

Low-Rank Representation Numerical Methods for Uncertainty Quantification

Hermann G. Matthies*

Alexander Litvinenko*, Mike Espig†, Dishi Liu*, Elmar Zander*

* Institute of Scientific Computing, TU Braunschweig
Brunswick, Germany

† Max Planck Institute for Mathematics in the Sciences
Leipzig, Germany

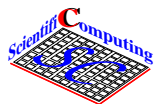
* `wire@tu-bs.de`

* `http://www.wire.tu-bs.de`

† `http://www.mis.mpg.de`

Overview

1. Tensors are natural
2. Solution in tensor product space
3. Model reduction and sparse representation
4. Low rank representation and algorithms
5. Examples and conclusion



Parametric problems and covariance

For a function $w : \Omega \rightarrow \mathcal{U}$ (Hilbert), there is a **linear** map $W \in \mathcal{L}(\mathcal{U}, \mathcal{S})$:

$$W : \mathcal{U} \ni v \mapsto \langle w(\cdot) | v \rangle_{\mathcal{U}},$$

with a Hilbert subspace $\mathcal{S} \subset \mathbb{R}^{\Omega}$.

If $(\Omega, \mathbb{P}, \mathfrak{A})$ is **measure** space, \mathbb{P} used to define \mathcal{S} .

Covariance $C := W^*W \in \mathcal{L}(\mathcal{U}, \mathcal{U})$ is symmetric (self-adjoint),
positive semi-definite \Rightarrow has **spectrum** $\sigma(C) \subseteq \mathbb{R}_+$,
spectral decomposition with **projectors** E_{λ}

$$Cv = \int_0^{\infty} \lambda dE_{\lambda}v = \sum_{\lambda_j \in \sigma_p(C)} \lambda_j \langle e_j | v \rangle_{\mathcal{U}} e_j + \int_{\mathbb{R}_+ \setminus \sigma_p(C)} \lambda dE_{\lambda}v.$$

\mathcal{U} with inner product from **kernel** C can be made into a **RKHS**.

Karhunen-Loève-expansion and tensors

Often C **compact** \Rightarrow last integral vanishes

$$C = \sum_j \lambda_j \langle e_j | \cdot \rangle_{\mathcal{U}} e_j = \sum_j \lambda_j e_j \otimes e^j.$$

C compact $\Rightarrow W$ compact $\Rightarrow \exists s_j \in \mathcal{S}$

$$(Wv)(\omega) = \langle w(\omega) | v \rangle_{\mathcal{U}} = \sum_j \sqrt{\lambda_j} \langle e_j | v \rangle_{\mathcal{U}} s_j(\omega)$$

or

$$W = \sum_j \sqrt{\lambda_j} (s_j \otimes e^j), \quad w(\omega) = \sum_j \sqrt{\lambda_j} s_j(\omega) e_j, \quad w \in \mathcal{S} \otimes \mathcal{U}.$$

The **singular value decomposition**, a.k.a. **Karhunen-Loève**-expansion.

A sum of **rank-1** operators / **tensors**.

Deterministic model, discretisation, solution

Consider operator equation, physical **system** modelled by A :

$$A(u) = f \quad u \in \mathcal{U}, f \in \mathcal{F},$$

$$\Leftrightarrow \forall v \in \mathcal{U} : \quad \langle A(u), v \rangle = \langle f, v \rangle,$$

\mathcal{U} — space of **states**, \mathcal{F} — dual space of **actions / forcings**.

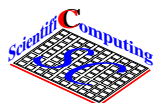
Solution is usually by first **discretisation**

$$\mathbf{A}(\mathbf{u}) = \mathbf{f} \quad \mathbf{u} \in \mathcal{U}_N \subset \mathcal{U}, \mathbf{f} \in \mathcal{F}_N \subset \mathcal{F},$$

and then **(iterative)** numerical **solution** process

$$\mathbf{u}_{k+1} = \Phi(\mathbf{u}_k), \quad \lim_{k \rightarrow \infty} \mathbf{u}_k = \mathbf{u}.$$

Φ evaluates (pre-conditioned) **residua** $\mathbf{f} - \mathbf{A}(\mathbf{u}_k)$.



Model with uncertainties

With **uncertainties** modelled by appropriate probab. space $(\Omega, \mathbb{P}, \mathfrak{A})$:

$$A[\omega](u(\omega)) = f(\omega) \quad \text{a.s. in } \omega \in \Omega,$$

State $u(\omega)$ is \mathcal{U} -valued **random variable** (RV),
solution is in a **tensor** space $\mathcal{W} = \mathcal{S} \otimes \mathcal{U}$.

Variational statement: $\forall v \in \mathcal{W} : \quad \mathbb{E}(\langle A[\cdot](u(\cdot)), v \rangle) = \mathbb{E}(\langle f(\cdot), v \rangle)$.

Similarly after semi-discretisation in \mathcal{U} :

$$\mathbf{A}[\omega](\mathbf{u}(\omega)) = \mathbf{f}(\omega) \quad \text{a.s. in } \omega \in \Omega,$$

assume $\{\mathbf{v}_j\}_{j=1}^N$ a basis in \mathcal{U}_N , then the approx. solution in $\mathcal{S} \otimes \mathcal{U}_N$

$$\mathbf{u}(\omega) = \sum_{j=1}^N u_j(\omega) \mathbf{v}_j.$$

Direct integration / sampling solution

Builds on fact that ultimately a quantity of interest $\mathbb{E}(\Psi(u))$ is wanted.

$$\mathbb{E}(\Psi(u)) = \int_{\Omega} \Psi(\omega, u(\omega)) \mathbb{P}(d\omega) \approx \sum_{z=1}^Z w_z \Psi(\omega_z, u(\omega_z))$$

Pick (e.g. **Monte Carlo**) $\{\omega_z\}_{z=1}^Z$ points, $\forall \omega_z$ do solution process

$$\mathbf{u}_{k+1}(\omega_z) = \Phi[\omega_z](\mathbf{u}_k(\omega_z)),$$

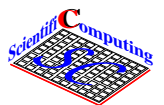
$$\text{giving } \mathbf{u}(\omega_z) = \sum_j u_j(\omega_z) \mathbf{v}_j = \sum_j u_j^z \mathbf{v}_j.$$

Effectively choosing $\mathcal{S}_Z \subset \mathcal{S}$, solution is in $\mathcal{W}_{Z,N} := \mathcal{S}_Z \otimes \mathcal{U}_N$.

(Usually $\mathbf{u}(\omega_z)$ discarded after use in integration.)

Random state represented by **solution samples** $[\mathbf{u}(\omega_0), \dots, \mathbf{u}(\omega_Z)]$,

$$\text{or the **tensor** } \mathbf{u}_N^Z := \{u_j^z\}_{j=1, \dots, N}^{z=1, \dots, Z}.$$



Solution by emulation

Emulation — replace **expensive** simulation by **inexpensive** approximation
(alias response surfaces, proxy / surrogate models, etc.)

Choose subspace $\mathcal{S}_B \subset \mathcal{S}$ with basis $\{X_\beta\}_{\beta=1}^B$,
make **ansatz** for each $u_j(\omega) \approx \sum_{\beta} u_j^\beta X_\beta(\omega)$, giving

$$\mathbf{u}(\omega) = \sum_{j,\beta} u_j^\beta X_\beta(\omega) \mathbf{v}_j = \sum_{j,\beta} u_j^\beta X_\beta(\omega) \otimes \mathbf{v}_j.$$

Solution is in **tensor product** $\mathcal{W}_{B,N} := \mathcal{S}_B \otimes \mathcal{U}_N \subset \mathcal{S} \otimes \mathcal{U} = \mathcal{W}$.

Random state $\mathbf{u}(\omega)$ represented by **tensor** $\mathbf{u}_N^B := \{u_j^\beta\}_{j=1,\dots,N}^{\beta=1,\dots,B}$,
computed by **sampling** (pre-conditioned) **residual** $\mathbf{f}(\omega) - \mathbf{A}[\omega](\mathbf{u}_k(\omega))$.

Tensor product structure

Story does **not end** here as one may choose $\mathcal{S} = \bigotimes_m \mathcal{S}_m$,
 approximated by $\mathcal{S}_B = \bigotimes_{m=1}^M \mathcal{S}_{B_m}$ with $\mathcal{S}_{B_m} \subset \mathcal{S}_m$.

Solution represented as a tensor of **grade** $M + 1$
 in $\mathcal{W}_{B,N} = \left(\bigotimes_{m=1}^M \mathcal{S}_{B_m} \right) \otimes \mathcal{U}_N$.

For higher grade tensor product structure, more reduction is possible,
 — but that is a story for **another** talk, here we stay with $M = 1$.

Sparse representation entails

- **reduce** $\mathbf{u}_N^B := [u_j^\beta]$ to **important** information $\mathbf{u} \approx \mathbf{u}_N^B$,
- **never** store all of \mathbf{u}_N^B , but only \mathbf{u} ,
- operate **efficiently** on sparse representation \mathbf{u} .

Discretisation — model reduction

On **continuous** level **discretisation** is choice of subspace

$$\mathcal{W}_{B,N} := \mathcal{S}_B \otimes \mathcal{U}_N \subset \mathcal{S} \otimes \mathcal{U} = \mathcal{W}$$

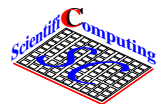
and—**important for computation**—basis in it.

On **discrete** level **reduced models** find subspace $\mathcal{W}_R \subset \mathcal{W}_{B,N}$
with **smaller** dimensionality $\dim \mathcal{W}_R = R \ll B \times N$.

They can work on \mathcal{S}_B or \mathcal{U}_N , or both.

Different approaches to **choose** reduced model:

- **Before** the solution process (e.g. proper generalised decomposition).
- **After** the solution process (essentially **data compression**).
- **During** solution, computing solution and reduction **simultaneously**.



Low-rank approximation

Focus on **array** of numbers $\mathbf{u}_N^B := [u_j^\beta]$, view as **matrix / tensor**:

$$\sum_{\beta=1}^B \sum_{j=1}^N u_j^\beta \mathbf{e}_\beta \otimes \mathbf{e}^j$$

with canonical unit vectors $\mathbf{e}_\beta, \mathbf{e}^j$.

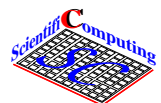
The sum has $B * N$ terms, the number of entries in \mathbf{u}_N^B .

Rank- R representation is approximation with R terms

$$\sum_{\beta=1}^B \sum_{j=1}^N u_j^\beta \mathbf{e}_\beta \otimes \mathbf{e}^j \approx \sum_{\ell=1}^R \mathbf{y}_\ell \otimes \mathbf{w}^\ell$$

It contains only $R * (B + N) \ll B * N$ numbers.

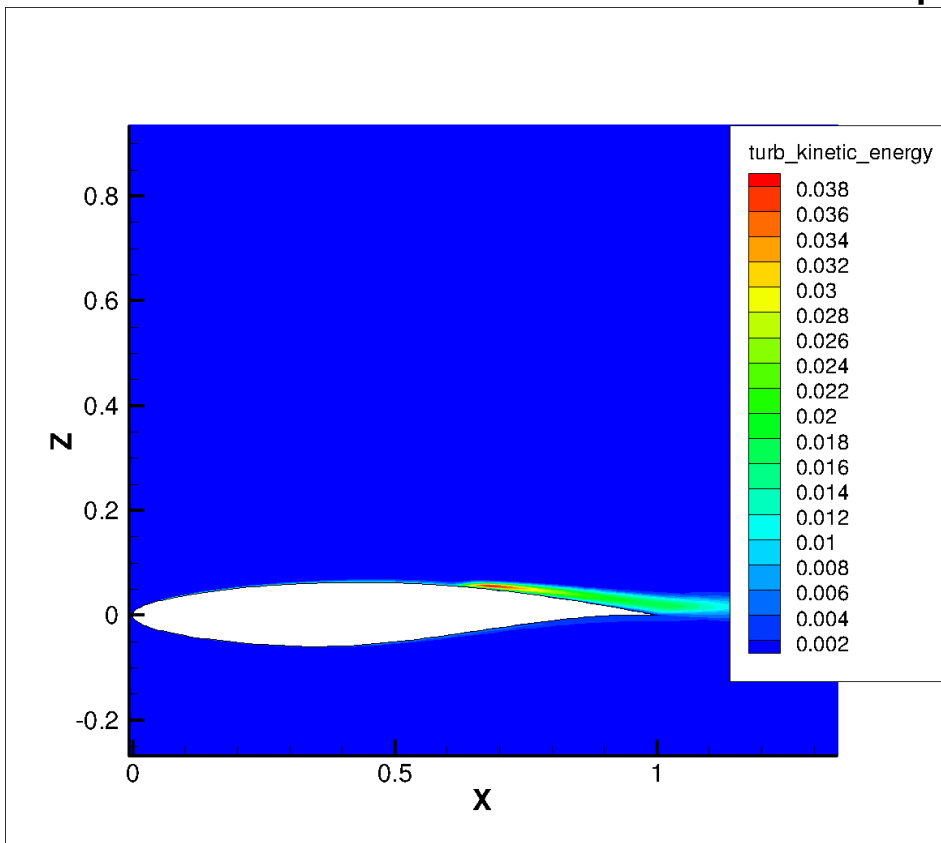
We will use **truncated SVD**.



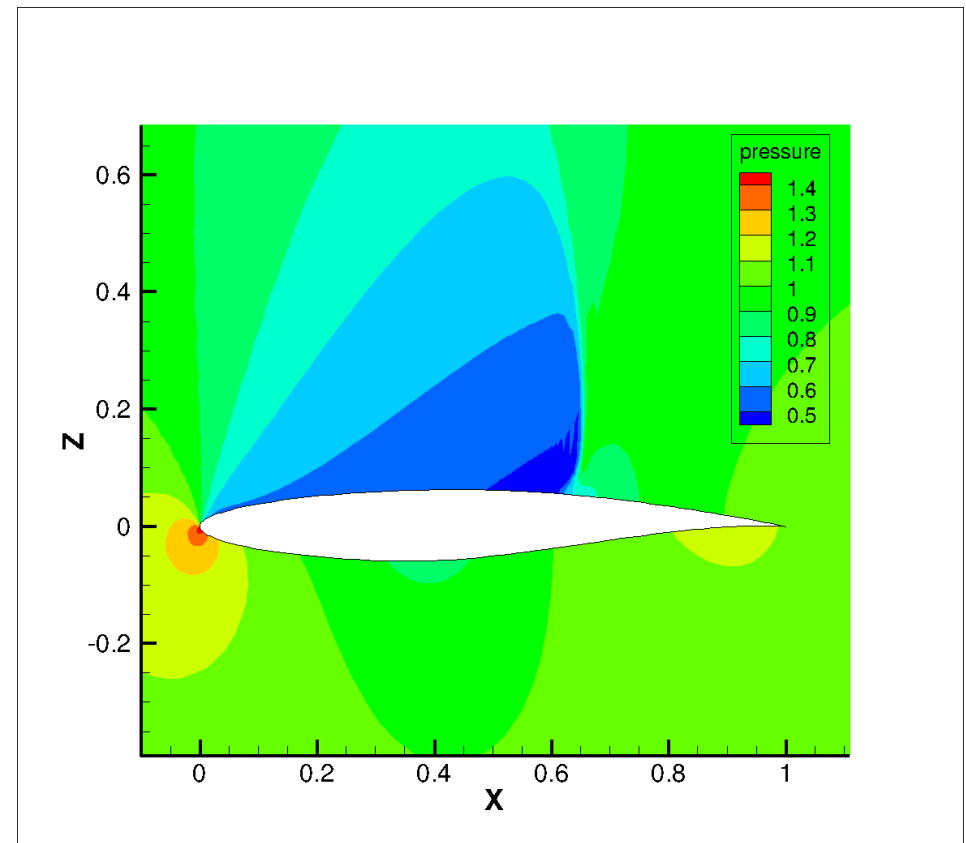
Use in UQ-MC sampling solution I

Example: Compressible **RANS-flow** around RAE air-foil.

Sample solution



turbulent kinetic energy



pressure

Use in UQ-MC sampling solution II

Inflow and air-foil shape uncertain.

Data compression achieved by updated SVD:

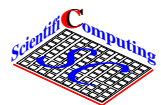
Made from 600 MC Simulations, SVD is updated every 10 samples.

$$N = 260,000 \quad Z = 600$$

Updated SVD: Relative errors, memory requirements:

| rank R | pressure | turb. kin. energy | memory [MB] |
|----------|----------|-------------------|-------------|
| 10 | 1.9e-2 | 4.0e-3 | 21 |
| 20 | 1.4e-2 | 5.9e-3 | 42 |
| 50 | 5.3e-3 | 1.5e-4 | 104 |

Dense matrix $\in \mathbb{R}^{260000 \times 600}$ costs 1250 MB storage.



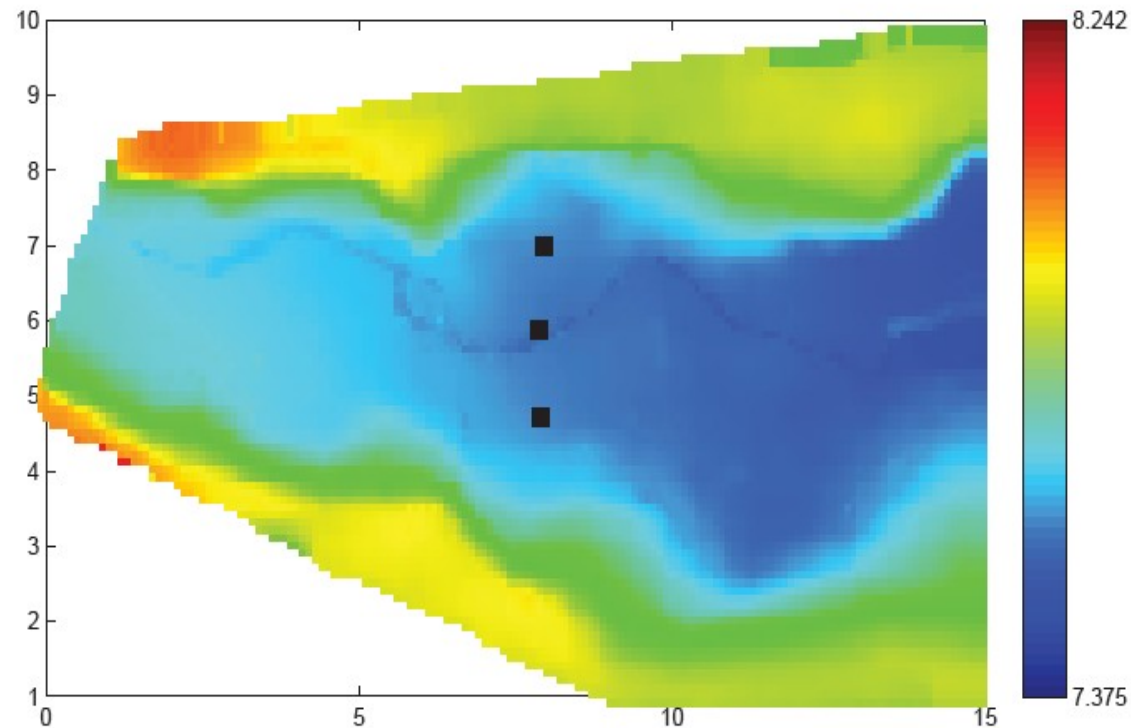
Use in time-space sampling I

Example: UQ-computations of **time-dependent shallow-water flow**.
1:50 Scale model in Milano of Toce river (Italy)



Use in time-space sampling II

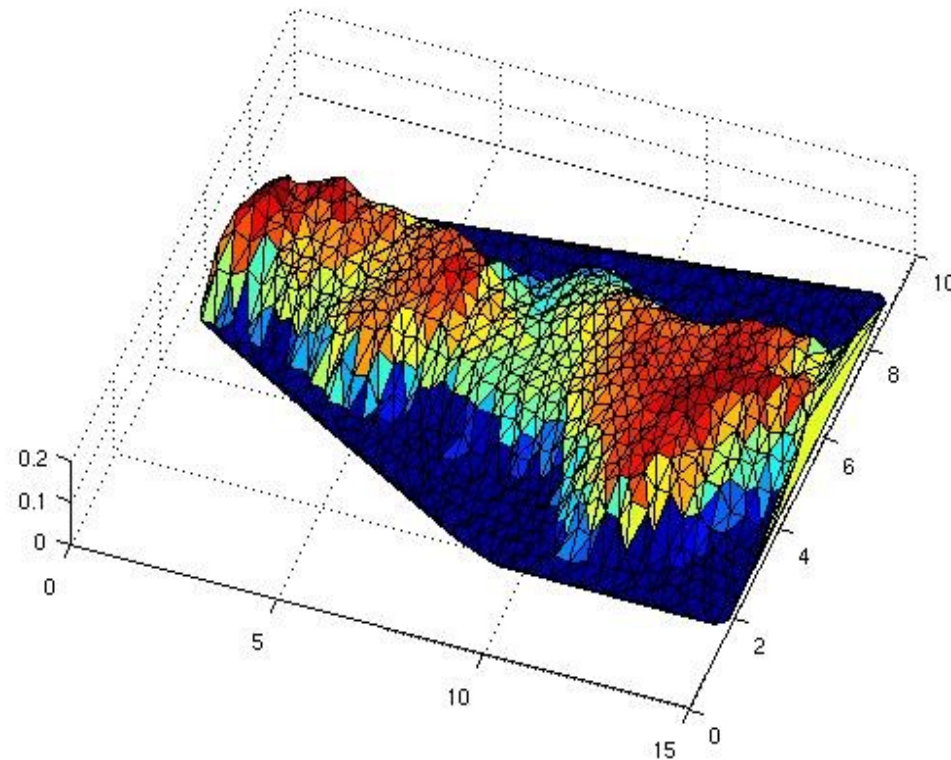
Topography in model — uncertain elevation.



Also uncertain **inflow** and **bed friction**—**Manning's** coefficient.

Use in time-space sampling III

Computation with QMC-sampling
Water level with 5 % exceedance probability



Use in emulation

Solution process to obtain co-efficients for stochastic problem

$$\mathbf{u}_{k+1} = \Phi(\mathbf{u}_k)$$

may be written as tensorised mapping

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \Xi(\mathbf{u}_k) = \mathbf{u}_k - \left(\sum_{m=1}^M \mathbf{Y}_m \otimes \mathbf{G}^m \right) (\mathbf{u}_k).$$

With $\mathbf{u}_0 = \sum_{j=1}^{R_0} \mathbf{y}_{0,j} \otimes \mathbf{g}^{0,j}$, this gives

$$\mathbf{u}_1 = \sum_{j=1}^{R_0} \mathbf{y}_{0,j} \otimes \mathbf{g}^{0,j} - \sum_{m=1}^M \mathbf{Y}_m(\mathbf{u}_0) \otimes \mathbf{G}^m(\mathbf{u}_0).$$

Rank of \mathbf{u}_{k+1} grows by M .

Possible for pre-conditioned linear iteration,
and modified-, full-, inexact- and quasi-Newton iteration.

Truncated iteration

If **iteration** and **rank-truncation** \mathbf{T}_ϵ are **alternated**, rank stays low.

Here rank-truncation by updated SVD.

$$\hat{\mathbf{u}}_{k+1} = \sum_{j=1}^{R_k} \mathbf{y}_{k,j} \otimes \mathbf{g}^{k,j} - \sum_{m=1}^M \mathbf{Y}_m(\mathbf{u}_k) \otimes \mathbf{G}^m(\mathbf{u}_k),$$

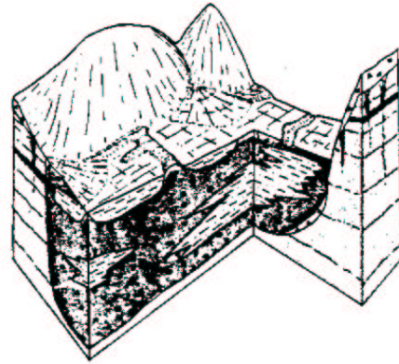
$$\mathbf{u}_{k+1} = \mathbf{T}_\epsilon(\hat{\mathbf{u}}_{k+1}) \quad \text{with} \quad \|\mathbf{T}_\epsilon(\mathbf{v}) - \mathbf{v}\| \leq \epsilon.$$

It can be **shown** that truncated iteration **converges** until **stagnation** for

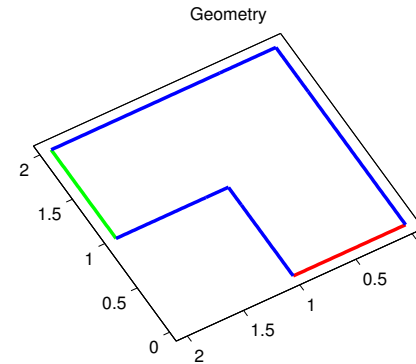
Theorem [Hackbusch, Khoromskij, Tyrtysnikov] super-linearly convergent iterative process with stagnation range 2ϵ ,

Theorem [Zander, M.] linearly convergent process with contraction factor ϱ and stagnation range $\epsilon/(1 - \varrho)$.

Stochastic Galerkin for diffusion equation



Aquifer



2D Model

Simple stationary model of groundwater flow with stochastic data

$$-\nabla \cdot (\kappa(x, \omega) \nabla u(x, \omega)) = f(x, \omega) \quad x \in \mathcal{D} \subset \mathbb{R}^d \quad \& \text{ b.c.}$$

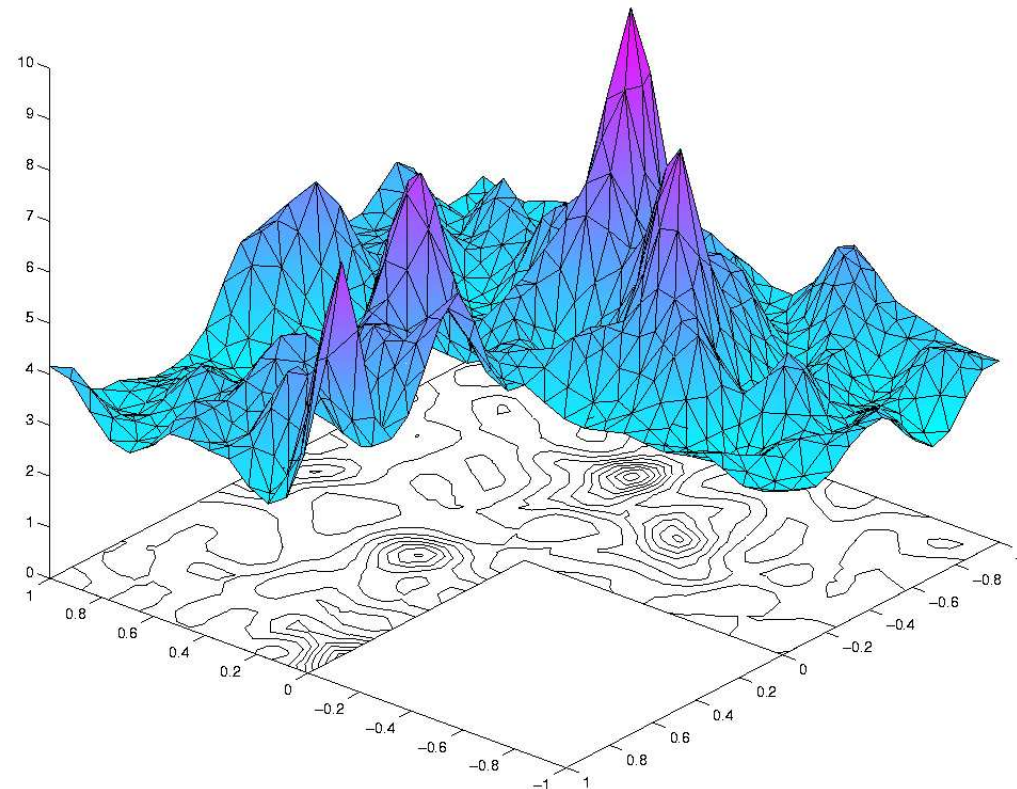
leads after Galerkin discretisation on

$$\mathcal{W}_{B,N} = \mathcal{S}_B \otimes \mathcal{U}_N \subset \mathcal{S} \otimes \mathcal{U} = \mathcal{W} = L_2(\Omega, \mathbb{P}) \otimes \dot{H}^1(\mathcal{D}) \text{ to}$$

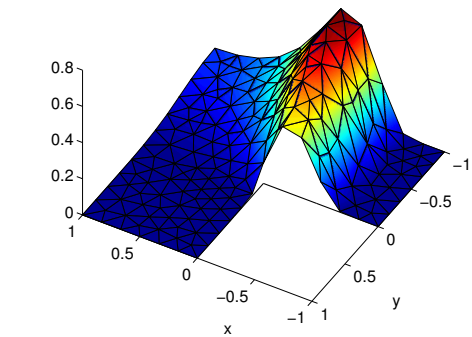
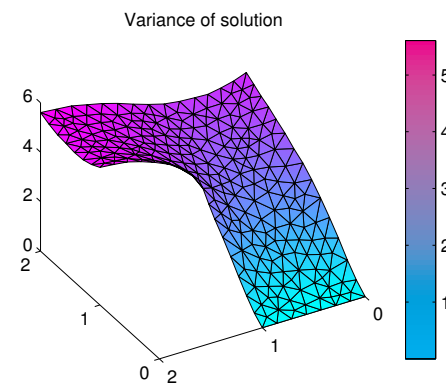
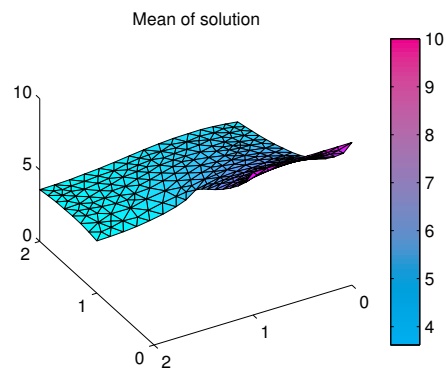
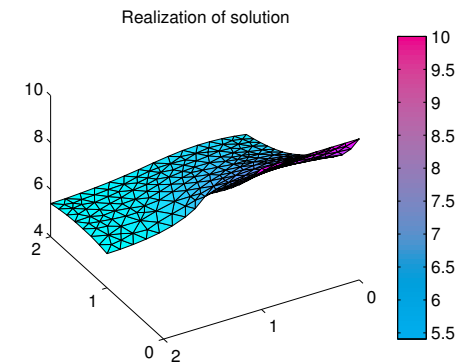
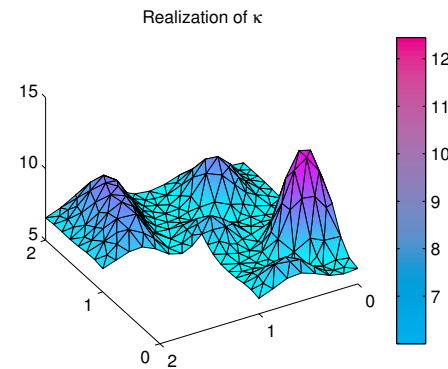
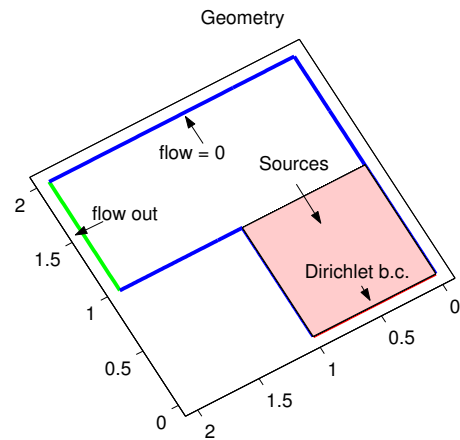
$$\mathbf{A} \mathbf{u} = \left(\sum_{m=1}^M \xi_m \Delta^{(m)} \otimes \mathbf{A}_m \right) \mathbf{u} = \mathbf{f}$$

Realisation of $\kappa(x, \omega)$ — β -distributed

A sample realization



Example solution



$$\Pr\{u(x) > 8\}$$

Computational complexity for linear case

Residuum is $\mathbf{f} - \mathbf{A} \mathbf{u}_k = \mathbf{f} - \left(\sum_{m=1}^M \xi_m \Delta^{(m)} \otimes \mathbf{A}_m \right) \mathbf{u}_k$.

Computation on **full** \mathbf{u}_k needs

$M \times B$ \mathbf{A} -multiplications + $M \times N$ Δ -multiplications.

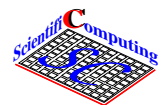
Computation on **low rank- R tensor product** \mathbf{u}_k needs

$M \times R$ \mathbf{A} -multiplications + $M \times R$ Δ -multiplications,
which is **much less**.

Pre-conditioner \mathbf{P} should be used as $\mathbf{P} = \sum_{p=1}^P \Lambda^{(p)} \otimes \mathbf{P}_p$.

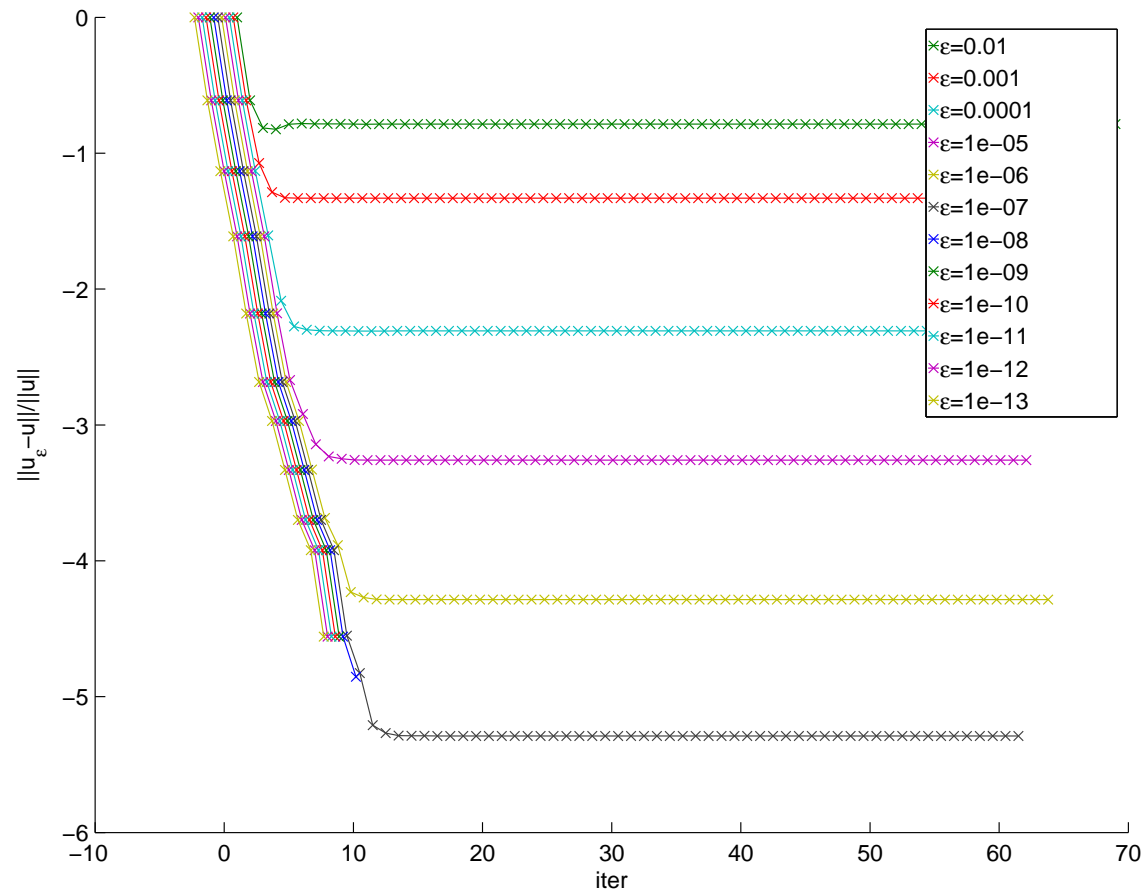
Simple example: mean value pre-conditioner $\mathbf{P} = \mathbf{I} \otimes \mathbf{P}_0$
with \mathbf{P}_0 pre-conditioner for $\mathbb{E}(\mathbf{A})$.

Similar **savings** as before, with M **replaced** by P .



Iteration accuracy

Convergence of truncated iteration.



Conclusion

- Stochastic calculations produce **huge** amount of data.
- For efficiency try and use **sparse** representation throughout: ansatz in **low-rank** tensor products, as well as storage of solution and residuum—and iterator in tensor products.
- Works in sampling and emulation.
- Works also for non-linear problems and solvers.
- Works also for time-dependent problems.

