Project Parameter Estimation

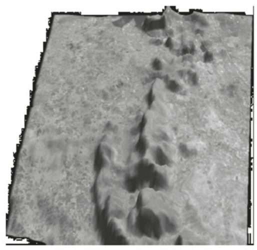
Part II: Short introduction to Kriging

Jemil Avers Butt
Prof. Dr. Andreas Wieser

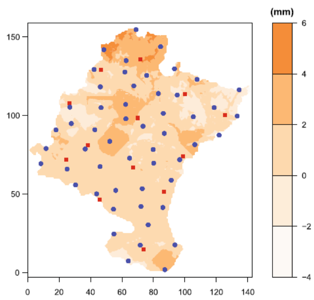
Applications of Kriging

Interpolation

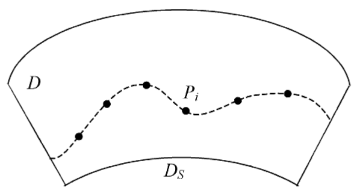
- **DEM**
  - Not going to spoil you



- **GPS positional errors**
  - Quality of GPS measurements influenced by factors, which are spatially distributed themselves
  - Systematic positional errors due to unmodelled effects
  - Interesting error patterns

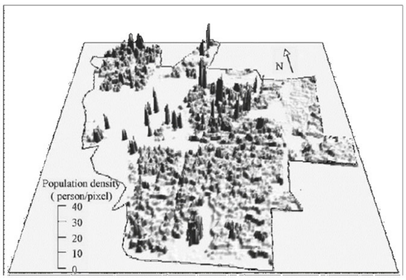


- **Geoid**
  - Kriging on a sphere
  - Earths gravity potential modeled as consisting of three parts, one of them random field to be predicted
  - How to translate the stationarity assumptions into a spherical framework?
- **(Bisgletscher Application Example)**



Applications of Kriging

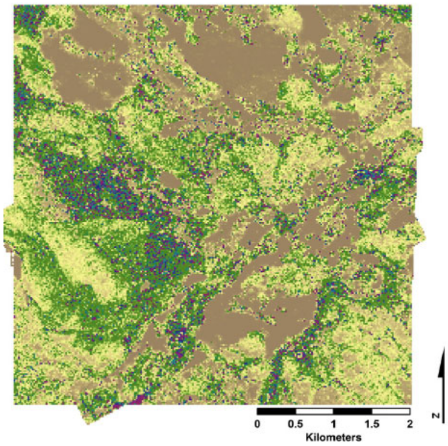
Estimation



Wu et al.

- **Population density**
  - **- Derive underlying population from Thematic mapper imagery**
  - **- Regression Kriging using impervious surface as auxiliary data**
  - **- Error: -0.3 % entire area, 10-15 % Block level**

- **Biomass**
  - **- Integrating field measurements, LiDAR and SAR measurements**
  - **- Estimation biomass in regions, where other-wise seldom possible**
  - **- «Sensor Fusion» task**
  - **- Alternative to «stratify and multiply» approach**

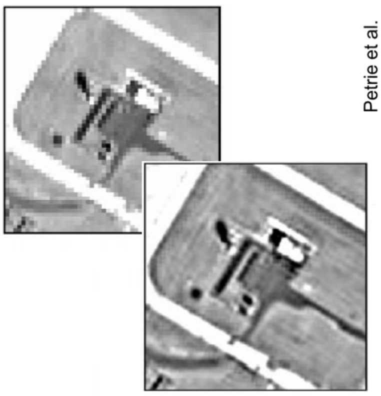


Tsui et al.

Applications of Kriging

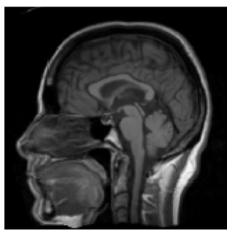
Image Analysis

- **Enhance spatial resolution**
  - **-Use relationships between overlapping high and low resolution images to sharpen the low res image.**
  - **-Uses Inverse to Block Kriging**
    - **Point measurements -> Average values over blocks**
  - **-Limited usefulness, when spatial relationships change harshly in the image to be sharpened or coregistration or choice of training area is lacking.**



Petrie et al.

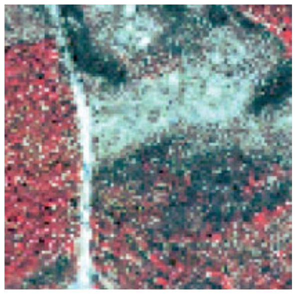
- **Image registration**
  - **- Exploit spatial relations between two images to optimally estimate a displacement field between them**
  - **-The displacement field is known only sparsely beforehand due to a landmark-based approach**
  - **- Method is generalization of widely used thin-plate-splines**



Applications of Kriging

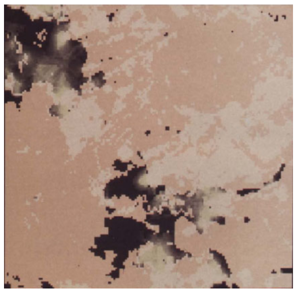
Image analysis

- Image fusion
  -Merging Information from remote sensors surveying the same region
  -Current algorithms have normally problems, when Images do not stem from the same sensor and time
  -Considering spatial correlations between two images leads to simple procedures and promising results



Meng, Borders,Madden 2009

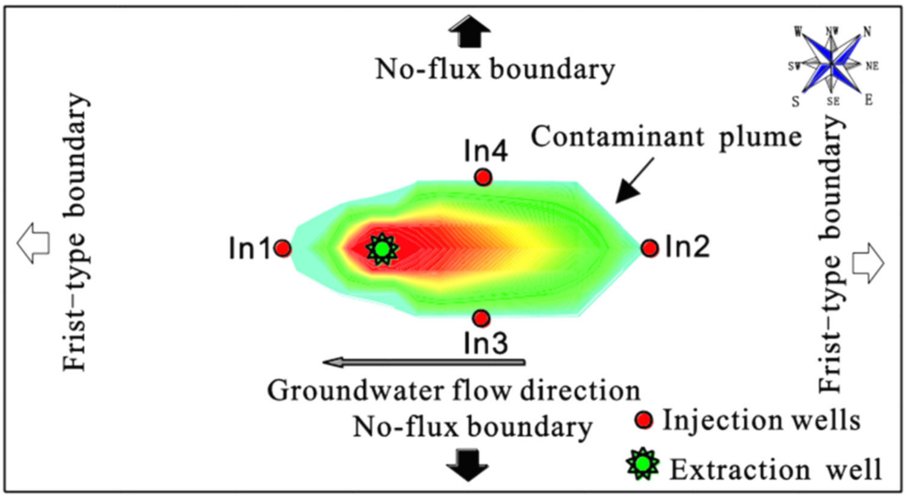
- Inpainting
  -Satellite images with obstructed views due to clouds are examined
  -The probabilities of pixels being pasture or not are estimated using indicator kriging
  -Quality of results depend on amount of spatial pattern



Rossi et al.

Applications of Kriging

Surrogate models



Luo et al.

- Kriging as a metamodelling tool
- Interpolate Computer experiments and by that reduce the computational Burden
- Discuss! How would you interpolate for example simulations? What are the coordinates?

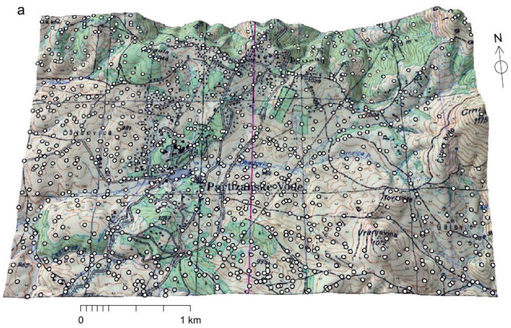
Applications of Kriging

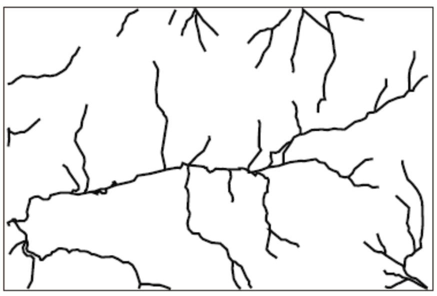
Data analysis

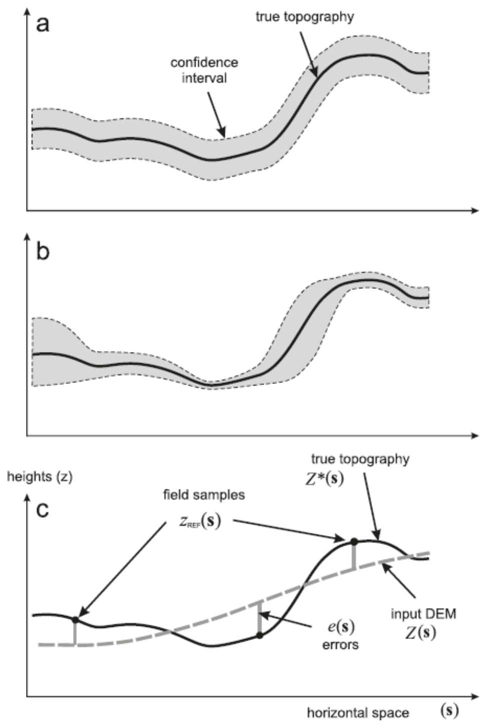
- **Epidemic data analysis**
  -Analysis of number of Dengue cases and tuberculosis occurrences in india
  -Various forms of Kriging Methods applied (Simple, ordinary, universal, bayesian Kriging)
  -Variation pattern in time/space used for prediction of cases in districts, for which data of neighboring districts was already compiled

- **Landscape spatial patterns**
  -Strong link between landscape patterns and ecological functions is assumed
  -Hundreds of quantitative measures to capture characteristics of spatial heterogeneity
  -Covariance can capture heterogeneity at different scales

Applications of Kriging





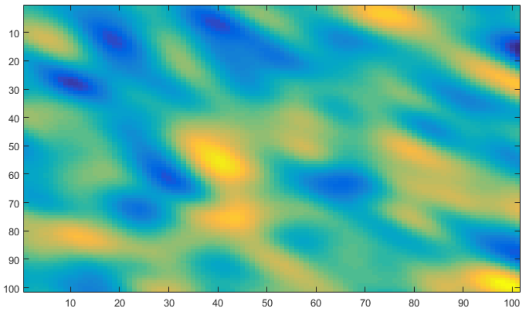


Hengl et al.

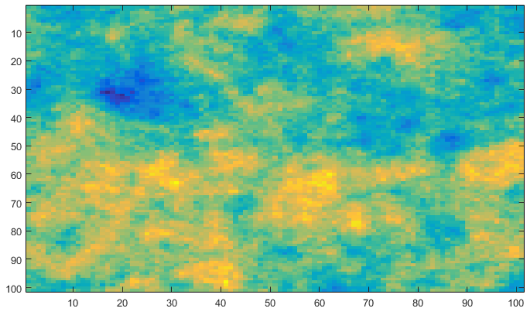
DEM Interpolation
……any Suggestions?

Advanced applications: Filtering, Simulation

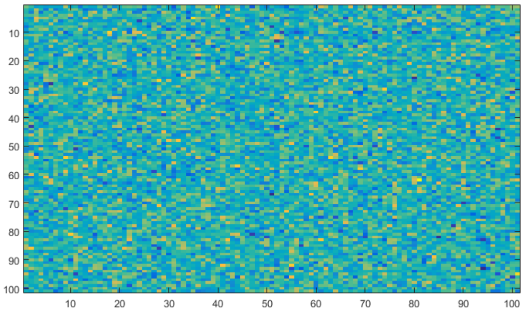
Signal



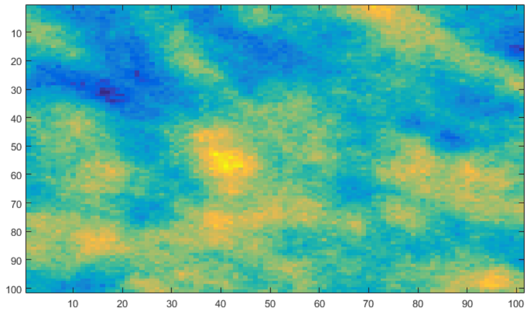
Deviation



Noise

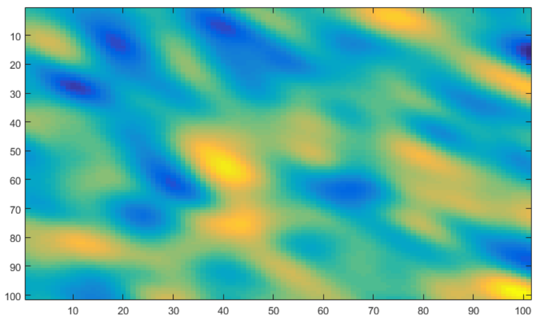


Data

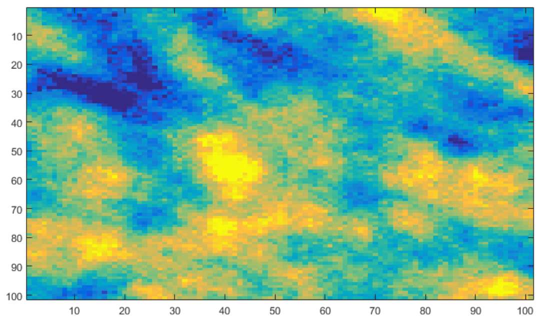


Advanced applications: Filtering, Simulation

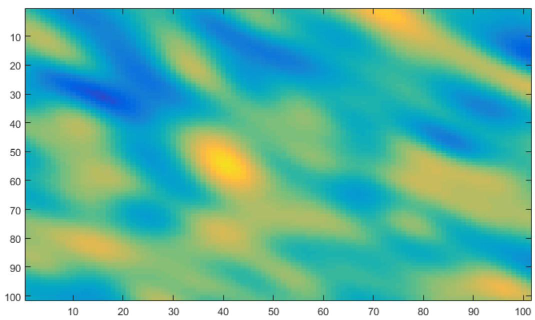
Signal



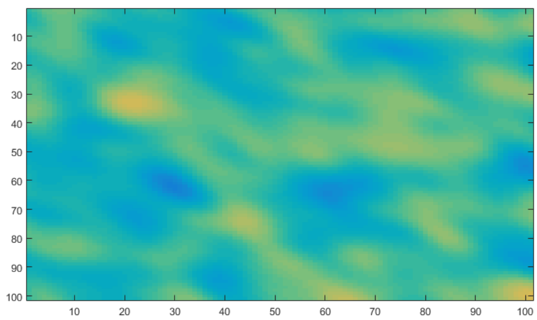
Data



Estimated Signal

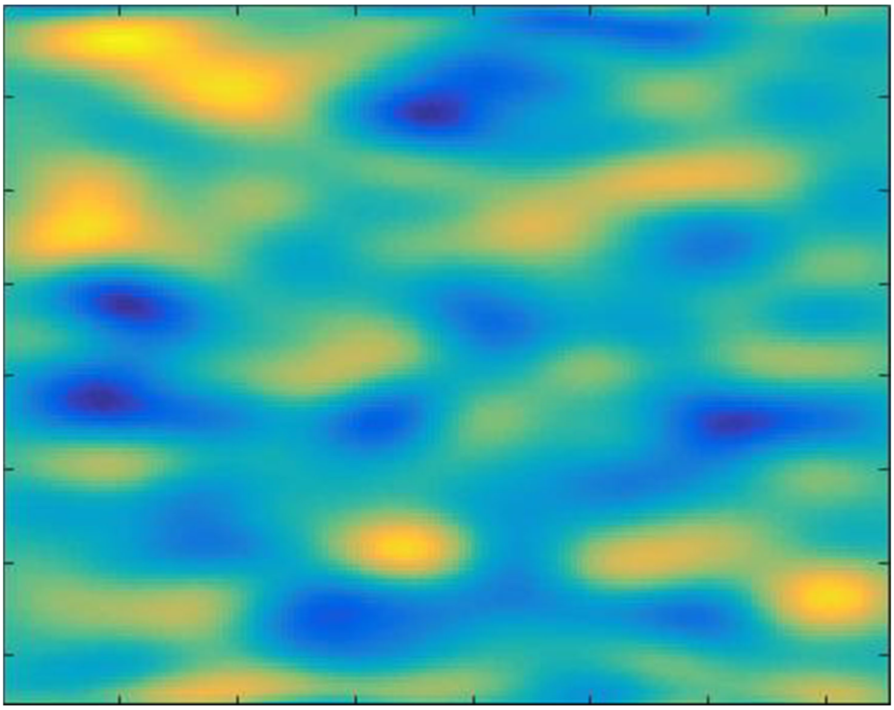


Error



Advanced applications: SPDE, Filtering, Simulation

A spatiotemporal random field with squared exponential covariance



Advanced applications: SPDE, Filtering, Simulation

- **Mitigation of atmospheric effects in terrestrial radar interferometry**

A hierarchy of estimation tasks (debatable!)

Name	Signal	Noise	Example
	Deterministic	Stochastic	
	Stochastic	None	
	Stochastic	Stochastic	

Introduction of additional constraints and uncertainties possible:

- ✓ **Regression on axiliary data**
- ✓ **Regression on second RF**
- ✓ **Inclusion of physical laws**
- ✓ **Restriction to subspaces of R^n**

All of the above can be handled in the scalar, vector, tensor case.

A compendium of Kriging types and their SLAE's

Univariate methods

- **Simple Kriging : Kriging with known mean**
- **Ordinary Kriging: Kriging with unkown constant mean**
- **Universal Kriging: Kriging with unkown nonconstant mean and possibly external drift**
- **Intrinsic Kriging: Kriging for instationary RF's that can be mapped linearly to stationary RF's**

Multivariate methods

- **Simple Cokriging: Kriging where the RF to be estimated is only stochastically related to the observed RF**
- **Universal Cokriging: Like Simple Cokriging but with drifts for each RF**
- **Disjunctive Kriging: Kriging with estimators that are nonlinear functions of the data**

Simple Kriging

Kriging with known mean

$$\hat{Z}_{s_0} = \sum_{k=1}^{n} \lambda_k Z_k \qquad \hat{Z}_{s_0} = \vec{\lambda}^T \vec{z} \qquad \text{where } Z_{\cdot}(\cdot) \text{ is 0 mean}$$

$$\vec{\lambda} = \underline{C}^{-1} \vec{c}_0$$

$$\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} c(s_1,s_1) & \cdots & c(s_1,s_n) \\ \vdots & \ddots & \vdots \\ c(s_n,s_1) & \cdots & c(s_n,s_n) \end{bmatrix}^{-1} \begin{bmatrix} c(s_1,s_0) \\ \vdots \\ c(s_n,s_0) \end{bmatrix}$$

Ordinary Kriging

Kriging with unknown but constant mean

$$\hat{Z}_{s_0} = \sum_{k=1}^{n} \lambda_k Z_k \qquad \hat{Z}_{s_0} = \vec{\lambda}^T \vec{z} \qquad \begin{bmatrix} \vec{\lambda} \\ \mu_1 \end{bmatrix} = \begin{bmatrix} \underline{C} & \underline{F} \\ \underline{F}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \vec{c}_0 \\ f_0 \end{bmatrix}$$

$$\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu_1 \end{bmatrix} = \begin{bmatrix} c(s_1,s_1) & \cdots & c(s_1,s_n) & 1 \\ \vdots & \ddots & \vdots & \vdots \\ c(s_n,s_1) & \cdots & c(s_n,s_n) & 1 \\ 1 & \cdots & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} c(s_1,s_0) \\ \vdots \\ c(s_n,s_0) \\ 1 \end{bmatrix}$$

Universal Kriging

Kriging with unkown nonconstant mean and possibly external drift

$$\hat{Z}_{s_0} = \sum_{k=1}^{n} \lambda_k Z_k \qquad \hat{Z}_{s_0} = \vec{\lambda}^T \vec{z}$$

$$\begin{bmatrix} \vec{\lambda} \\ \vec{\mu} \end{bmatrix} = \begin{bmatrix} \underline{C} & \underline{F} \\ \underline{F}^T & \underline{0} \end{bmatrix}^{-1} \begin{bmatrix} \vec{c}_0 \\ \vec{f}_0 \end{bmatrix}$$

$$\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu_1 \\ \vdots \\ \mu_q \end{bmatrix} = \begin{bmatrix} c(s_1,s_1) & \cdots & c(s_1,s_n) & f^1(s_1) \cdots f^q(s_1) \\ \vdots & \ddots & \vdots & \vdots \quad \ddots \quad \vdots \\ c(s_n,s_1) & \cdots & c(s_n,s_n) & f^1(s_n) \cdots f^q(s_n) \\ f^1(s_1) & \cdots & f^1(s_n) & 0 \quad \cdots \quad 0 \\ \vdots & \ddots & \vdots & \vdots \quad \ddots \quad \vdots \\ f^q(s_1) & \cdots & f^q(s_n) & 0 \quad \cdots \quad 0 \end{bmatrix}^{-1} \begin{bmatrix} c(s_1,s_0) \\ \vdots \\ c(s_n,s_0) \\ f^1(s_0) \\ \vdots \\ f^q(s_0) \end{bmatrix}$$

Note: f^l in addition to deterministic functions can be any everywhere known variable;
- for example altitudes, temperature or whatever else is available

Instrinsic Kriging

Kriging for instationary RF's that can be mapped linearly to stationary RF's

$$\hat{Z}_{s_0} = \sum_{k=1}^{n} \lambda_k Z_k \qquad \hat{Z}_{s_0} = \vec{\lambda}^T \vec{z}$$

$$\begin{bmatrix} \vec{\lambda} \\ \vec{\mu} \end{bmatrix} = \begin{bmatrix} \underline{G} & \underline{F} \\ \underline{F}^T & \underline{0} \end{bmatrix}^{-1} \begin{bmatrix} \vec{g}_0 \\ \vec{f}_0 \end{bmatrix}$$

$$\begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \mu_1 \\ \vdots \\ \mu_q \end{bmatrix} = \begin{bmatrix} g(s_1,s_1) & \cdots & g(s_1,s_n) & f^1(s_1) \cdots f^q(s_1) \\ \vdots & \ddots & \vdots & \vdots \quad \ddots \quad \vdots \\ g(s_n,s_1) & \cdots & g(s_n,s_n) & f^1(s_n) \cdots f^q(s_n) \\ f^1(s_1) & \cdots & f^1(s_n) & 0 \quad \cdots \quad 0 \\ \vdots & \ddots & \vdots & \vdots \quad \ddots \quad \vdots \\ f^q(s_1) & \cdots & f^q(s_n) & 0 \quad \cdots \quad 0 \end{bmatrix}^{-1} \begin{bmatrix} g(s_1,s_0) \\ \vdots \\ g(s_n,s_0) \\ f^1(s_0) \\ \vdots \\ f^q(s_0) \end{bmatrix}$$

Note: This is exactly the same system as for UK but with the generalized covariances $g(s,t)$ instead of the regular covariances $c(s,t)$.

Generalized covariances

Here $g(\cdot,\cdot)$ is called the generalized covariance. It satisfies the following equation: Let ∇ be a differential operator such that $\nabla Z_{\cdot}(\cdot) = X_{\cdot}(\cdot)$ is a second order stationary RF with stationary covariance $c_X(\cdot,\cdot)$ and $c_Z(\cdot,\cdot)$ be the instationary covariance function of $Z_{\cdot}(\cdot)$.

Then $\nabla \otimes \nabla g(\cdot,\cdot) = \nabla \otimes \nabla c_Z(\cdot,\cdot)$; $g(\cdot,\cdot) - c_Z(\cdot,\cdot) \in \ker \nabla \otimes \nabla$ such that $g(\cdot,\cdot)$ denotes the equivalence class of functions differing from the true underlying $c_Z(\cdot,\cdot)$ by an element that gets mapped to 0 by $\nabla \otimes \nabla$. This is done simply because $c_Z(\cdot,\cdot)$ is not inferrable from the data but $g(\cdot,\cdot) = \int \otimes \int \; c_X(\cdot,\cdot)$ is, where $\left(\int \nabla - \mathrm{id}\right) Z_{\cdot}(\cdot) \in \ker \nabla$.

Simple Cokriging

Kriging where the RF to be estimated is only stochastically related to the observed RF

$$\hat{Z}^1_{s_0} = \sum_{i=1}^{p}\sum_{j=1}^{n} \lambda_{ij} Z^i{}_j \qquad \hat{Z}_{s_0} = \sum_{i=1}^{p} \vec{\lambda}_i^T \vec{z}^i$$

$$\begin{bmatrix} \vec{\lambda}_1 \\ \vdots \\ \vec{\lambda}_p \end{bmatrix} = \begin{bmatrix} \underline{C}_{11} & \cdots & \underline{C}_{1p} \\ \vdots & \ddots & \vdots \\ \underline{C}_{p1} & \cdots & \underline{C}_{pp} \end{bmatrix}^{-1} \begin{bmatrix} \vec{c}_{10} \\ \vdots \\ \vec{c}_{p0} \end{bmatrix}$$

Note: For example $\underline{C}_{12}$ contains the (cross-) covariances between the RF $Z^1_{\cdot}(\cdot)$ and the RF $^2_{\cdot}(\cdot)$. So the estimation uses p datasets all coming from some RF hopefully strongly correlated to the one, which is to be estimated.

Simple Cokriging

Kriging where the RF to be estimated is only stochastically related to the observed RF

$$\hat{Z}^1_{s_0} = \sum_{i=1}^{p}\sum_{j=1}^{n} \lambda_{ij} Z^i{}_j \qquad \hat{Z}_{s_0} = \sum_{i=1}^{p} \vec{\lambda}_i^T \vec{z}^i$$

$$\begin{bmatrix} \lambda_{11} \\ \vdots \\ \lambda_{1n} \\ \lambda_{21} \\ \vdots \\ \lambda_{2n} \\ \vdots \\ \lambda_{p1} \\ \vdots \\ \lambda_{pn} \end{bmatrix} = \begin{bmatrix} C_{11}(s_1,s_1) & \cdots & C_{11}(s_1,s_n) & \cdots & \cdots & \cdots C_{1p}(s_1,s_1) & \cdots & C_{1p}(s_1,s_n) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ C_{11}(s_n,s_1) & \cdots & C_{11}(s_n,s_n) & \cdots & \cdots & \cdots C_{1p}(s_n,s_1) & \cdots & C_{1p}(s_n,s_n) \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ C_{p1}(s_1,s_1) & \cdots & C_{p1}(s_1,s_n) & \cdots & \cdots & \cdots C_{pp}(s_1,s_1) & \cdots & C_{pp}(s_1,s_n) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ C_{p1}(s_n,s_1) & \cdots & C_{p1}(s_n,s_n) & \cdots & \cdots & \cdots C_{pp}(s_n,s_1) & \cdots & C_{pp}(s_n,s_n) \end{bmatrix}^{-1} \begin{bmatrix} c_{11}(s_1,s_0) \\ \vdots \\ c_{11}(s_n,s_0) \\ \vdots \\ \vdots \\ \vdots \\ c_{1p}(s_1,s_0) \\ \vdots \\ c_{1p}(s_n,s_0) \end{bmatrix}$$

Universal Cokriging

Like Simple Cokriging but with algebraically independent drifts for each RF

$$\hat{Z}^1_{s_0} = \sum_{i=1}^{p}\sum_{j=1}^{n} \lambda_{ij} Z^i{}_j \qquad \hat{Z}_{s_0} = \sum_{i=1}^{p} \vec{\lambda}_i^{\,T} \vec{z}^{\,i}$$

$$\begin{bmatrix} \vec{\lambda}_1 \\ \vdots \\ \vec{\lambda}_p \\ \vec{\mu}_1 \\ \vdots \\ \vec{\mu}_p \end{bmatrix} = \begin{bmatrix} \underline{C}_{11} & \cdots & \underline{C}_{1p} & \underline{F}_1 & \cdots & \underline{0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \underline{C}_{p1} & \cdots & \underline{C}_{pp} & \underline{0} & \cdots & \underline{F}_p \\ \underline{F}_1^T & \cdots & \underline{0} & \underline{0} & \cdots & \underline{0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \underline{0} & \cdots & \underline{F}_p^T & \underline{0} & \cdots & \underline{0} \end{bmatrix}^{-1} \begin{bmatrix} \vec{c}_{10} \\ \vdots \\ \vec{c}_{p0} \\ \vec{f}_{10} \\ \vec{0} \end{bmatrix}$$

Note: For example $\underline{C}_{12}$ contains the (cross-) covariances between the RF $Z^1_\cdot(\cdot)$ and the RF $^2_\cdot(\cdot)$. So the estimation uses p datasets all coming from some RF hopefully strongly correlated to the one, which is to be estimated.

Explanation of terms

Z_{s_0} : The random variable to be estimated. Is located at position $s_0 \in U$.

$\hat{Z}_{s_0}$: The estimator for the RV Z_{s_0}

$U \subset \mathbb{R}^N$: The underlying N-dimensional space; to each $s \in U$ is associated a RV by the random fields Z, Z^i, X and so on. In most practical cases $U \subset \mathbb{R}^2$ and this corresponds to the usual 2-d RF's we already know well.

$Z_\cdot(\cdot)$: A RF. In the simple, ordinary, universal and instrinsic Kriging cases, the RF for which estimation is to be performed.

$Z_\cdot(\cdot) : \Omega \times U \ni (\omega, s) \mapsto Z_s(\omega) \in \mathbb{R}$

$\left\{Z^i_\cdot(\cdot)\right\}_{i=1}^{p}$: A set of p RF's. The estimation is to be performed for the RF $Z^1_\cdot(\cdot)$.

$Z^i_\cdot(\cdot) : \Omega \times U \ni (\omega, s) \mapsto Z^i{}_s(\omega) \in \mathbb{R}$

Ω : The probability (measure) space with $\sigma -$ Algebra and probability measure

m$(\cdot)$: The mean function. $m(\cdot) : U \ni s \mapsto m(s) = E[Z_s] \in \mathbb{R}$

$c(\cdot,\cdot)$: The covariance function. $c(\cdot,\cdot) : U \times U \ni (s,t) \mapsto c(s,t) = \sigma(Z_s, Z_t) \in \mathbb{R}$

$\sigma(\cdot,\cdot)$: The covariance function. $\sigma(\cdot,\cdot) : L^2(\Omega) \times L^2(\Omega) \ni (Z_s, Z_t) \mapsto \sigma(Z_s, Z_t)$

$= E[Z_s Z_t] - E[Z_s]E[Z_t] \in \mathbb{R}$

Explanation of terms

$\gamma(\cdot,\cdot)$: The variogram function $\gamma(\cdot,\cdot): U \times U \ni (s,t) \mapsto \gamma(s,t) = E[(Z_s - Z_t)^2] \in \mathbb{R}$

$\{\lambda_k\}_{k=1}^n$: Set of n coefficients, $\lambda_k \in \mathbb{R}$

$\{z_k\}_{k=1}^n$: Set of n data values, $z_k \in \mathbb{R}$, where z_k is a realization of $Z_{s_k}(\cdot)$

$\vec{\lambda}$: (n,1) vector $[\lambda_1,...,\lambda_n]^T$

$\vec{z}$: (n,1) vector $[z_1,...,z_n]^T$

λ_{ij} : Coefficient for z_j^i, where z_j^i is realization of $Z_{s_j}^i(\cdot)$

$\vec{\lambda}_i$: (n,1) vector $[\lambda_{11},...,\lambda_{1n}]^T$

$\{\mu_k\}_{k=1}^q$: Set of q Lagrange multipliers $\mu_k \in \mathbb{R}$

$\vec{\mu}$: (q,1) vector $[\mu_1,...,\mu_q]^T$

$\vec{\mu}_i$: (q,1) vector $[\mu_{11},...,\mu_{1q}]^T$

μ_{ij} : Lagrange multiplier for $\sum_{k=1}^q f^j(s_k)$ which is a trend function for the RF $Z_\cdot^i(\cdot)$

Explanation of terms

$\underline{C}$: The (n,n) matrix $\{c(s_i,s_j)\}_{i=1}^n {}_{j=1}^n$

$\vec{c}_0$: The (n,1) vector$[c(s_1,s_0),...,c(s_n,s_0)]^T$

$f^l(\cdot)$: The l-th trend function of the RF $Z_\cdot(\cdot)$. $f^l(\cdot): U \ni s \mapsto f^l(s) \in \mathbb{R}$

$\underline{F}$: The (m,q) matrix $\{f^j(s_i)\}_{i=1}^{n_{j=1}^q}$

$\underline{C}_{ij}$: The (n,n) matrix $\{c_{ij}(s_k,s_l)\}_{k=1}^{n_{l=1}^n}$

$c_{ij}(\cdot,\cdot)$: The (cross-) covariance function. $c_{ij}(\cdot,\cdot): U \times U \ni (s,t) \mapsto c_{ij}(s,t)$
$= \sigma(Z_s^i(\cdot), Z_t^j(\cdot)) \in \mathbb{R}$

$\underline{F}_i$: The (n,q) matrix $\{f^l(s_j)\}_{j=1}^{n_{l=1}^q}$ of trend functions of the RF $Z_\cdot^i(\cdot)$

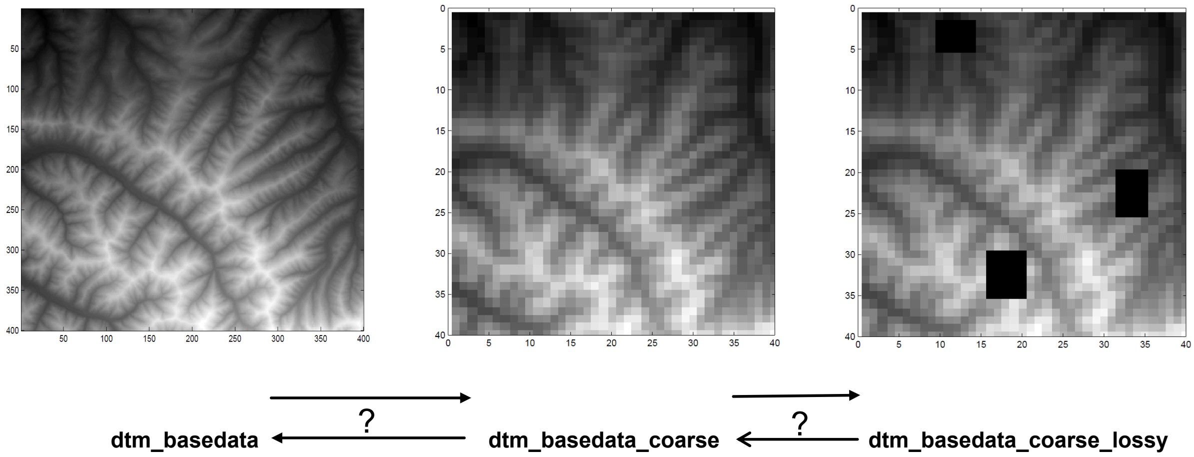
$\vec{c}_{i0}$: The (n,1) vector $[c_{1i}(s_0,s_1),....,c_{1i}(s_0,s_n)]^T$

$\vec{f}_{10}$: The (q,1) vector$[f^1(s_0),...,f^q(s_0)]^T$

Programming Project

Programming project: background

- You have acquired coarse data concerning the elevation of a region that you are interested in.
- Sadly, in this DEM there are coverage holes that need to be eliminated before you can use it for further applications.
- Solve this Problem with ordinary Kriging and another interpolation method of your choice.

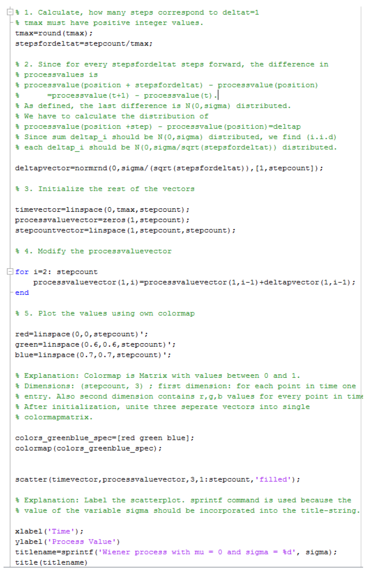


Programming Project

Programming project: goals

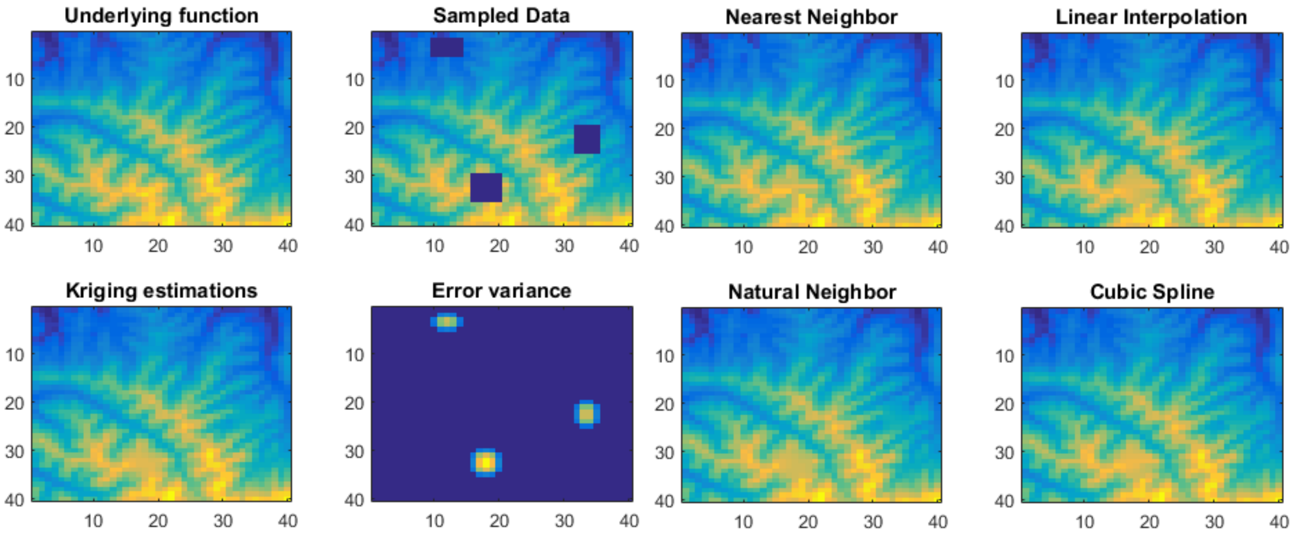
- Compare the output of both methods and explain the differences.
- Also explain the deviations from the true values and how you could improve your results.
- Achieving both of the above will probably force you to go through a variety of steps:
    1. Estimate covariances or semivariances
    2. Fit a model into an experimental variogram
    3. Use these models to look up the values needed to derive the weights for the BLUP
    4. Plot the difference between predicted and true values
    5. Give a quality estimate for each estimation
- You should do this in MATLAB in an orderly fashion

- In the end: Have a well documented function, that takes as inputs a set of measurements and coordinates and outputs a regularly spaced grid of interpolated values and coordinates.

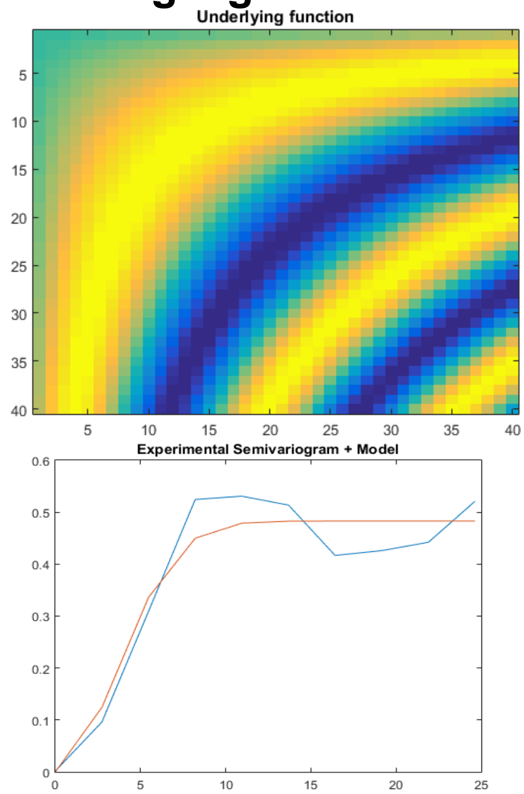


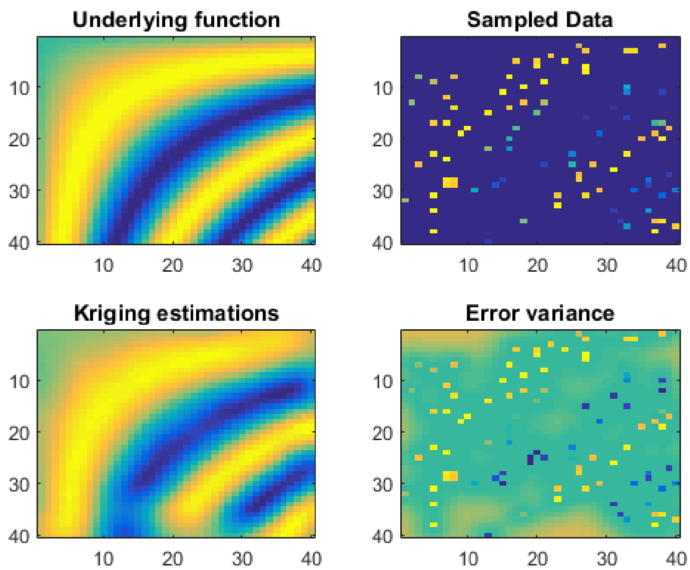
```
% 1. Calculate, how many steps correspond to deltat=1
% tmax must have positive integer values.
tmax=round(tmax);
stepsfordeltat=stepcount/tmax;

% 2. Since for every stepsfordeltat steps forward, the difference in
% processvalues is
% processvalue(position + stepsfordeltat) - processvalue(position)
%      =processvalue(t+1) - processvalue(t).
% As defined, the last difference is N(0,sigma) distributed.
% We have to calculate the distribution of
% processvalue(position +step) - processvalue(position)=deltap
% Since sum deltap_i should be N(0,sigma) distributed, we find (i.i.d)
% each deltap_i should be N(0,sigma/sqrt(stepsfordeltat)) distributed.

deltapvector=normrnd(0,sigma/(sqrt(stepsfordeltat)),[1,stepcount]);

% 3. Initialize the rest of the vectors

timevector=linspace(0,tmax,stepcount);
processvaluevector=zeros(1,stepcount);
stepcountvector=linspace(1,stepcount,stepcount);

% 4. Modify the processvaluevector

for i=2: stepcount
    processvaluevector(1,i)=processvaluevector(1,i-1)+deltapvector(1,i-1);
end

% 5. Plot the values using own colormap

red=linspace(0,0,stepcount)';
green=linspace(0.6,0.6,stepcount)';
blue=linspace(0.7,0.7,stepcount)';

% Explanation: Colormap is Matrix with values between 0 and 1.
% Dimensions: (stepcount, 3) ; first dimension: for each point in time one
% entry. Also second dimension contains r,g,b values for every point in time
% After initialization, unite three seperate vectors into single
% colormapmatrix.

colors_greenblue_spec=[red green blue];
colormap(colors_greenblue_spec);


scatter(timevector,processvaluevector,3,1:stepcount,'filled');

% Explanation: Label the scatterplot. sprintf command is used because the
% value of the variable sigma should be incorporated into the title-string.

xlabel('Time');
ylabel('Process Value')
titlename=sprintf('Wiener process with mu = 0 and sigma = %d', sigma);
title(titlename)
```

Resulting algorithm

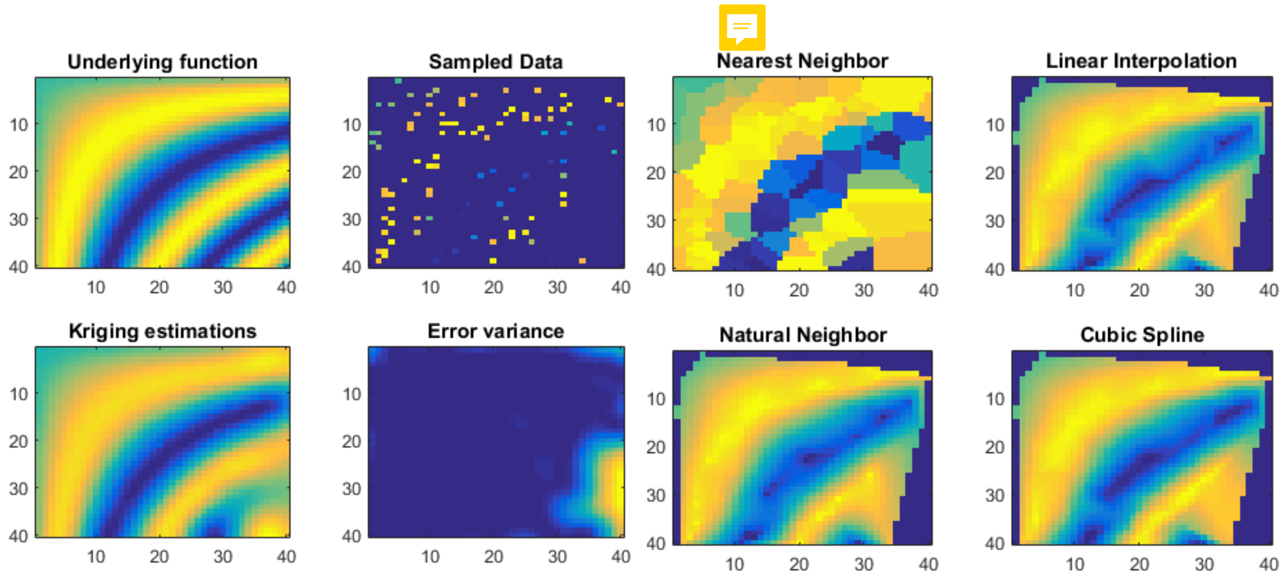


Resulting algorithm



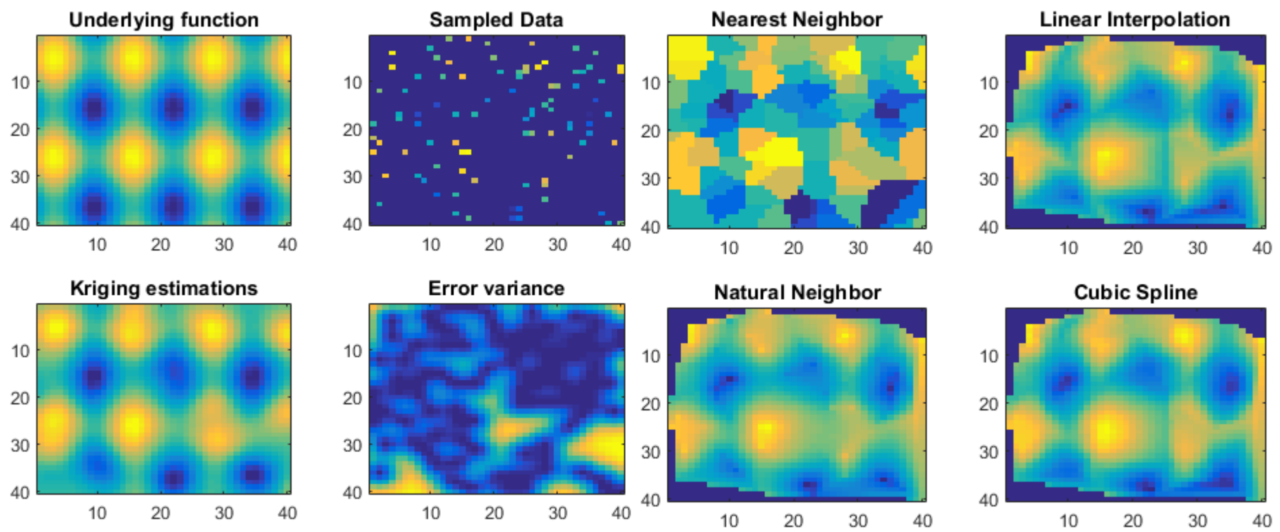


Resulting algorithm



ETH zürich

Resulting algorithm



WARNING: Garbage in, garbage out!