Comparing different interpolation methods on two-dimensional test functions

Thomas Mühlenstädt, Sonja Kuhnt

TU Dortmund

ENBIS EMSE, St. Etienne

01.07.2009

Institut für Mathematische Statistik und industrielle Anwendungen

Statistische Modellbildung

Computer experiments

Simulation model for real world phenomena *Fang et al. (2006)*

- Deterministic
- Computational expensive

Needed: Easy to calculate surrogate

Setting:

- Design $D = \{\vec{x}_1, \ldots, \vec{x}_n\}, \vec{x}_i = (x_{i,1}, x_{i,2})$.
- One-dimensional output $y_1, \ldots, y_n$
- $y_i = f(\vec{x}_i)$, f unknown

Multivariate interpolation

Treated approaches:

- Kriging (Gaussian Random Fields)
- Thin plate spline (*TPS*)
- Natural neighbor interpolation (*NNI*)
- Kernel interpolation (*KI*)

Kriging

Santner et al. (2003)

$$Y(\vec{x}_i) = g_\beta(\vec{x}_i) + Z(\vec{x}_i), 1 \leq i \leq n,$$

- $g_\beta(\vec{x}_i)$ regression part (here: $g_\beta = \beta \in \mathbb{R}$)
- $Z(\vec{x}) \sim (0, \sigma^2)$ normally distributed
- $Z(\vec{x}_1)$ and $Z(\vec{x}_2), \vec{x}_1 \neq \vec{x}_2$ explicitly dependent:

$$cor_\theta(Z(\vec{x}_1), Z(\vec{x}_2)) \to 1 \text{ for } \vec{x}_1 \to \vec{x}_2$$

Kriging

$$cor(Z(\vec{x}_i), Z(\vec{x}_{i'})) = \exp\left(-\sum_{d=1}^{2} \theta_i \left|x_{i,d} - x_{i',d}\right|^2\right)$$

- Estimation of parameters β, θ, σ^2: REML
- Optimization of Log-likelihood: by *R* command `optim` for 50 different initial values

Thin plate spline (*TPS*)

- Generalization of cubic splines *Micula (2002)*
- Solves the following optimization problem:

  Search f^*, such that $I(f)$ is minimized
  in a suitable functional space
  under the constraint of interpolation

$$I(f) = \int_{\mathbb{R}^2} \left(\frac{\partial^2 f(\vec{x})}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 f(\vec{x})}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f(\vec{x})}{\partial x_2^2} \right)^2 d\vec{x}$$

Thin plate spline (*TPS*)

- Can be solved explicitly:

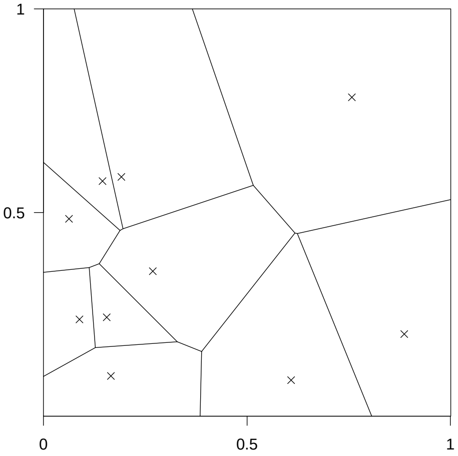
$$f^*(\vec{x}) = \sum_{i=1}^{n} \lambda_i \phi(\|\vec{x} - \vec{x}_i\|_2) + \lambda_{n+1} + \lambda_{n+2} x_1 + \lambda_{n+3} x_2$$

- $\phi(r) = r^2 log(r)$
- Computation $\lambda_1, \ldots, \lambda_{n+3}$: system of linear equations

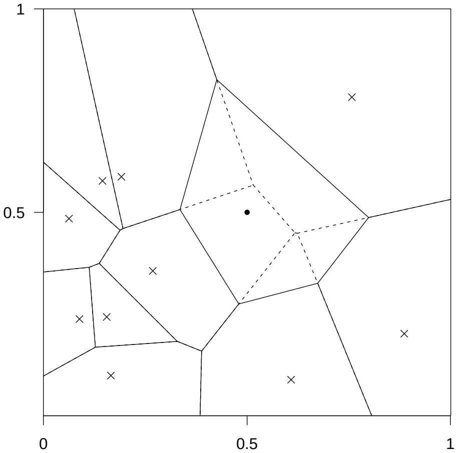
Natural neighbor interpolation (*NNI*)

- Weighted mean of the y-values *Sibson (1980)*
- Strictly local method
- Uses the Voronoi diagramm for weighting

Natural neighbor interpolation (*NNI*)



Natural neighbor interpolation (*NNI*)

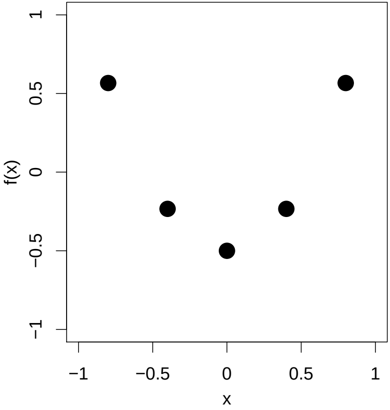


Kernel interpolation (*KI*)

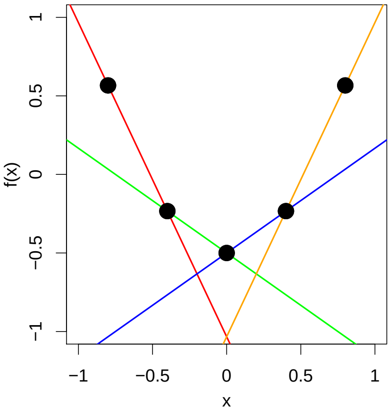
- Weight locally fitted linear functions under the constraint of interpolation *Mühlenstädt and Kuhnt (2009)*
- Split up the convex hull of the design into simplices S_j (Delaunay triangulation)
- Fit a linear function $\hat{y}_j(x)$ to each simplex S_j
- Weight the linear functions:

$$\frac{\sum_{j=1}^{N} g_j(\vec{x})\hat{y}_j(\vec{x})}{\sum_{j=1}^{N} g_j(\vec{x})}, \quad g_j(\vec{x}) = \frac{1}{\prod_{i=0}^{2} \|\vec{x}_i^j - \vec{x}\|_2^2}$$

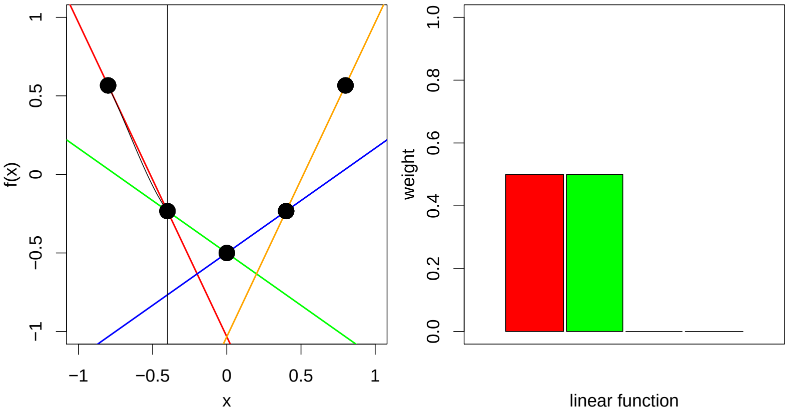
Kernel interpolation (*KI*)



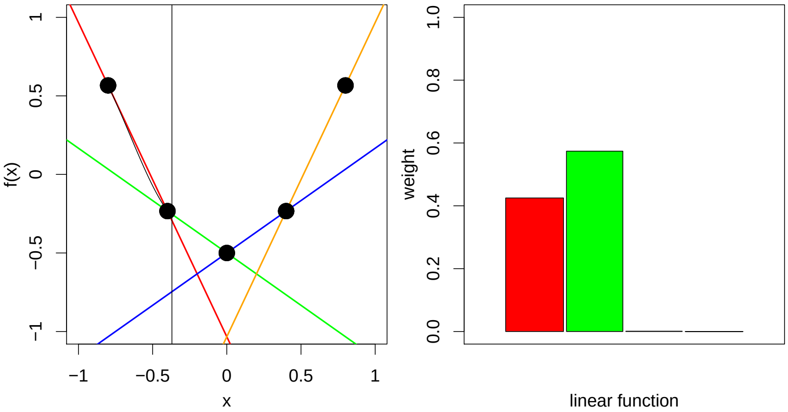
Kernel interpolation (*KI*)



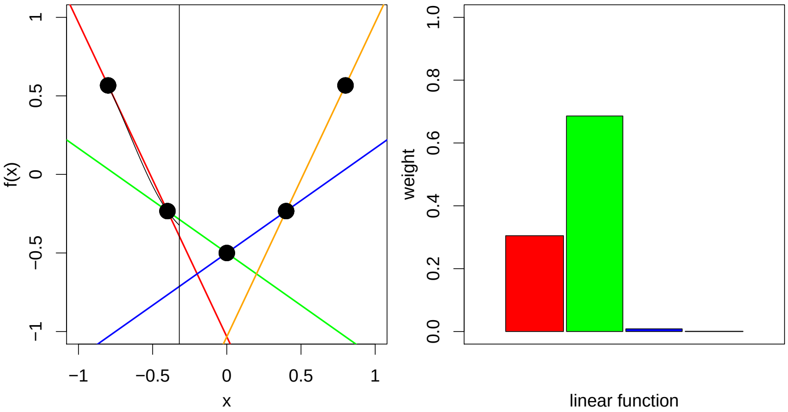
Kernel interpolation (*KI*)



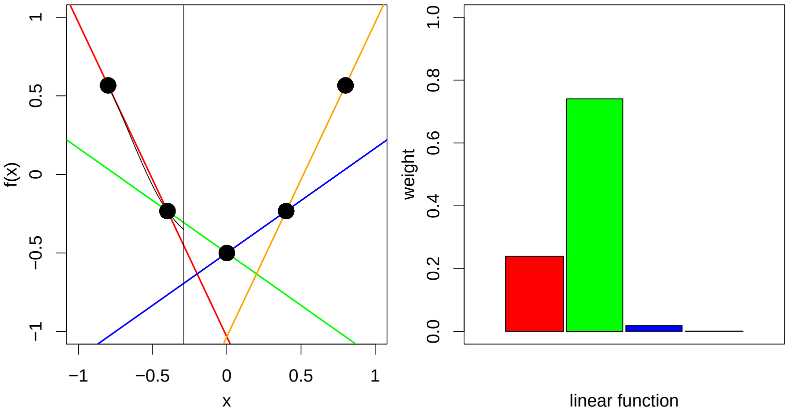
Kernel interpolation (*KI*)



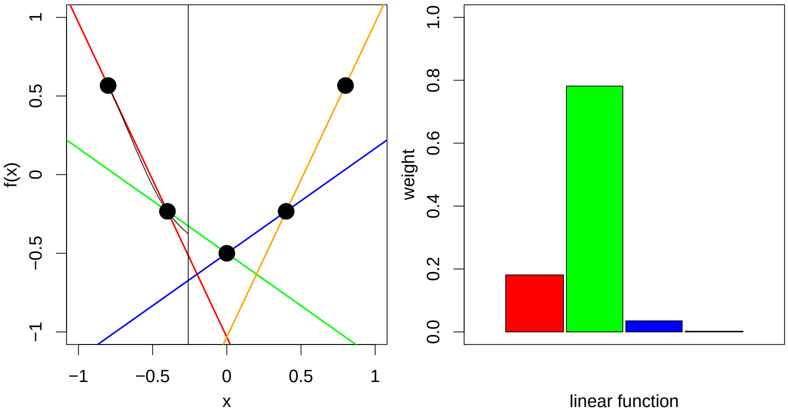
Kernel interpolation (*KI*)



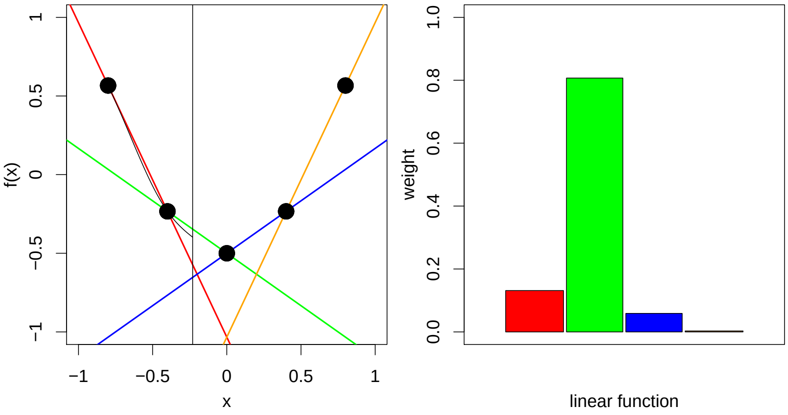
Kernel interpolation (*KI*)



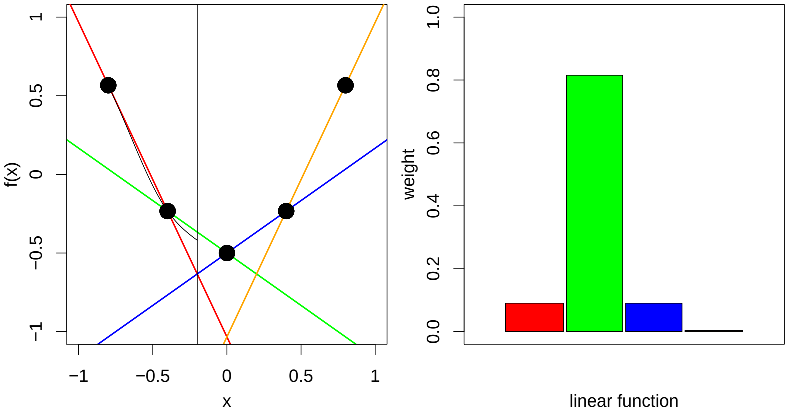
Kernel interpolation (*KI*)



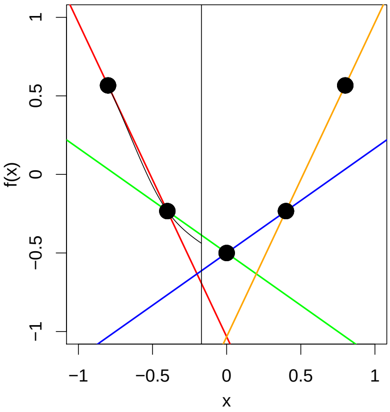
Kernel interpolation (*KI*)

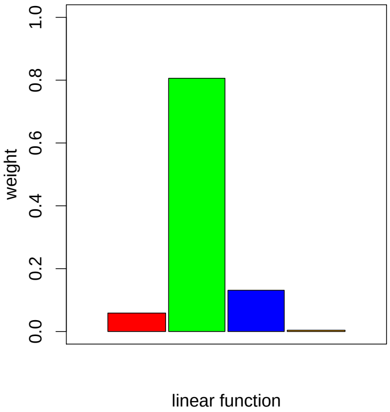


Kernel interpolation (*KI*)

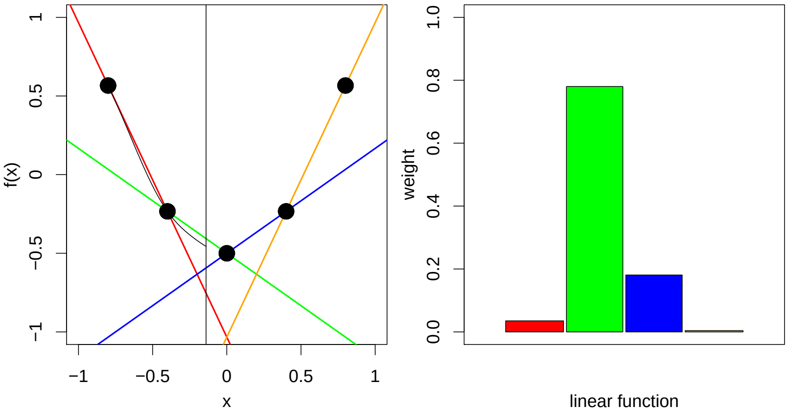


Kernel interpolation (*KI*)

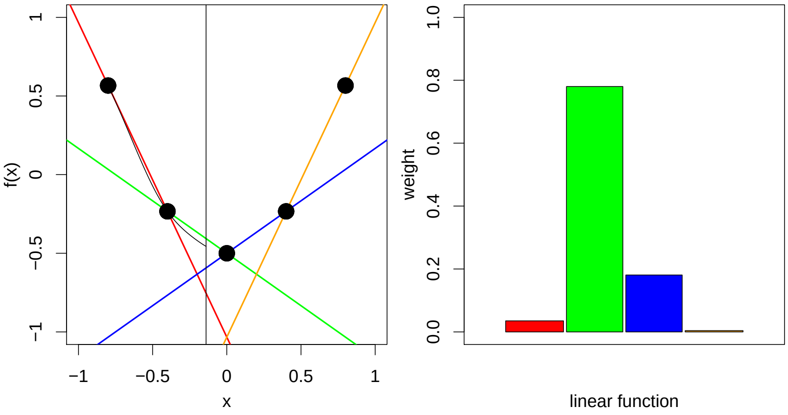




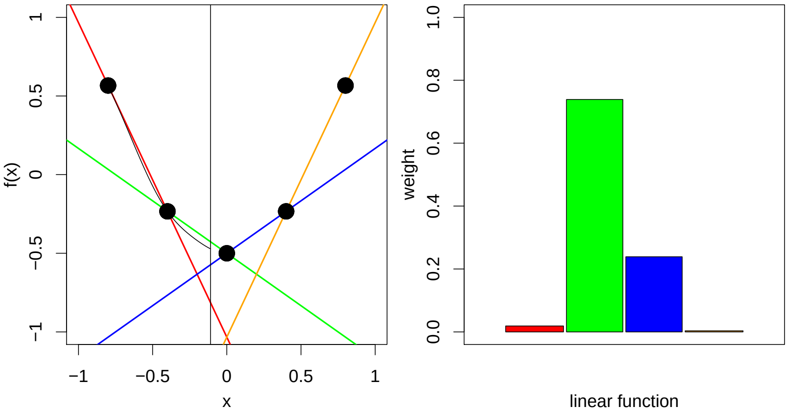
Kernel interpolation (*KI*)



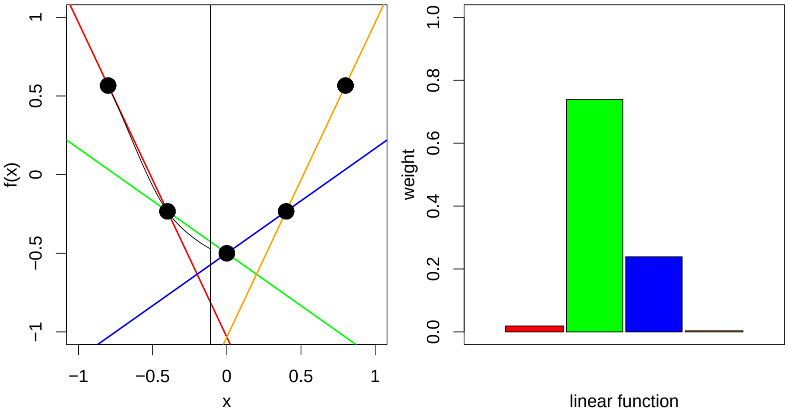
Kernel interpolation (*KI*)



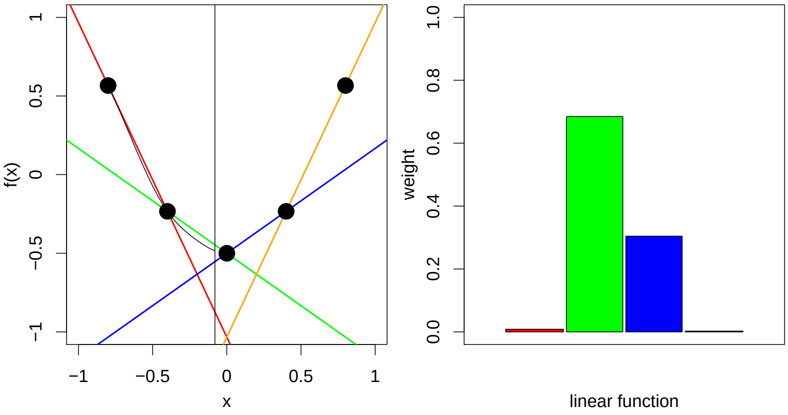
Kernel interpolation (*KI*)



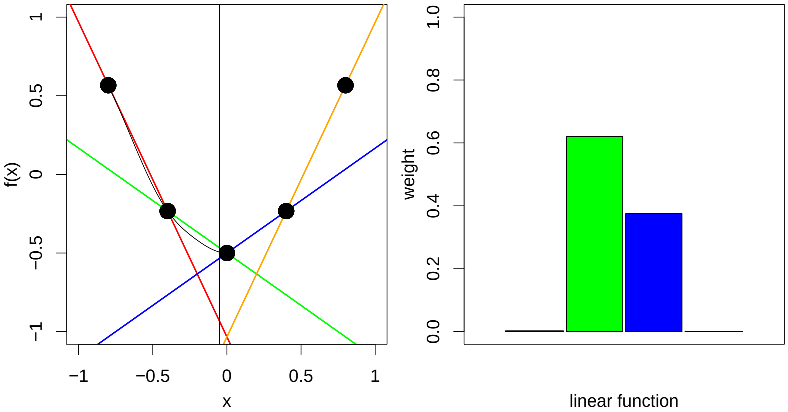
Kernel interpolation (*KI*)



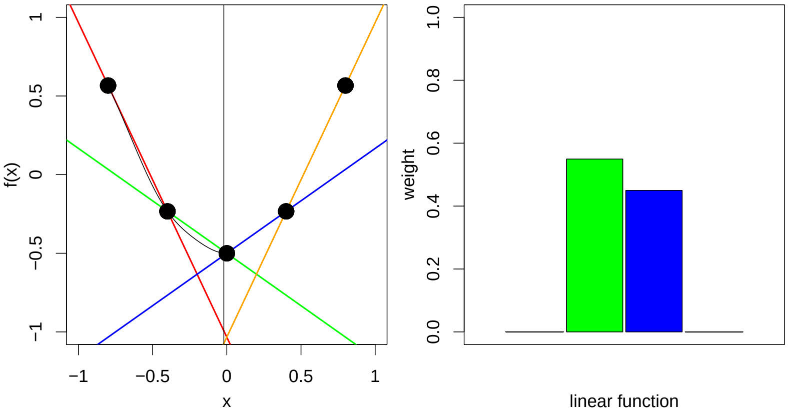
Kernel interpolation (*KI*)



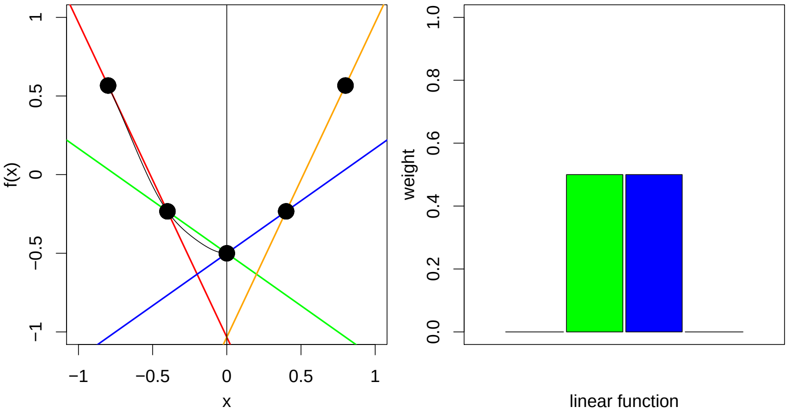
Kernel interpolation (*KI*)



Kernel interpolation (*KI*)



Kernel interpolation (*KI*)



Kernel interpolation (*KI*)

Properties of the Kernel interpolation:

- Continuous
- Differentiable (also at observation points)
- Able to predict outside of observation range
- Exactly reproduces linear functions

Comparison of interpolation methods

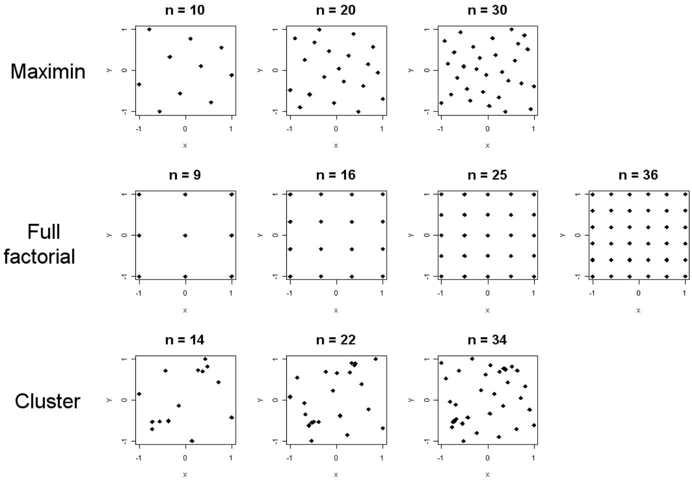
Which interpolation method should be used?

- 5 analytical 2-dim. examples
- Aim: high prediction power
- Prediction power: root mean square error (RMSE)
- Different experimental designs

Designs for computer experiments

- Difference to 'standard' DoE: no random error
- Example for a 'good' space filling designs: maximin latin hypercube design *Fang et al. (2006)*
- Also often encountered: factorial designs
- For sequential procedures: designs with clusters

Used Designs



Root Mean square error($RMSE$)

- Criterium for comparing different predictions:

$$RMSE(y, \hat{y}, \vec{r}_1, \ldots, \vec{r}_m) := \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y(\vec{r}_i) - \hat{y}(\vec{r}_i))^2}$$

- $\vec{r}_1, \ldots, \vec{r}_m$ points in the design space
- $\vec{r}_1, \ldots, \vec{r}_m$ 10000 points of a Sobol' Uniform Sequence

Implementation aspects

- Easy: Thin plate spline
- Acceptable:
  - Kriging
  - Kernel interpolation (if Delaunay triangulation is available)
- Difficult: Natural neighbor interpolation (Calculation of the Voronoi diagram constrained to design space)

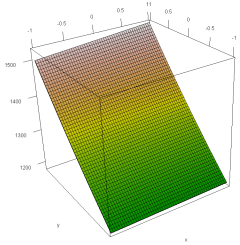
Computation times

Design	*TPS*	*KI*	Kriging	*NNI*
D_{10}^{Mm}	10	15	13 - 22	839
D_{20}^{Mm}	13	27	69 - 162	1536
D_{30}^{Mm}	15	39	178 - 448	2239

- Computation times in seconds
- For 10000 prediction points
- Based on the maximin latin hypercube designs

Die force

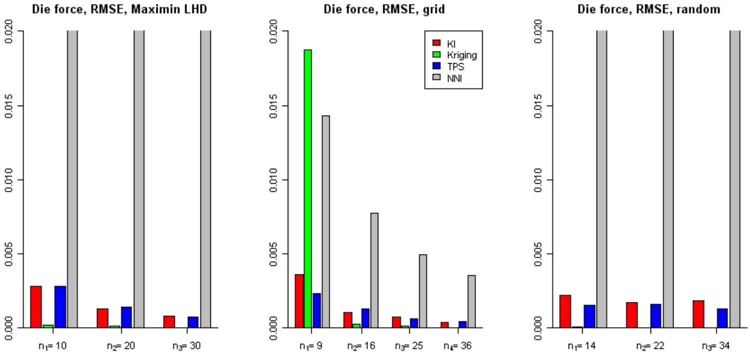
- Model for the effective die force in sheet metal forming processes depending on friction and blankholder force
- Nearly linear in one variable



$$f(\vec{x}) = 0.9996(1090.91 + 4x_1x_2)\exp\left(x_1\frac{\pi}{2}\right),$$

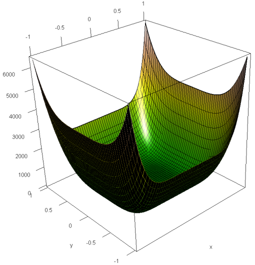
$$D = [0.05, 0.2] \times [5, 30]$$

Die force


Die force, RMSE, Maximin LHD
0.000
0.005
0.010
0.015
0.020
n_1= 10
n_2= 20
n_3= 30
Die force, RMSE, grid
KI
Kriging
TPS
NNI
n_1= 9
n_2= 16
n_3= 25
n_4= 36
Die force, RMSE, random
n_1= 14
n_2= 22
n_3= 34

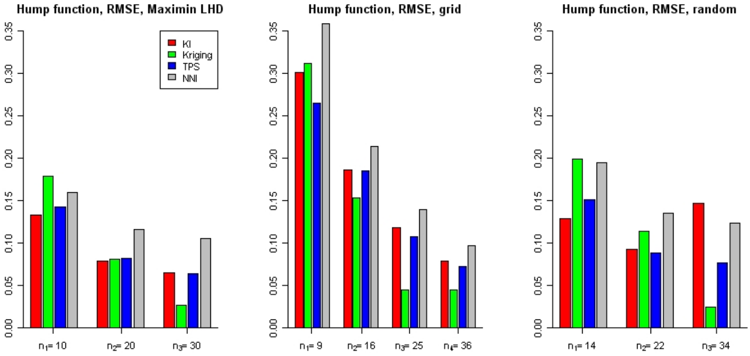
Hump function

- Standard example from optimization literature
- Extreme values on boundary



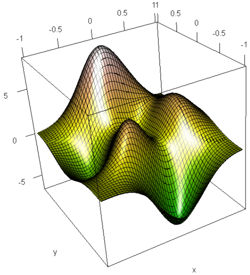
$$f(x,y) = 1.0316 + 4x^2 - 2.1x^4 + \frac{1}{3}x^6 + xy - 4y^2 + 4y^4,$$
$$D = [-5,5]^2$$

Hump function



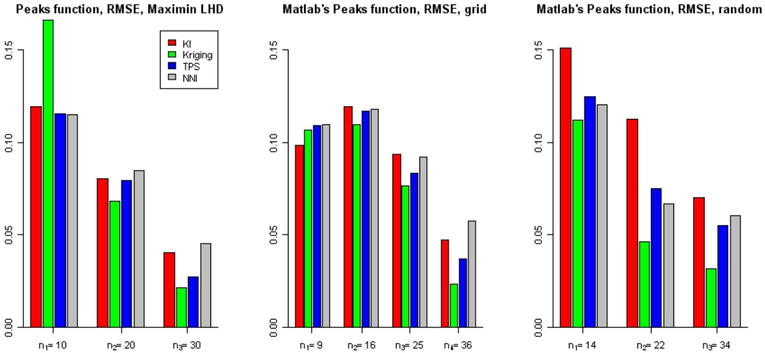
Matlabs peaks function

- Good example for a hilly contour



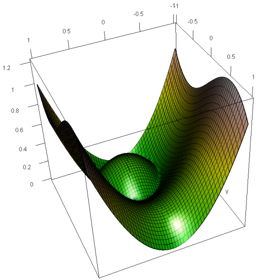
$$
\begin{aligned}
f(\vec{x}) =& 3(1 - x_1)^2 \exp\left(-x_1^2 - (x_2+1)^2\right) - \\
& 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)\exp\left(-x_1^2 - x_2^2\right) - \\
& \frac{1}{3}\exp\left(-(x_1+1)^2 - x_2^2\right), \quad D = [-2,2]^2
\end{aligned}
$$

Matlabs peaks function



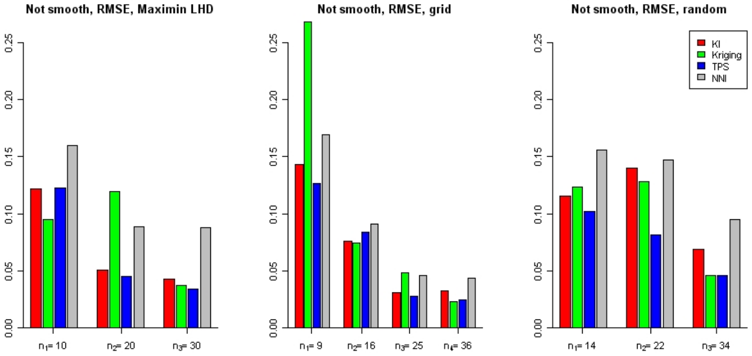
Not smooth

- Assumption of smoothness often unrealistic
- Continuous but not differentiable



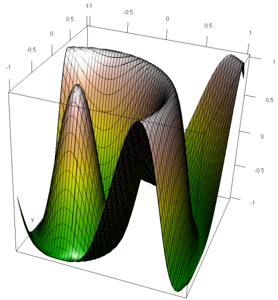
$$f(x, y) = |x^2 + \sin(0.5\pi y) - y|, \qquad D = [0, 1]^2.$$

Not smooth


Not smooth, RMSE, Maximin LHD
0.00
0.05
0.10
0.15
0.20
0.25
n1= 10
n2= 20
n3= 30
Not smooth, RMSE, grid
n1= 9
n2= 16
n3= 25
n4= 36
Not smooth, RMSE, random
KI
Kriging
TPS
NNI
n1= 14
n2= 22
n3= 34

Sibson's function

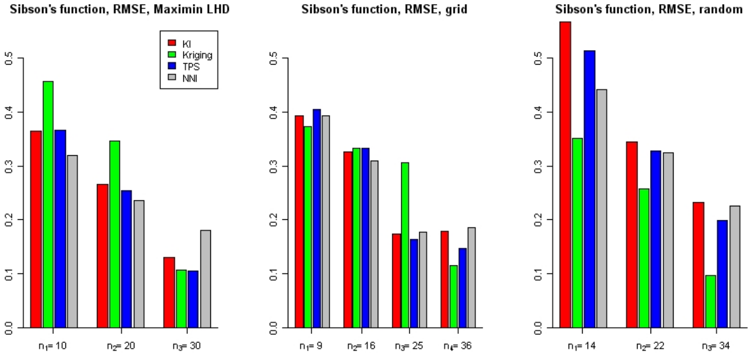
- Proposed by Sibson (1980) for illustrating *NNI*
- High complexity for size of sample



$$f(\vec{x}) = cos\left(4\pi\sqrt{(x_1 - 0.25)^2 + (x_2 - 0.25)^2}\right),$$

$$D = [0,1]^2$$

Sibson's function



Summary

- No overall winner
- Decision depends on design
- Kriging often very efficient, especially for higher sample sizes, designs with clusters
- *KI* and *TPS* good for small sample sizes
- *NNI* not recommendable

Literatur

Fang, K.-T., Li, R., and Sudjianto, A. (2006). *Design and Modeling for Computer Experiments*. Computer Science and Data Analysis Series. Chapman & Hall/CRC, New York.

Mühlenstädt, T. and Kuhnt, S. (2009). Kernel interpolation. Technical report, Faculty of Statistics, Technische Universität Dortmund, Dortmund, Germany.

Micula, G. (2002). A variational approach to spline functions theory. *General Mathematics*, 10(1):21–50.

Santner, T., Williams, B., and Notz, W. (2003). *The Design and Analysis of Computer Experiments*. Springer Series in Statistics. Springer Verlag, New York.

Sibson, R. (1980). A brief description of natural neighbor interpolation. In Barnett, V., editor, *Interpreting multivariate data*, pages 21–36. Wiley.