Reduced Basis Methods for Parametric PDEs

Jonathan Roth

Motivation RBM

speed up computation of inverse problems

Classical FEM

weak form of parametric PDE

$$b(u(\mu), v; \mu) = f(v; \mu) \qquad \forall v \in X$$

a grid with n DOFs and hat functions gives

$$KD = F \text{ with } K \in \mathbb{R}^{n \times n}$$

for **one** specific parameter set

RBM Idea I

parameter samples $S_N := \{\mu^1, \ldots, \mu^N\}$

$\downarrow$

solution snapshots $U_N := \{u^1, \ldots, u^N\}$

$\downarrow$

reduced basis space $X_N := \mathrm{span}(\{u^i\}_{i=1}^N)$

$\downarrow$

reduced basis (RB) $\Phi_N := \{\varphi_1, \ldots, \varphi_N\}$

RBM Idea II

$$b(u_N(\mu), v; \mu) = f(v; \mu) \qquad \forall v \in X_N$$

no hat functions...

...use a linear combination of basis vectors

$$K_N(\mu) D_N = F_N(\mu) \qquad K_N \in \mathbb{R}^{N \times N}$$

$$u_N(\mu) = \sum_{i=1}^{N} (D_N)_i \varphi_i$$

How to get the RB

- **Lagrange**
  snapshots are basis itself

- **Proper Orthogonal Decomposition**
  - orthogonal
  - hierarchical
    → truncation possible, k < N < n

$$S_N \longrightarrow U_N \longrightarrow X_N \longrightarrow \Phi_N$$

How to get the snapshots

- **forward run on parameter samples**
- samples can be drawn
  - dumb
    → uniform
  - intelligent
    → problem specific

$$S_N \downarrow U_N \downarrow X_N \downarrow \Phi_N$$

The Thermal Problem

- square domain

$$\Omega = [0,1] \times [0,1]$$

- domain devided in reactangular blocks with different conductivities
  - B_1 in x_1 direction
  - B_2 in x_2 direction

$$nBlocks := B_1 * B_2$$

- conductivity is parameter

The Thermal Problem

problem is frequently used.

usual dimension is $\mathbf{B_1 = B_2 = 3}$.

we want to go bigger: $\mathbf{B_1 = B_2 = 10}$.

(much more takes too much computation time)

The Thermal Problem

$$
\begin{aligned}
-\lambda u_{,ii} = \dot{q}_{i,i} = 0 &\quad \text{on } \Omega \\
u = 0 &\quad \text{on the top boundary } \Gamma_{top} \\
(\lambda u_{,i})n_i = -\dot{q}_i n_i = 1 &\quad \text{on the bottom boundary } \Gamma_{base} \\
(\lambda u_{,i})n_i = -\dot{q}_i n_i = 0 &\quad \text{on all other boundaries of } \Omega
\end{aligned}
$$

... some maths ...

$$
b(u, v; \mu) = f(v; \mu) \qquad \forall v
$$

rbMIT

- RB software distributed by MIT
- symbolical calculations
- takes so much time, bores me and my computer

→ is there some other library?

→RBmatlab

RBmatlab validation

assume conductivity constant across domain

- solution should be linear
- solution should be exact
- solution should not depend on discretization

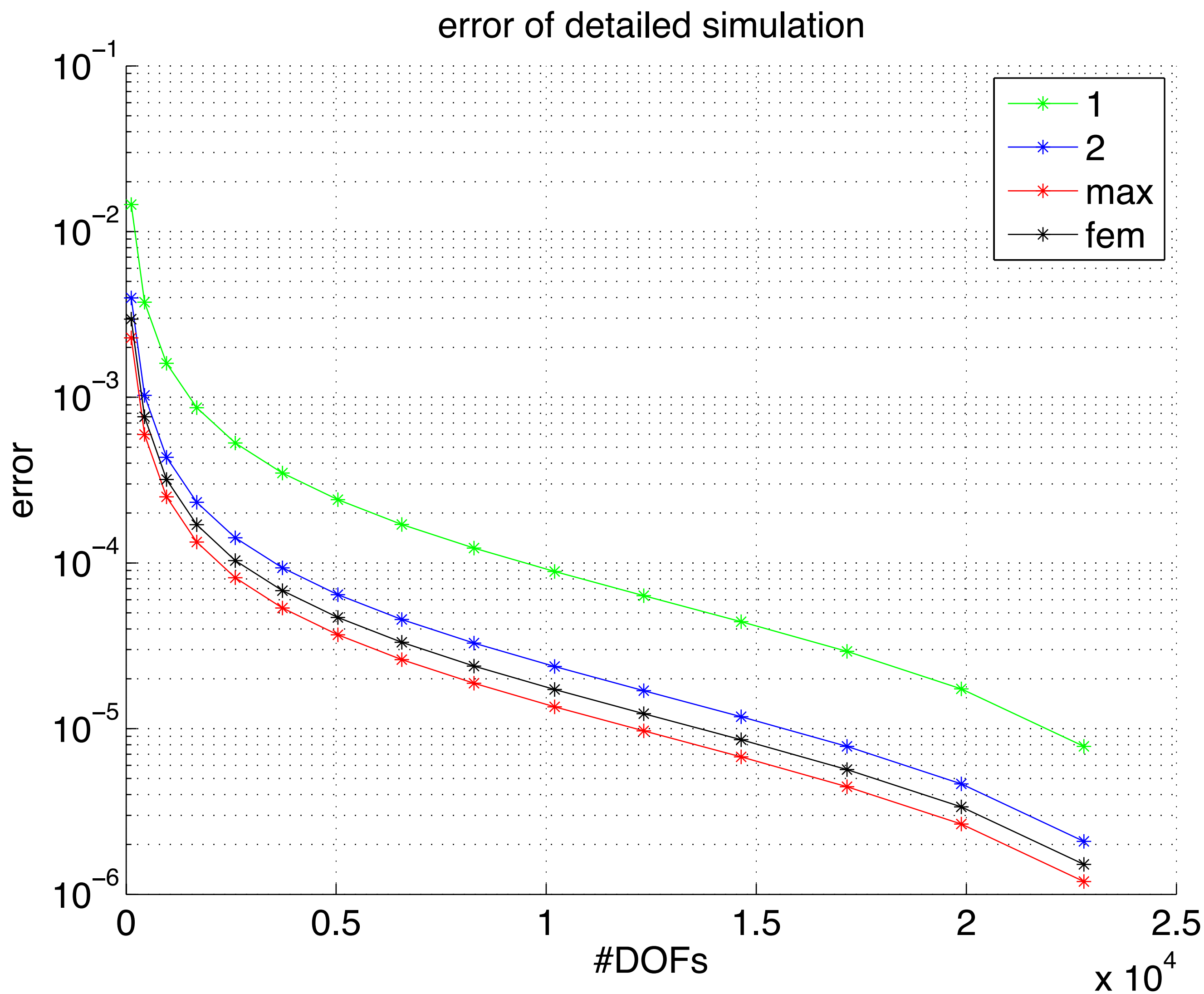
→ it does!

Detailed Discretization

domain	=	**10 x 10** blocks.
block	=	**nElems x nElems** elements.
element	=	**2** triangles.

error for **nElems = 10** is smaller than **1e-4**.

Discretization Error



Offline: Build Samples I

build N=1000 snapshots

try different sampling strategies...

- dumb (uniform)
- intelligent (log-normal)

Offline: Build Samples II

intelligent (log-normal)

→ use covariance function

$$\mathrm{X} = \exp(\mathrm{Y}) \qquad Y \sim \mathcal{N}(\mu, \Sigma)$$

$$[\Sigma]_{ij} = \sigma^2 * \exp\left(-\frac{d_{ij}^2}{a_0^2}\right)$$

Offline: Build Reduced Basis

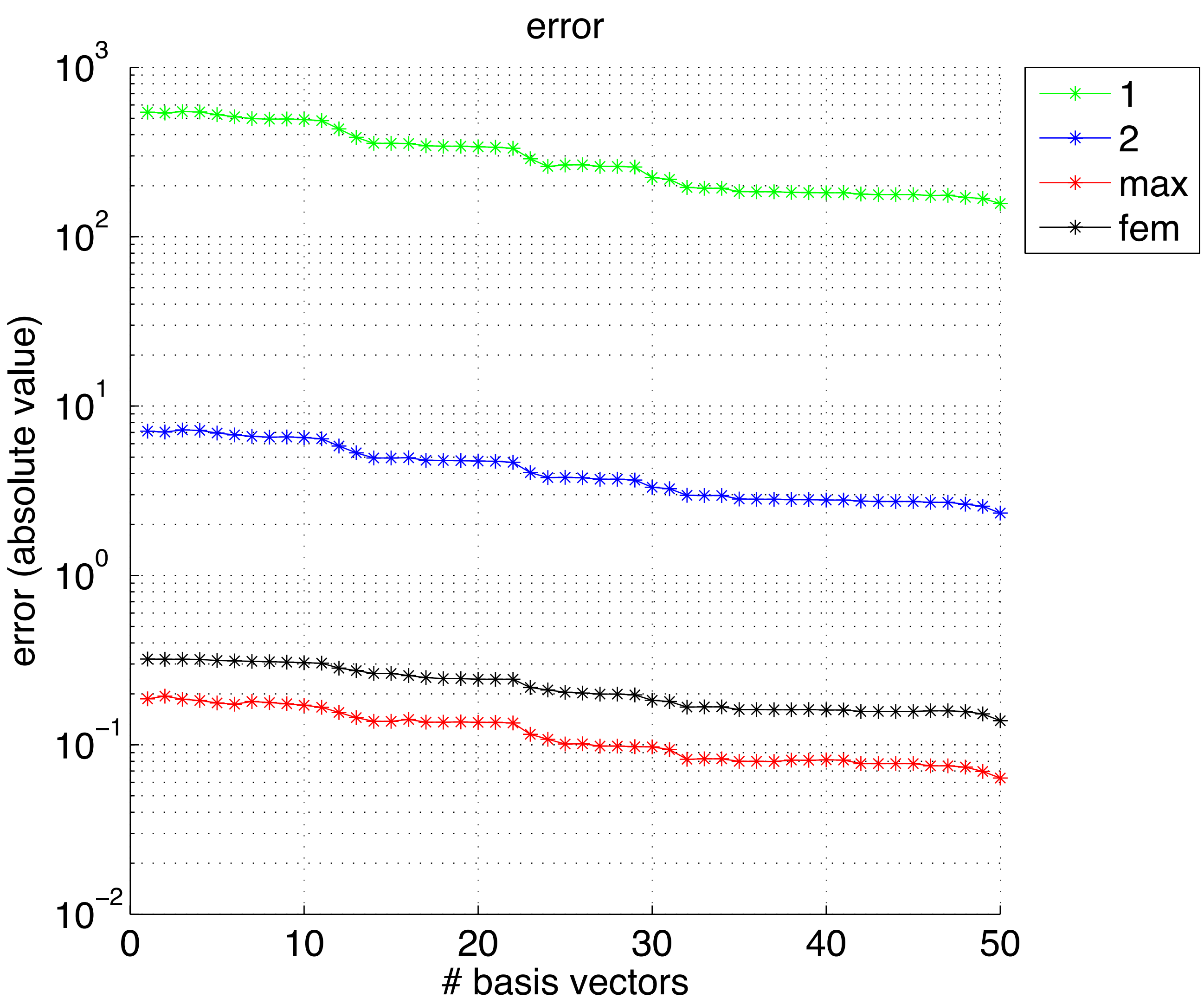
POD (truncation possible, hierarchical)

$n = 10201$, $N = 1000$

$N \times N$ is still huuuuge → truncate to $k = 50$.

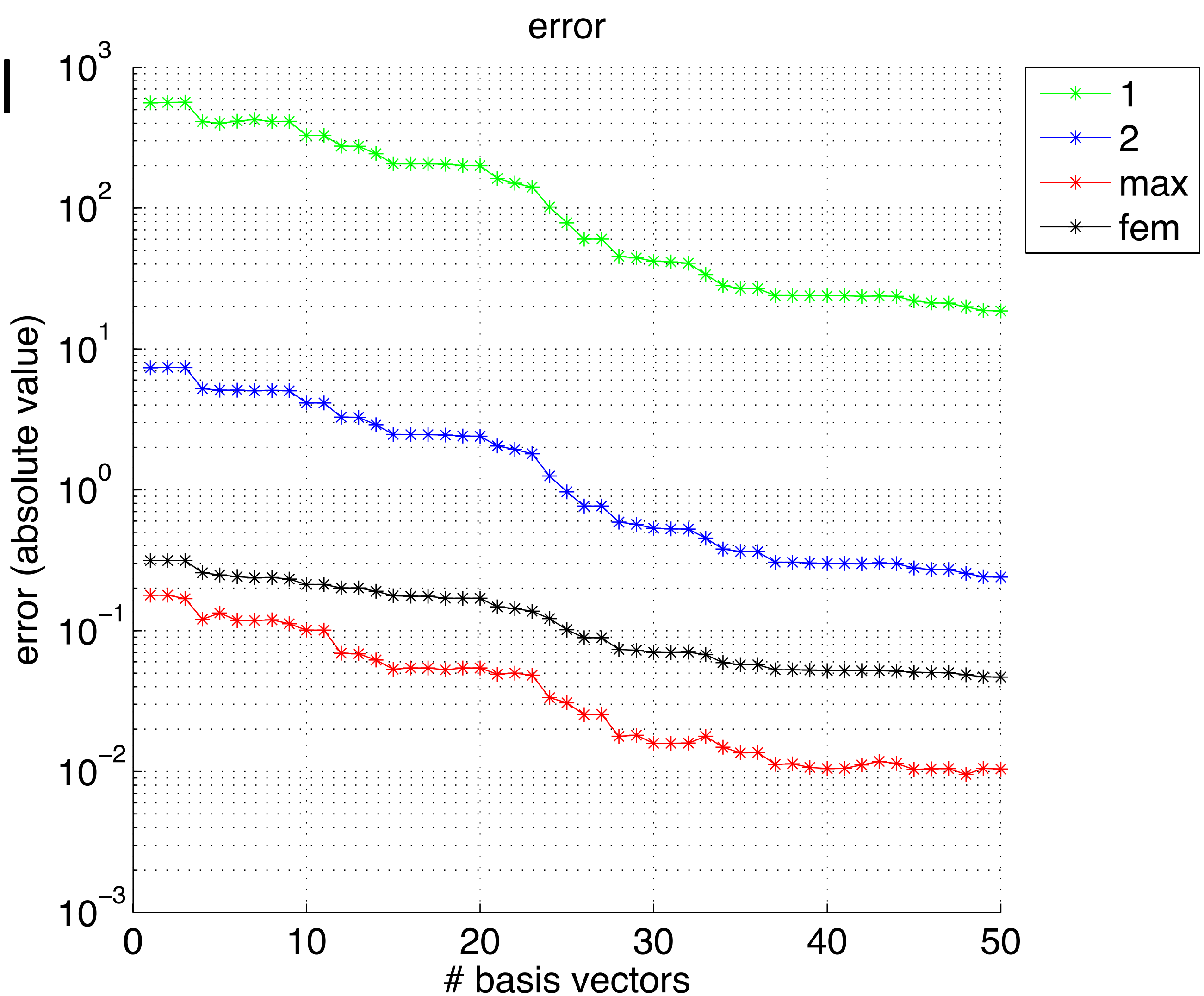
Online Results

uniform



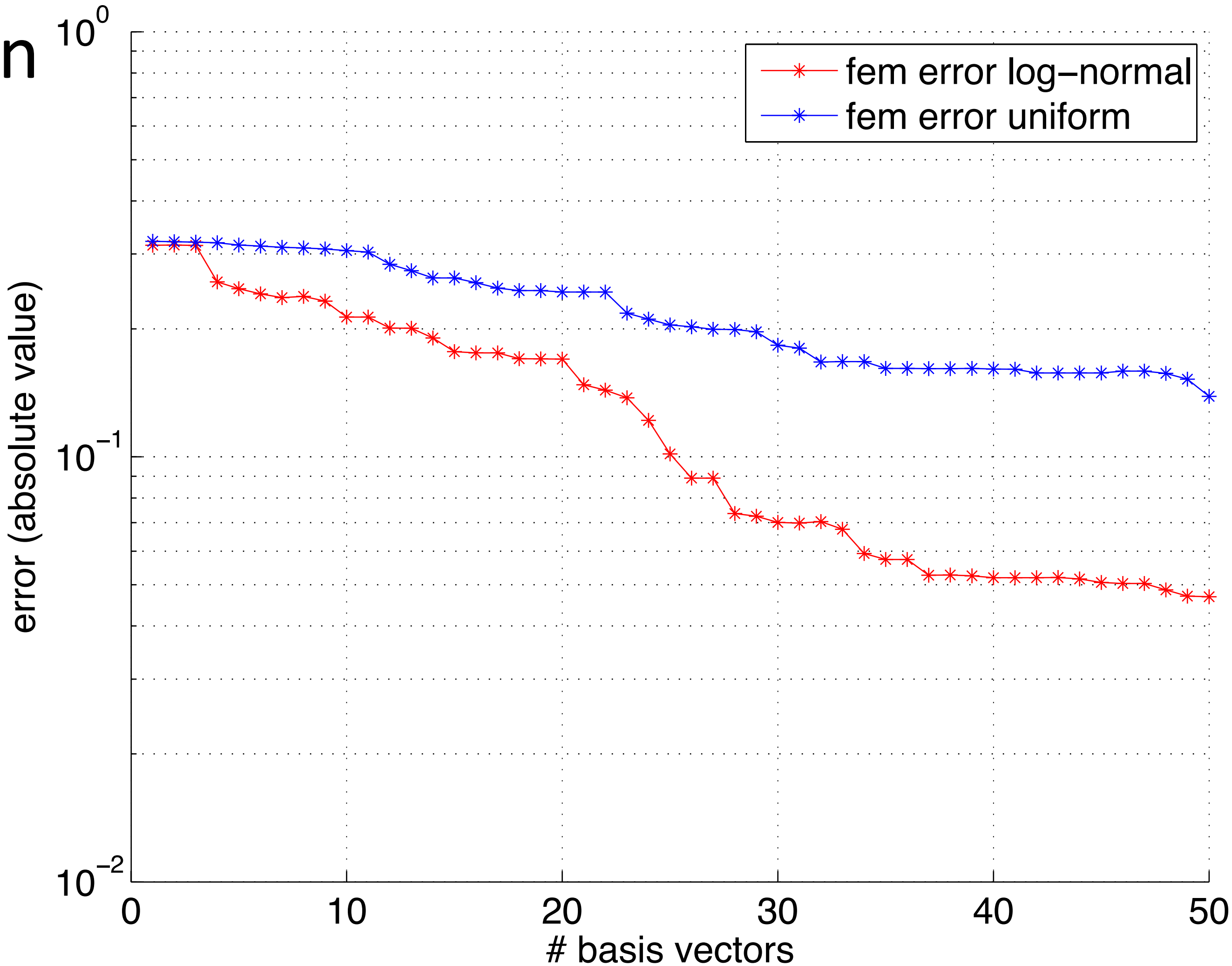
Online Results

log-normal



Online Results

comparison



Sampling revised I

- drawing random numbers as described is slow.
- thus, truncate the covariance matrix using POD as well and draw less random numbers.

$$\Sigma = U \Lambda U^T$$

$$\Sigma = U \Lambda^{1/2} \Lambda^{1/2} U^T = \hat{L} \hat{L}^T$$

Sampling revised II

$$\mathrm{X} = \exp(\mathrm{Y}) \qquad Y \sim \mathcal{N}(\mu, \Sigma)$$

$$Y = \mu + \hat{L}Z$$

$$Z = \begin{pmatrix} z_1 \\ \vdots \\ z_M \end{pmatrix} \text{ with } z_i \sim \mathcal{N}(0,1)$$

Sampling revised III

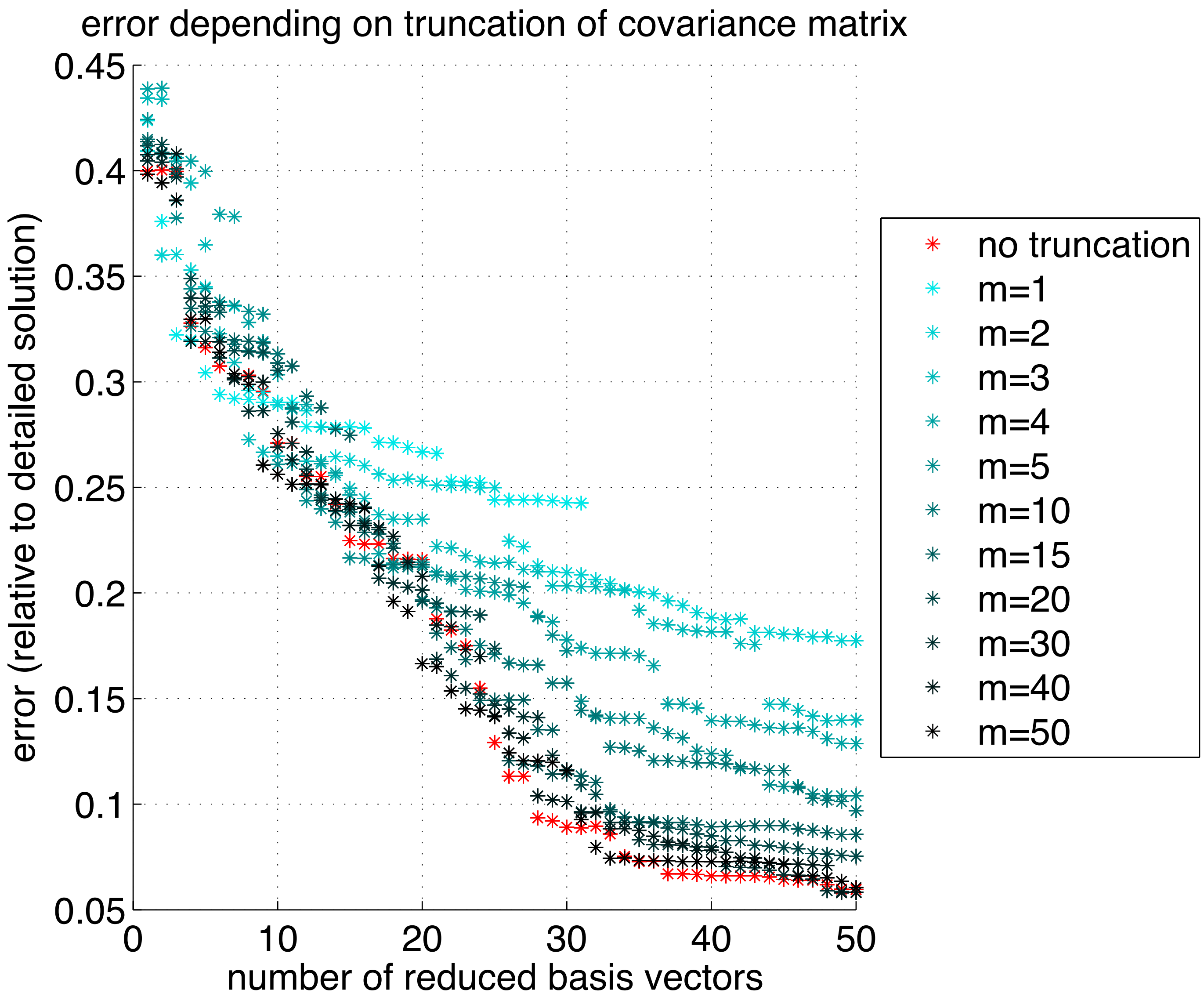
$$Y = \mu + \hat{L}Z$$

with **M = 100**, **m < M**:

$$Y = \mu + \sum_{i=0}^{M} u_i \sqrt{\lambda_i} z_i$$

$$\hat{Y} = \mu + \sum_{i=0}^{m} u_i \sqrt{\lambda_i} z_i$$

Truncated Sampling: Result



ToDo

- redo truncated sampling
  (error in rescaling of total variance)

- analyze the output variance
  (get to know how complicated the problem is)