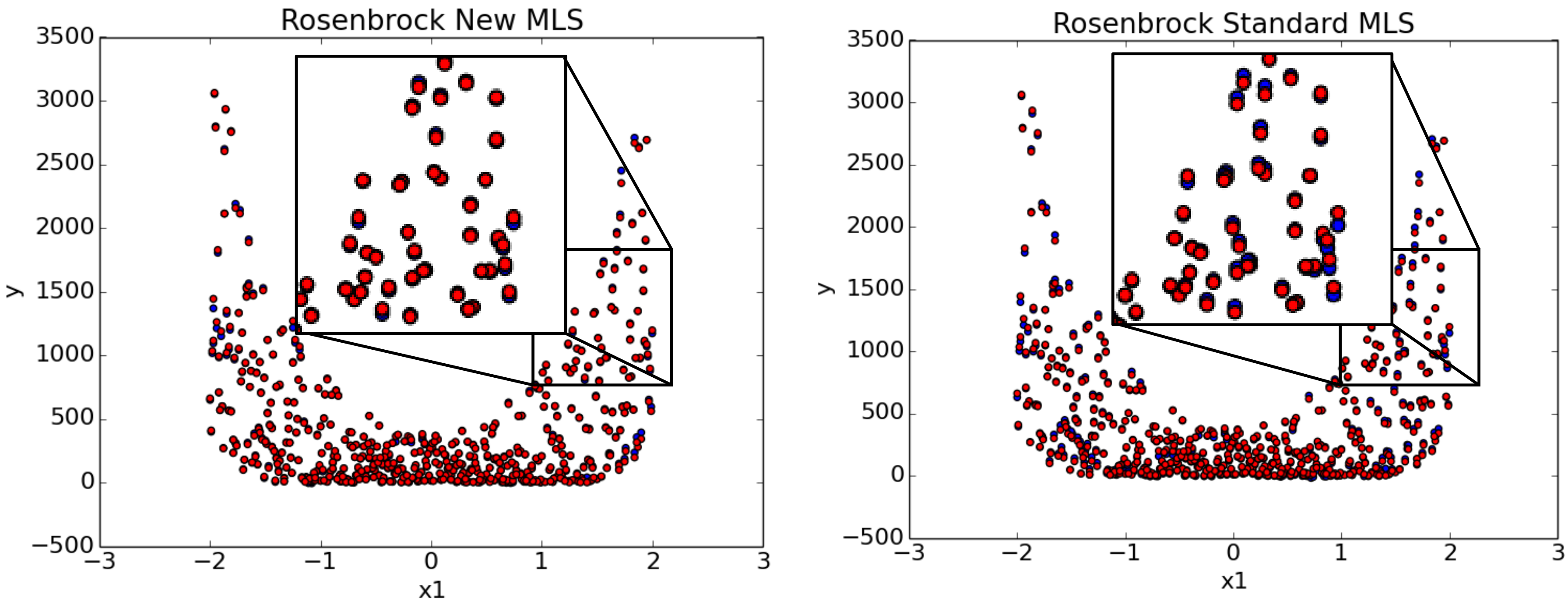
Disadvantage:

- Suitable coefficients for the “swarm" to determine behavior (8 coefficients).

(source: [8])

2. Previous results - Advanced moving least square approximation

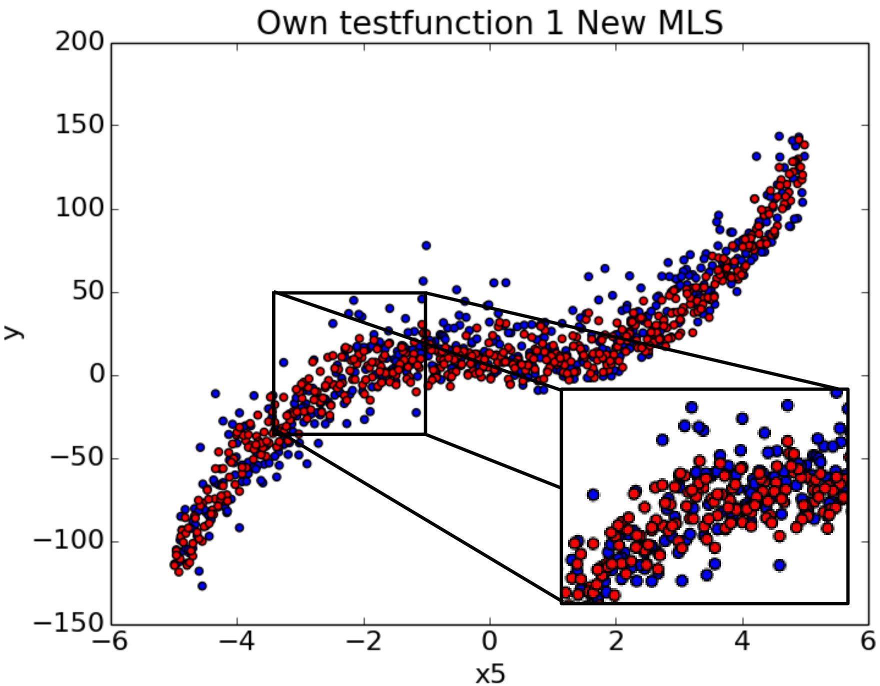
Rosenbrock: $y = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2; -2 \leq x_i \leq 2$

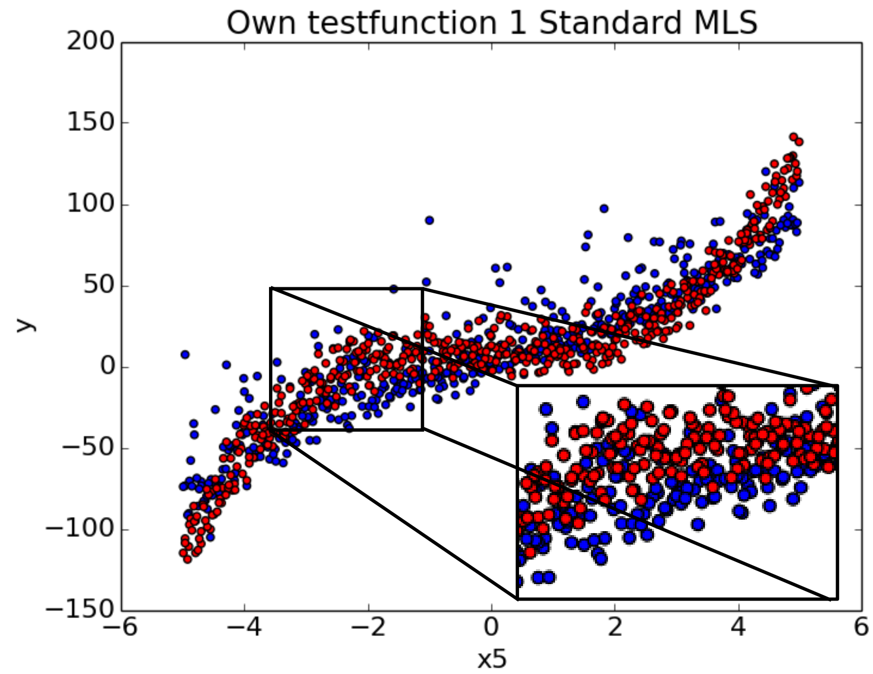


Blue = Approximation
Red = Calculation

2. Previous results - Advanced moving least square approximation

Own testfunction 1: $y = x_1 2 + x_2^2 + sin(x_3) + cos(x_4) + x_5^3; -5 \leq x_i \leq 5$

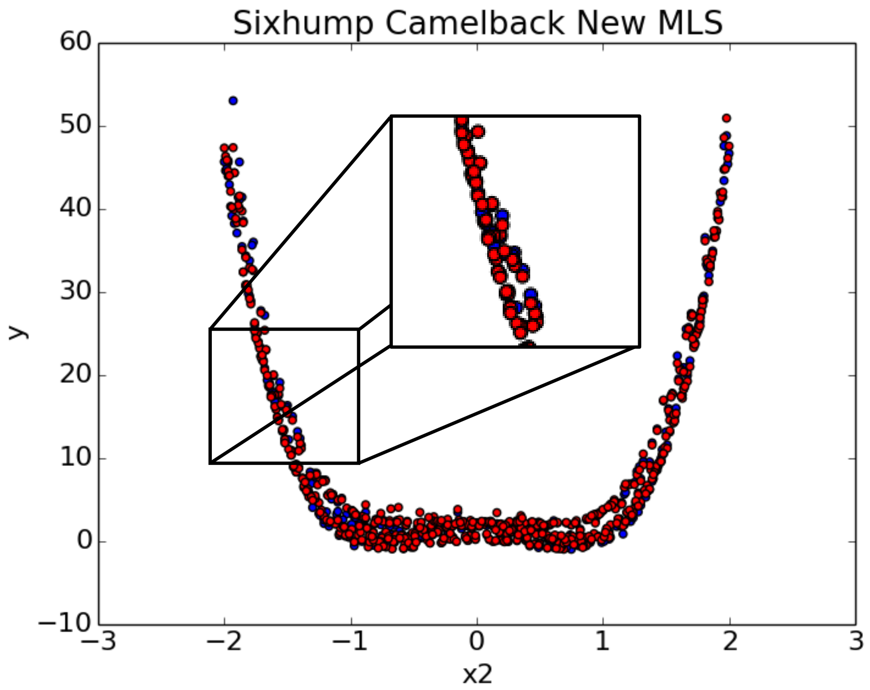


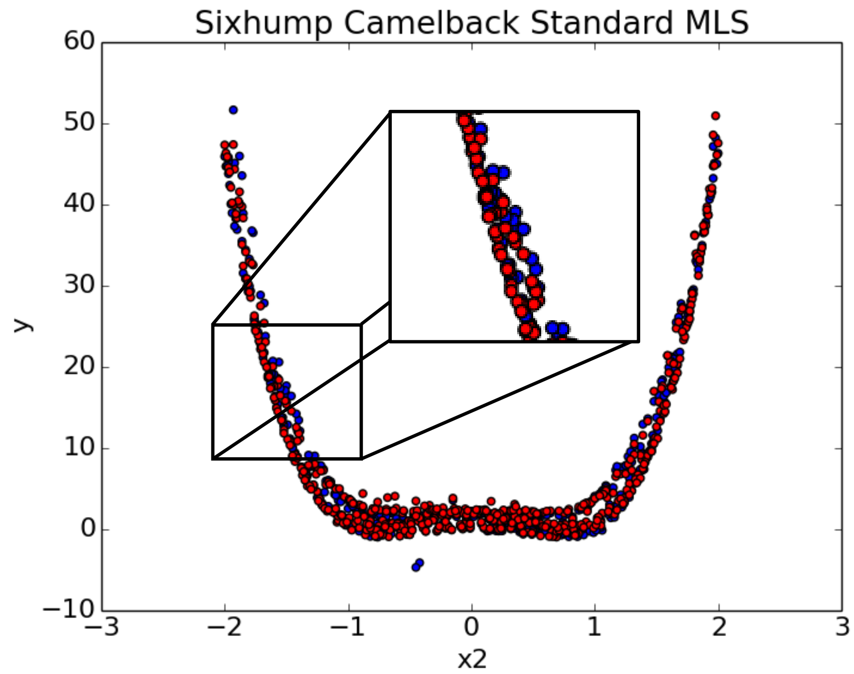


Blue = Approximation
Red = Calculation

2. Previous results - Advanced moving least square approximation

Sixhump Camelback: $y = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4; -2 \leq x_i \leq 2$





Blue = Approximation
Red = Calculation