Uncertainty quantification using tensor decompositions

Matthew Reynolds[1]

[1]Department of Aerospace Engineering Sciences, CU Boulder

NREL talk
Nov 2nd, 2016

Outline

1. Introduction to separated representations
2. Randomized alternating least squares
3. Optimization with canonical tensor decompositions
4. Non-intrusive algorithms

Outline

1. Introduction to separated representations
2. Randomized alternating least squares
3. Optimization with canonical tensor decompositions
4. Non-intrusive algorithms

Separated representations

Assume we have f (say in $L^2\left(\left[a,b\right]^d\right)$) that we want to approximate. A straightforward method of approximating this function is to use tensor product basis functions (e.g. a Fourier series),

$$f\left(z_1, z_2, \ldots z_d\right) \simeq \sum_{i_1=1}^{N_1} \cdots \sum_{i_d=1}^{N_d} c_{i_1,\ldots,i_d} \phi_{i_1}\left(z_1\right) \phi_{i_2}\left(z_j\right) \ldots \phi_{i_d}\left(z_d\right).$$

Bad idea! The number of coefficients $c_{i_1 \ldots i_d}$ grows exponentially w.r.t d. This is the curse of dimensionality.

Separated representations continued

- To mitigate computational difficulties associated with the curse of dimensionality, we work with functions with special *structures.*
- One such structure is that the function of interest admits a separated representation,

$$f(z_1, z_2, \ldots z_d) \simeq \sum_{l=1}^{r} \sigma_l \phi_1^l(z_1)\, \phi_2^l(z_2) \cdots \phi_d^l(z_d).$$

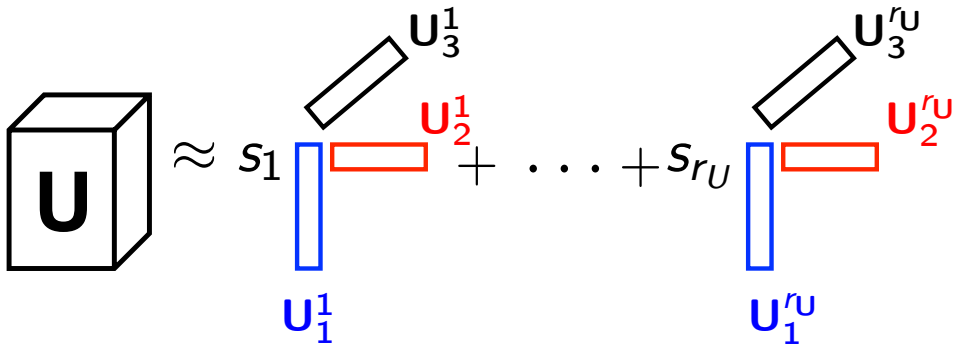
- The number of terms, r, is called the separation rank is assumed to be *small*.
- Given a separated representation,

$$u(z_1, z_2, \ldots z_d) = \sum_{l=1}^{r} \sigma_l u_1^l(z_1)\, u_2^l(z_2) \cdots u_d^l(z_d),$$

Canonical tensor decomposition

any discretization of $u_j^l(z_j)$, where $\mathbf{U}_j^l = \left\{ u_j^l\left(z_{i_j}\right) \right\}_{i_j=1}^{N_j}$, and $j = 1, \ldots, d$, leads to a Canonical Tensor Decomposition, or CTD,

$d = 3$:



$$\mathbf{U} = \sum_{l=1}^{r_\mathbf{U}} s_l^\mathbf{U} \mathbf{U}_0^l \otimes \mathbf{U}_1^l \otimes \cdots \otimes \mathbf{U}_d^l,$$

where $\otimes$ denotes the vector outer product and,

- $r_\mathbf{U}$ is the separation rank of $\mathbf{U}$.
- d is the number of dimensions.
- s_l are normalization constants s.t. $\left\| \mathbf{U}_j^l \right\|_2 = 1$.

CTDs: operations and storage

Standard operations with and storage of CTDs are linear in d, provided that the separation rank is independent of d.

Operation	**Definition in canonical format**	**Cost**
Storage	$\mathbf{U} = \sum_{l=1}^{r_\mathbf{U}} s_l \bigotimes_{j=1}^{d} \mathbf{U}_j^l$	$\mathcal{O}(d \cdot r_\mathbf{U} \cdot N)$
Inner product	$\langle \mathbf{U}, \mathbf{V} \rangle = \sum_{l=1}^{r_\mathbf{U}} \sum_{m=1}^{r_\mathbf{V}} s_l^\mathbf{U} s_m^\mathbf{V} \prod_{j=1}^{d} \left\langle \mathbf{U}_j^l, \mathbf{V}_j^m \right\rangle$	$\mathcal{O}(d \cdot r_\mathbf{U} \cdot r_\mathbf{V} \cdot N)$
Frobenius norm	$\|\mathbf{U}\|_F^2 = \langle \mathbf{U}, \mathbf{U} \rangle$	$\mathcal{O}(d \cdot r_\mathbf{U}^2 \cdot N)$
"Matrix-vector" multiply	$\mathbb{A}\mathbf{U} = \sum_{l=1}^{r_\mathbb{A}} \sum_{m=1}^{r_\mathbf{U}} s_l^\mathbb{A} s_m^\mathbf{U} \bigotimes_{j=1}^{d} A_j^l \mathbf{U}_j^m$	$\mathcal{O}(d \cdot r_A \cdot r_\mathbf{U} \cdot N^2)$

Operations on CTDs

Many operations on CTDs and separated representations increase the rank of the representation. Let

$$\mathbf{U} = \sum_{l=1}^{r_{\mathbf{U}}} s_l^{\mathbf{U}} \mathbf{U}_1^l \otimes \cdots \otimes \mathbf{U}_d^l \text{ and } \mathbf{V} = \sum_{m=1}^{r_{\mathbf{V}}} s_m^{\mathbf{V}} \mathbf{V}_1^m \otimes \cdots \otimes \mathbf{V}_d^m$$

denote CTDs with $\mathbf{U}_k^l, \mathbf{V}_k^m \in \mathbb{R}^{M_k}$. Then:

- The sum of $\mathbf{U} + \mathbf{V}$ has rank $r_{\mathbf{U}} + r_{\mathbf{V}}$.
- The Hadamard product $\mathbf{U} * \mathbf{V}$ has rank $r_{\mathbf{U}} r_{\mathbf{V}}$.
- Application of a d-dimensional operator of rank $r_{\mathbb{A}}$, $\mathbb{A}\mathbf{U}$, has rank $r_{\mathbb{A}} r_{\mathbf{U}}$.

Connection to UQ

A stochastic (s.p.d) linear system of equations

$$\mathbf{A}(y_1,\ldots,y_d)\mathbf{u}(y_1,\ldots,y_d)=\mathbf{b}, \qquad y_1,\ldots,y_d \overset{iid}{\sim} \rho(y),$$

admits a discrete representation

$$\mathbf{A}(y_1^{i_1},\ldots,y_d^{i_d})\mathbf{u}(y_1^{i_1},\ldots,y_d^{i_d})=\mathbf{b}, \qquad i_k=1,\ldots,M, \quad k=1,\ldots,d.$$

This can be written in tensor format

$$\mathbb{A}\,\mathbf{U}=\mathbf{B}.$$

We aim to approximate $\mathbf{U}$ in CTD format[1]

$$\mathbf{U}\approx\sum_{l=1}^{r_U} s_l \mathbf{U}_1^l\otimes\cdots\otimes\mathbf{U}_d^l.$$

[1] Nouy et al. 07,08,09,10,11,12,13; Doostan et al. 07,08,09,13,14; Khoromskij et al. 10; Cances et al. 11; Kressner et al. 11; Espig et al. 12,13; ...

Connection to UQ

To solve $\mathbb{A}\mathbf{U} = \mathbf{B}$ for $\mathbf{U}$, we use the convergent iteration

$$\mathbf{U}_{k+1} = (\mathbb{I} - \mathbb{A})\mathbf{U}_k + \mathbf{B}$$
$$\|\mathbb{I} - \mathbb{A}\| < 1.$$

The separation rank of the iterate grows as

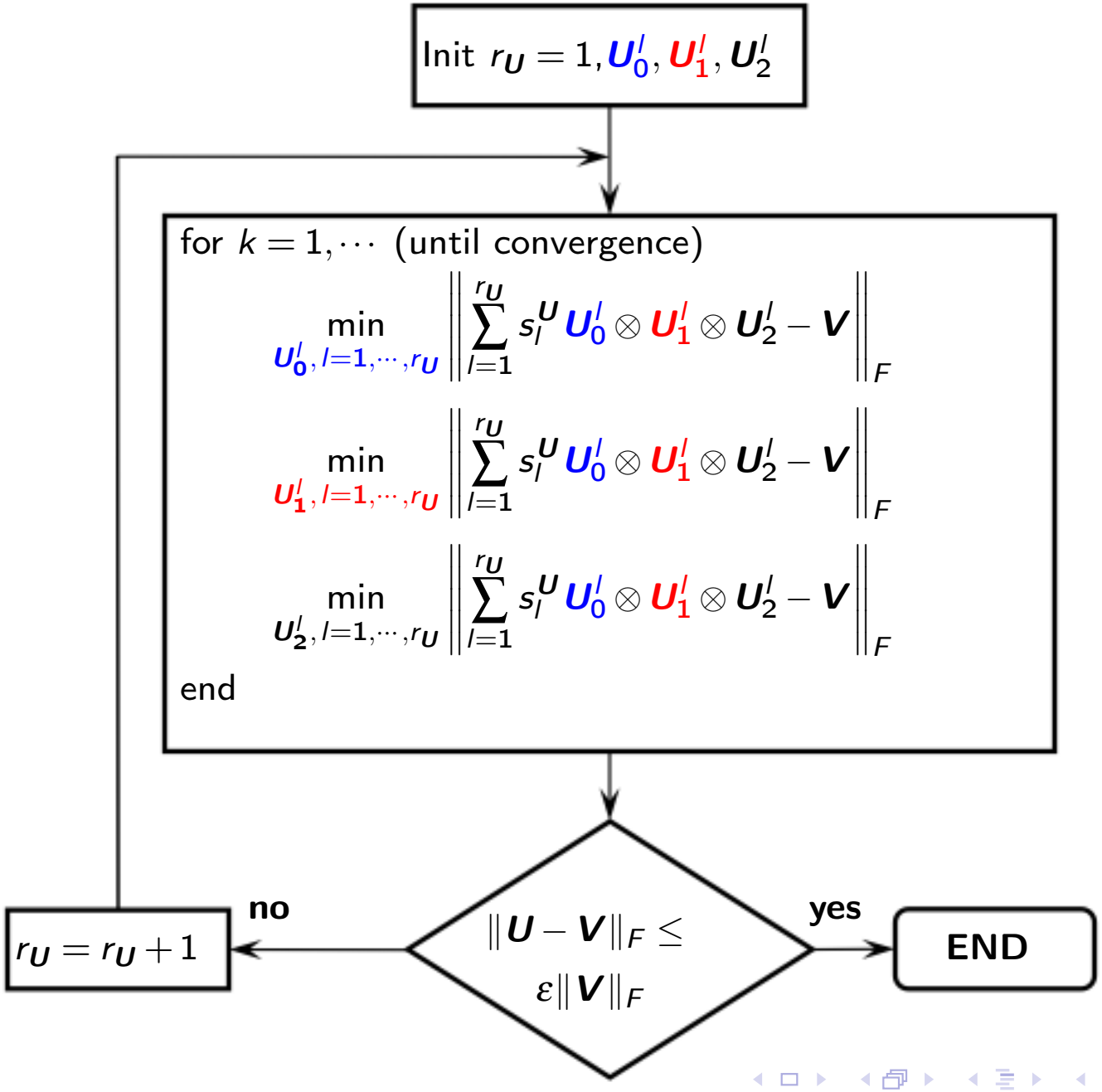
$$r_{U_{k+1}} = (r_A + 1) r_{\mathbf{U}_k} + 1.$$

To track computation *rank reduction* may be necessary:

$$\mathbf{V} = (\mathbb{I} - \mathbb{A})\mathbf{U}_k + \mathbf{B}$$
$$\mathbf{U}_{k+1} = \tau_\varepsilon(\mathbf{V}),$$

$$\mathbf{U}_{k+1} = \sum_{l=1}^{r_{\mathbf{U}_{k+1}}} s_l \bigotimes_{j=1}^{d} \mathbf{U}_j^l \quad \text{s.t.} \quad \|\mathbf{U}_{k+1} - \mathbf{V}\|_F \leq \varepsilon \|\mathbf{V}\|_F, \qquad r_{U_{k+1}} \ll r_V.$$

ALS for three directions



Solving the least squares problem for direction k

$$
\begin{aligned}
B_k \, \mathbf{c}_{j_k} &= \mathbf{b}_{j_k} \\
B_k(\hat{l}, \tilde{l}) &= \prod_{i \neq k} \left\langle \mathbf{U}_i^{\tilde{l}}, \mathbf{U}_i^{\hat{l}} \right\rangle \\
\mathbf{b}_{j_k}(\hat{l}) &= \sum_{l=1}^{r_{\mathbf{G}}} s_l^{\mathbf{G}} V_k^l(j_k) \prod_{i \neq k} \left\langle \mathbf{V}_i^l, \mathbf{U}_i^{\hat{l}} \right\rangle
\end{aligned}
$$

where $s_{\tilde{l}}^{\mathbf{U}}$ and $\mathbf{U}_k^{\tilde{l}}$ are computed from $\mathbf{c}_{j_k}$.

Remarks on ALS:

- The algorithm monotonically converges.
- Finding the least squares solution for each direction k requires forming the normal equations (because of tensor contraction).
- The normal equations, like their counterparts from matrix linear algebra, can be ill-conditioned.

Outline

1 Introduction to separated representations

2 Randomized alternating least squares

3 Optimization with canonical tensor decompositions

4 Non-intrusive algorithms

Randomized ALS

- To mitigate the potential conditioning problems found in ALS we developed a randomized variant of the algorithm.
- The idea came from linear algebra using random matrices[2]: Given $A \in \mathbb{R}^{N \times n}$, $N \geq n$, we multiply $A\mathbf{x} = \mathbf{b}$ by $R \in \mathbb{R}^{n' \times N}$ with independent random entries and solve

$$RA\mathbf{x} = R\mathbf{b}.$$

- If R is of appropriate size (i.e., n' is large enough), will be close to the least squares solution.
- RA typically has a smaller condition number than $A^T A$.

[2] See, e.g., V. Rokhlin and M. Tygert 08.

Randomized ALS continued

The randomized ALS algorithm changes B_k and $\mathbf{b}_{j_k}$ in $B_k \mathbf{c}_{j_k} = \mathbf{b}_{j_k}$,

$$
\begin{aligned}
B_k(\hat{l}, \tilde{l}) &= \prod_{i \neq k} \left\langle \mathbf{U}_i^{\tilde{l}}, \mathbf{U}_i^{\hat{l}} \right\rangle \\
\mathbf{b}_{j_k}(\hat{l}) &= \sum_{l=1}^{r_{\mathbf{G}}} s_l^{\mathbf{G}} V_k^l(j_k) \prod_{i \neq k} \left\langle \mathbf{V}_i^l, \mathbf{U}_i^{\hat{l}} \right\rangle
\end{aligned}
$$

$$\Downarrow$$

$$
\begin{aligned}
B_k(\hat{l}, \tilde{l}) &= \prod_{i \neq k} \left\langle \mathbf{U}_i^{\tilde{l}}, \mathbf{R}_i^{\hat{l}} \right\rangle \\
\mathbf{b}_{j_k}(\hat{l}) &= \sum_{l=1}^{r_{\mathbf{G}}} s_l^{\mathbf{G}} V_k^l(j_k) \prod_{i \neq k} \left\langle \mathbf{V}_i^l, \mathbf{R}_i^{\hat{l}} \right\rangle,
\end{aligned}
$$

where the entries of $\mathbf{R}_i^{\hat{l}}$ consist of independent signed Bernoulli random variables.

Main results

Theorem (Reynolds et al. 16)

For every $t \geq 0$ and with probability at least $1 - 2\exp\left(-ct^2\right)$,

$$\kappa(B_k) \leq \frac{1 + C\sqrt{r/r'} + t/\sqrt{r'}}{1 - C\sqrt{r/r'} - t/\sqrt{r'}} \kappa\left(\left(B_k^{ALS}\right)^{\frac{1}{2}}\right),$$

where C and c are constants[a].

[a] This result is a non-trivial application of methods from Vershynin 12.

- We lose monotone convergence. Solution? We only keep ALS sweeps that improve $\|\mathbf{U} - \mathbf{V}\|_F / \|\mathbf{V}\|_F$.
- There was some success using Gaussian R.V.s, but both the algorithm and the proofs work better with signed Bernoulli.
- $r' > r$, where $\hat{l} = 1, \ldots, r'$, is crucial.

Numerical example: elliptic PDE with random coefficient

$$\begin{aligned} -\nabla \cdot (a(\mathbf{x}, \mathbf{z}) \nabla u(\mathbf{x}, \mathbf{z})) &= 1, \quad \mathbf{x} \in \mathscr{D}, \\ u(\mathbf{x}, \mathbf{z}) &= 0, \quad \mathbf{x} \in \partial \mathscr{D}, \end{aligned}$$

is defined on $\mathscr{D} = (0,1) \times (0,1)$ with boundary $\partial \mathscr{D}$. $a(\mathbf{x}, \mathbf{z})$ is considered random and is modeled by

$$a(\mathbf{x}, \mathbf{z}) = a_0 + \sigma_a \sum_{k=1}^{d} \sqrt{\zeta_k} \varphi_k(\mathbf{x}) z_k,$$

where the random variables z_k are independent and uniformly distributed over the interval $[-1, 1]$, $a_0 = 0.1$, $\sigma_a = 0.01$, $d = 5$ [3]. $\{\zeta_k, \varphi_k\}_{k=1}^{d}$ are the d largest eigenpairs of the covariance function

$$C_{aa}(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_1}{l_c}\right),$$

where l_c denotes the correlation length, here set to $l_c = 2/3$.

[3] See, e.g., A. Doostan and G. Iaccarino (2009).

Discretizing the Elliptic PDE

We discretize the spatial domain $\mathscr{D}$ via triangular finite elements of size $h = 1/32$,

$$\left(K_0 + \sum_{k=1}^{d} K_k z_k \right) \mathbf{u}(\mathbf{z}) = \mathbf{f},$$

K_0 and K_k are obtained from the discretization of $\mathscr{D}$ and our representation of $a(\mathbf{x}, \mathbf{z})$. We solve for $\mathbf{u}(\mathbf{z})$ on a tensor-product grid

$$\{(z_1(j_1), \ldots, z_d(j_d)) : j_k = 1, \ldots, M_k\},$$

where $z_k(j_k)$ are Gauss-Legendre nodes with $M_k = 8$ for $k = 1 : d$. The discrete representation is now a tensor system of equations,

$$\mathbb{K}\mathbf{U} = \mathbf{F},$$

where

$$\mathbb{K}\mathbf{U} = \sum_{\hat{l}=0}^{d} \sum_{\tilde{l}=1}^{r_{\mathbf{U}}} s_{\tilde{l}}^{\mathbf{U}} \left(\mathbb{K}_0^{\hat{l}} \mathbf{U}_0^{\tilde{l}} \right) \otimes \left(\mathbb{K}_1^{\hat{l}} \mathbf{U}_1^{\tilde{l}} \right) \otimes \cdots \otimes \left(\mathbb{K}_d^{\hat{l}} \mathbf{U}_d^{\tilde{l}} \right).$$

Solving the tensor system

- We solve this system for $\mathbf{U}$ via the fixed point iteration,

$$\mathbf{U}_{i+1} = (\mathbb{I} - \mathbb{K})\mathbf{U}_i + \mathbf{F}$$

  where τ_ε is the rank reduction operator.
- One fixed point iteration will increase the rank of $\mathbf{U}_{i+1}$ to $r_{\mathbb{K}} r_{\mathbf{U}_i} + r_{\mathbf{U}_i} + 1$.
- To keep the rank reasonable we use a rank reduction operator (ALS or randomized ALS), τ_ε, where ε is a user-supplied truncation accuracy, at each iteration,

$$\begin{aligned}\mathbf{V} &= (\mathbb{I} - \mathbb{K})\mathbf{U}_i + \mathbf{F} \\ \mathbf{U}_{i+1} &= \tau_\varepsilon(\mathbf{V}),\end{aligned}$$

- In this example there are significant benefits to using randomized ALS instead of ALS.

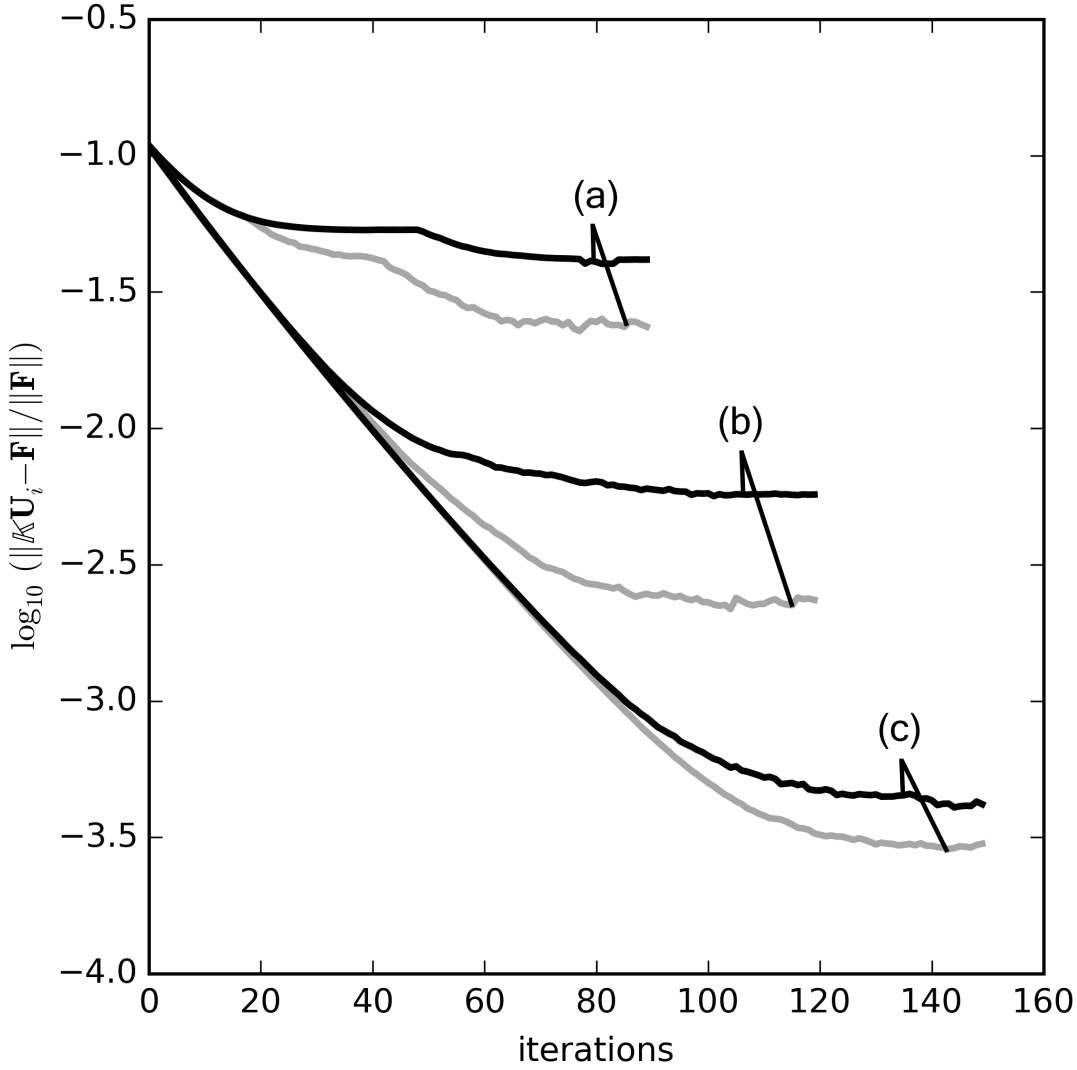


Figure 1: Residual error vs. iteration of results from linear solvers. Black lines: standard ALS. Gray lines: randomized ALS. ALS tolerances: (a) 1×10^{-3}, (b) 1×10^{-4}, (c) 1×10^{-5}.

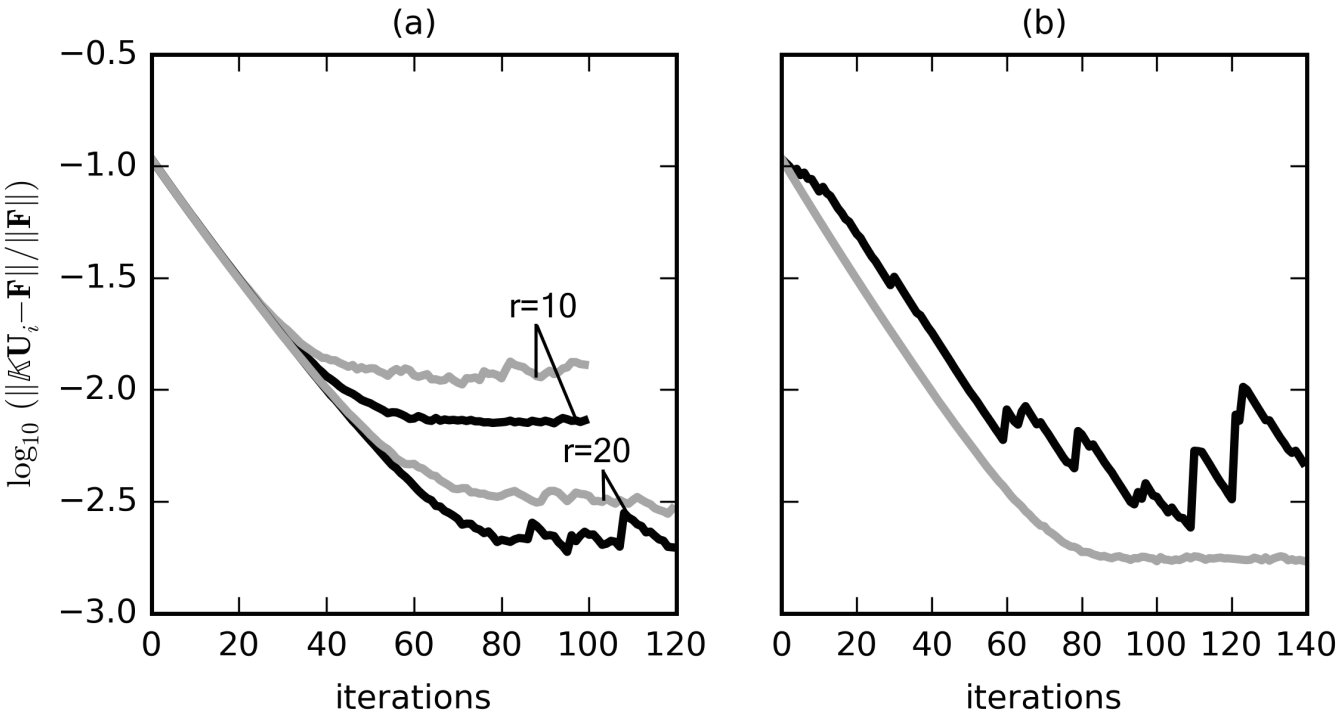


Figure 2: Relative residuals of linear solves vs. fixed point iteration number. (a) ranks $r = 10$, and $r = 20$. Gray lines correspond to randomized ALS, black to standard ALS. (b) Rank $r = 30$.

ALS type	ALS tol	max $\kappa(B_k)$	rank	residual
standard	1×10^{-5}	9.45×10^{11}	10	7.29×10^{-3}
	5×10^{-6}	1.27×10^{13}	20	1.97×10^{-3}
	1×10^{-6}	3.00×10^{13}	30	4.73×10^{-3}
randomized	1×10^{-5}	9.39×10^{5}	10	1.30×10^{-2}
	5×10^{-6}	4.12×10^{6}	20	2.93×10^{-3}
	1×10^{-6}	3.94×10^{7}	30	1.72×10^{-3}

Table 1: Maximum condition numbers and final relative residual errors of experiments with fixed separation ranks.

Outline

1 Introduction to separated representations

2 Randomized alternating least squares

3 Optimization with canonical tensor decompositions

4 Non-intrusive algorithms

Optimization given a CTD

Given a tensor $\mathbf{U}$ represented in the CTD format there are currently two options to find the entry with maximum absolute value.

- A linearly convergent algorithm based on the power method, see Espig et al. (2013).
- A new, quadratically convergent, algorithm based on sequential squaring, see Reynolds et al. (paper submitted to JCP, and on arXiv).

Both methods take advantage of the Hadamard product, which is easily computed in the canonical format.

Power method, Espig et al. (2013)

Algorithm 1

To find the entries with maximum absolute value of a tensor $\mathbf{U} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_d}$ in CTD format, given a tolerance $\varepsilon > 0$ for the reduction algorithm τ_ε, we define

$\mathbf{Y}_0 = \left(\prod_{i=1}^{d} \frac{1}{N_i} \right) (\mathbf{1}_1 \circ \mathbf{1}_2 \circ \cdots \circ \mathbf{1}_d)$, where $\mathbf{1}_i = (1, 1, \ldots, 1)^T \in \mathbb{R}^{N_i}$, and iterate (up to $k_{\max}$ times) as follows:

for $k = 1, 2, \ldots, k_{\max}$ **do**

1. $\mathbf{Q}_k = \mathbf{U} * \mathbf{Y}_{k-1}$, $\lambda_k = \langle \mathbf{Y}_{k-1}, \mathbf{Q}_k \rangle$, $\mathbf{Z}_k = \mathbf{Q}_k / \|\mathbf{Q}_k\|_F$
2. $\mathbf{Y}_k = \tau_\varepsilon(\mathbf{Z}_k)$

end for

Squaring method, Reynolds et al. (submitted)

Algorithm 2

To find the entries with maximum absolute value of a tensor $\mathbf{U} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_d}$ in CTD format, given a reduction tolerance $\varepsilon > 0$ for the reduction algorithm τ_ε, we define

$\mathbf{Y}_0 = \mathbf{U} / \|\mathbf{U}\|_F$, and iterate (up to $k_{\max}$ times) as follows:

for $k = 1, 2, \ldots, k_{\max}$ **do**

1. $\mathbf{Q}_k = \mathbf{Y}_{k-1} * \mathbf{Y}_{k-1}$
2. $\mathbf{Y}_k = \tau_\varepsilon(\mathbf{Q}_k)$, $\mathbf{Y}_k = \mathbf{Y}_k / \|\mathbf{Y}_k\|_F$

end for

Optimization algorithm test

Construct a CTD representation of a tensor $\mathbf{U}$ with the following properties:

- $\mathbf{U}$ has rank 4, a random rank 3 tensor plus a rank 1 tensor with a magnitude 1 spike in a random location.
- Random rank 3 tensor is constructed from factors with random entries drawn from a uniform distribution on $[0.9, 1]$.
- Dimensions of $\mathbf{U}$: $d = 8$, $M = 32$ entries in each direction.
- CTD reduction algorithm accuracy $\varepsilon = 10^{-6}$.
- Computation times are from MATLAB code running on individual cores of a Intel Xeon 2.40 GHz processor.

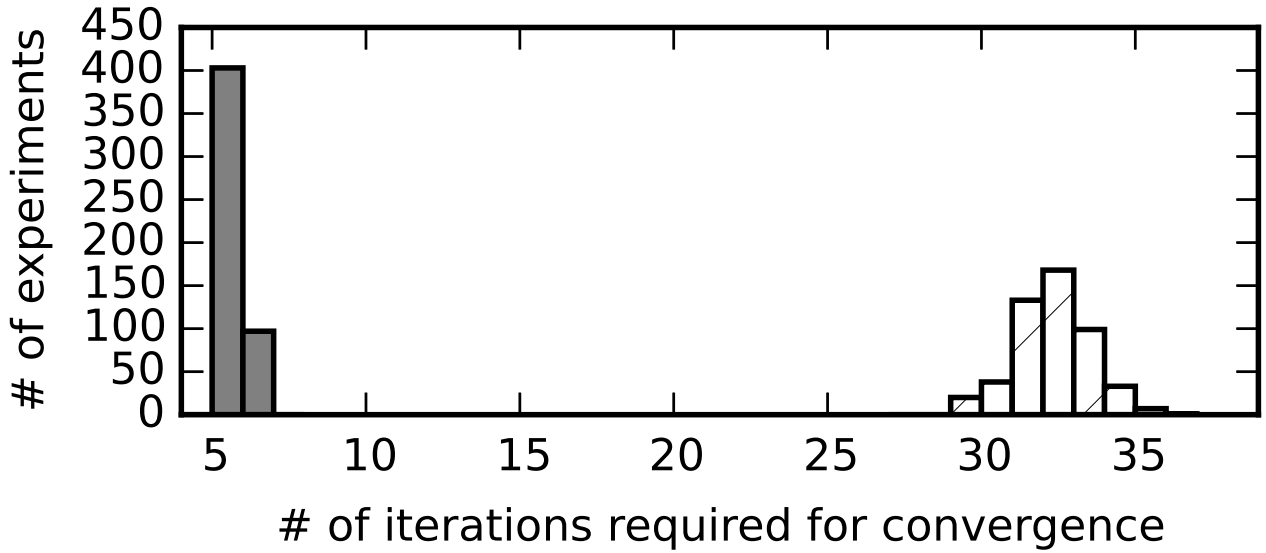


Figure 3: Number of iterations required for convergence to a rank one CTD by the power method optimization (hatched pattern) and the squaring method optimization (solid gray).

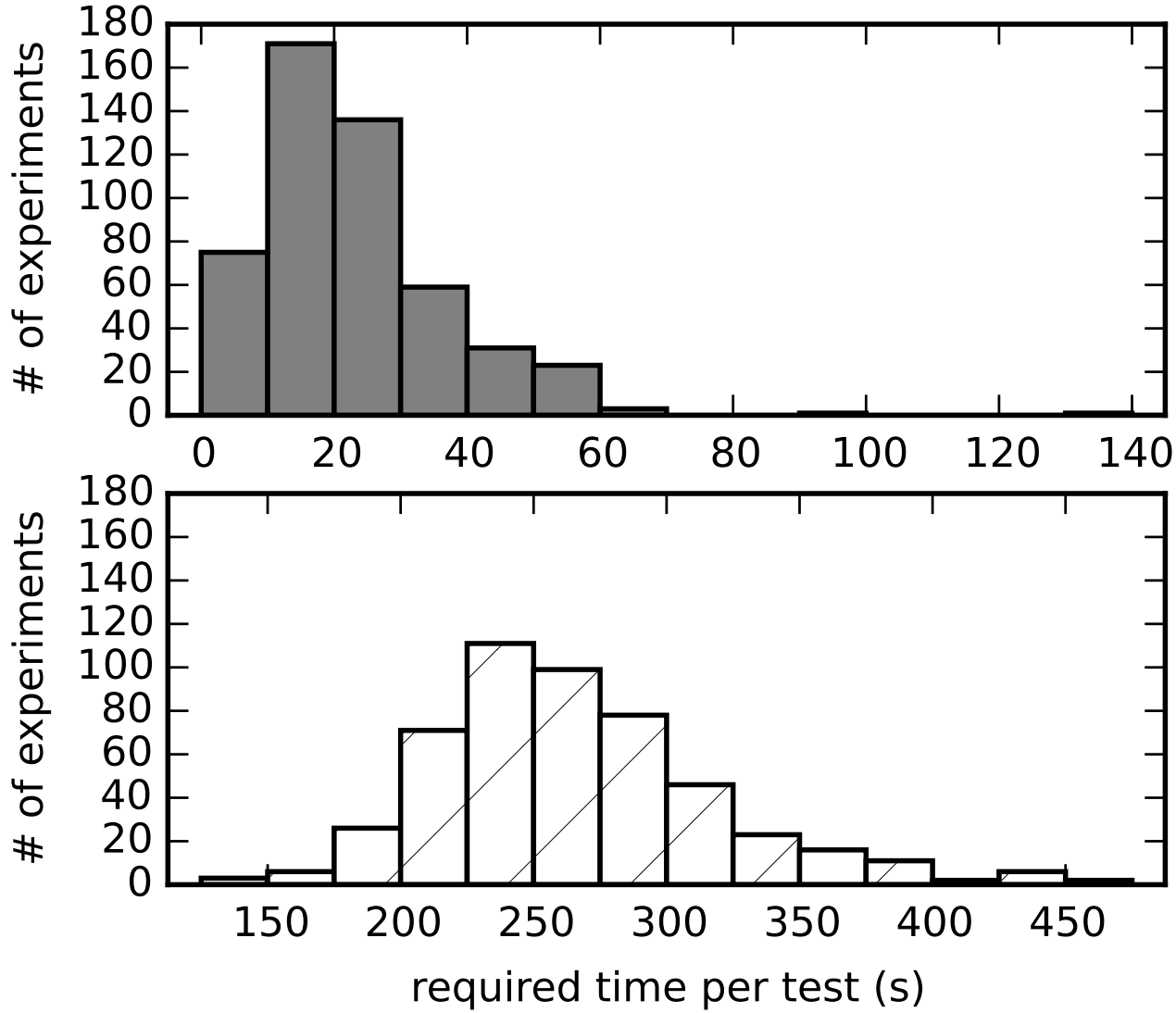


Figure 4: Computation times, in seconds, required for convergence to a rank one CTD by the power method optimization (hatched pattern) and the squaring method optimization (solid gray).

Optimization example 2: worst-case scenario response

In Reynolds et al. (submitted) we computed the solution to

$$\begin{aligned} -\nabla \cdot (a(\mathbf{x},\mathbf{z})\nabla u(\mathbf{x},\mathbf{z})) &= 1, \quad \mathbf{x} \in \mathscr{D}, \\ u(\mathbf{x},\mathbf{z}) &= 0, \quad \mathbf{x} \in \partial\mathscr{D}, \end{aligned}$$

defined on $\mathscr{D} = (0,1)\times(0,1)$ with boundary $\partial\mathscr{D}$.

- $a(\mathbf{x},\mathbf{z}) = a_0 + \sigma_a \sum_{k=1}^{d} \sqrt{\zeta_k}\varphi_k(\mathbf{x})z_k$, $a_0 = 0.1$, $\sigma_a = 0.01$, $d = 20$.
- We used 600 samples, via a regression approach[4], to compute the SR of $u(\mathbf{x},\mathbf{z})$ at $\mathbf{x}^* = (0.4839, 0.4839)$.
- Each sample used independent realizations of uniform r.v. $\{z_k\}_{k=1}^{d}$.
- The SR of $u(\mathbf{x}^*,\mathbf{z})$ was rank 5 and used Legendre polynomials of at most order 3.
- Relative RMS validation error, computed from 3400 samples, was $3.256 * 10^{-4}$.

[4] Beylkin et al. 09, Doostan et al. 13

Worst-case scenario response: results

We compared the output of the CTD squaring algorithm with output from the genetic algorithm (GA) from the MATLAB Optimization tool box. For these tests:

- We used 32 equispaced samples of $[-1,1]$ to discretize z_k in $u(\mathbf{x}^*,\mathbf{z})$ for $k=1,\ldots,d$.
- CTD rank reduction tolerance was set to $\varepsilon = 10^{-3}$ to keep the separation rank reasonably low.

Opt. Alg.	max	mean	std. dev.	av. PDE solns
GA	8.744×10^{-1}	8.731×10^{-1}	5.7×10^{-4}	4.419×10^{4}
CTD	8.750×10^{-1}	8.747×10^{-1}	3.6×10^{-4}	600

Table 2: Statistics from 500 tests of the CTD squaring algorithm and GA.

Outline

1 Introduction to separated representations

2 Randomized alternating least squares

3 Optimization with canonical tensor decompositions

4 Non-intrusive algorithms

Non-intrusive algorithm

Question:

Suppose we don't have access to solver code, how do we construct a separated representation of the solution to, e.g., our PDE in high dimension?

Answer:

We use a regression-type approach to construct the SR from sample generated by the PDE solver. See, e.g., Beylkin et al. (2009) or Doostan et al. (2013).

SR approximations from scattered data

To compute a separated representation

$$g(\mathbf{x}) = \sum_{l=1}^{r} s_l \prod_{i=1}^{d} g_i(x_i),$$

from scattered data $\{y_j\}_{j=1}^{N}$, a sequence of one dimensional problems are solved. The one-dimensional subproblems are formed by computing,

$$p_j^l = s_l \prod_{i=2}^{d} g_i^l\left(x_i^j\right),$$

for the sample indices j and rank terms l and subsequently minimizing the residual

$$\sum_{j=1}^{N} \left(y_j - \sum_{l=1}^{r} p_j^l g_1^l\left(x_1^j\right) \right)^2.$$

Functional approximations in separated form continued

The function $g_1^l(x_1)$ is of the following form

$$g_1^l(x_1) = \sum_{m=0}^{M} c_m^l \phi_m(x_1).$$

Thus, the minimization problem is

$$\underset{\mathbf{c}}{\operatorname{argmin}} \sum_{j=1}^{N} \left(y_j - \sum_{l=1}^{r} \sum_{m=0}^{M} c_m^l p_j^l \phi_m\left(x_1^j\right) \right)^2.$$

To solve the minimization problem form,

$$V(j,i) = \phi_i(x_j), \;\; i = 0,\ldots,M \text{ and } j = 1,\ldots,N,$$

then the block matrix,

$$A = [P_1 V, P_2 V, \ldots, P_r V],$$

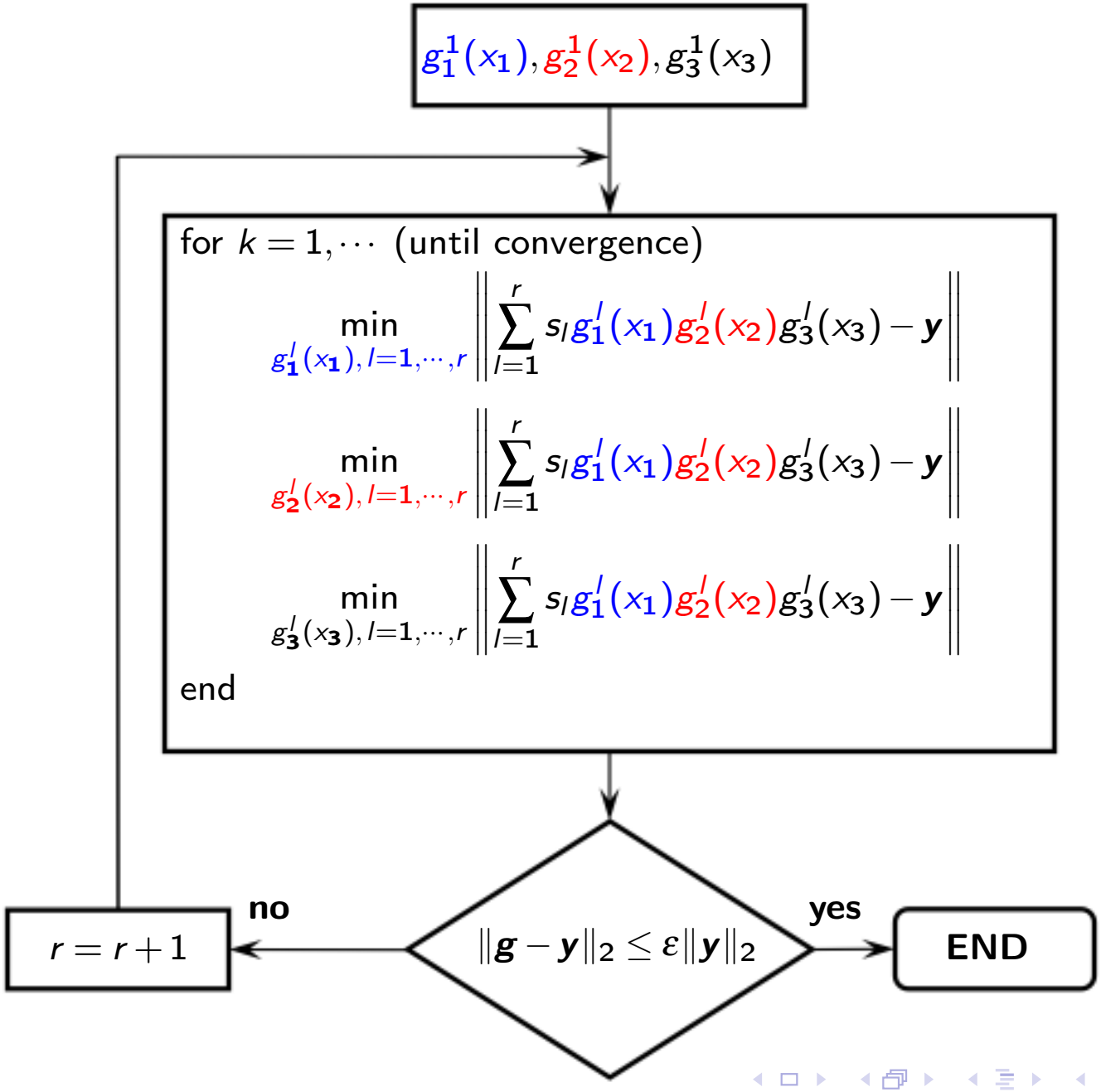
where

$$P_l = \operatorname{diag}\left(p_1^l, \ldots, p_{N-1}^l\right),$$

and solve,

$$A\mathbf{c} = \mathbf{y}.$$

Functional approximations in separated form continued



Sampling approaches

Current research topic: how should we draw samples to construct separated representations?

- Which distribution should we use?
- How should we weight?
- Sampling strategy, e.g. D-optimal sampling.
- How does the sampling strategy affect the conditioning of the linear system?

Picking a sampling distribution

Sampling according to random variables defined by an orthogonality measure.
Standard sampling for polynomial chaos (PC) regression:

- d-dimensional Legendre polynomials: uniform distribution on $[-1,1]^d$
- d-dimensional Hermite polynomials: multi-variate normal (each coordinate i.i.d)

Asymptotic strategy for PC regression[5]

- d-dimensional Legendre polynomials: Chebyshev sampling
- d-dimensional Hermite polynomials: uniform sampling on the d-dimensional ball of radius $2\sqrt{p}$ (p is the total order of the polynomial)

[5] Hampton, Doostan 15

Hermite function example

We have been looking at the following example[6]

$$u(y_1, y_2) = s_0 + s_1 \psi_3(y_1)\psi_3(y_2) + s_2 \psi_2(y_1) + s_3 \psi_2(y_2) + s_4 \psi_3(y_1) + \varepsilon,$$

where $\{y_k\}_{k=1}^2$ are i.i.d random variables and $\psi_i(\cdot)$ are Hermite polynomials of degree i.

- Noise ε is a zero mean Gaussian r.v. with std. dev. 0.0005.
- $\{s_i\}_{i=1}^4 = \{0.55, \sqrt{2}/2, -\sqrt{2}/4, -\sqrt{2}/4, -1/10\}$.
- $\mathbb{E}[u] = 0.55$, and $\mathrm{Var}[u] = 0.76$ for noise-free u.

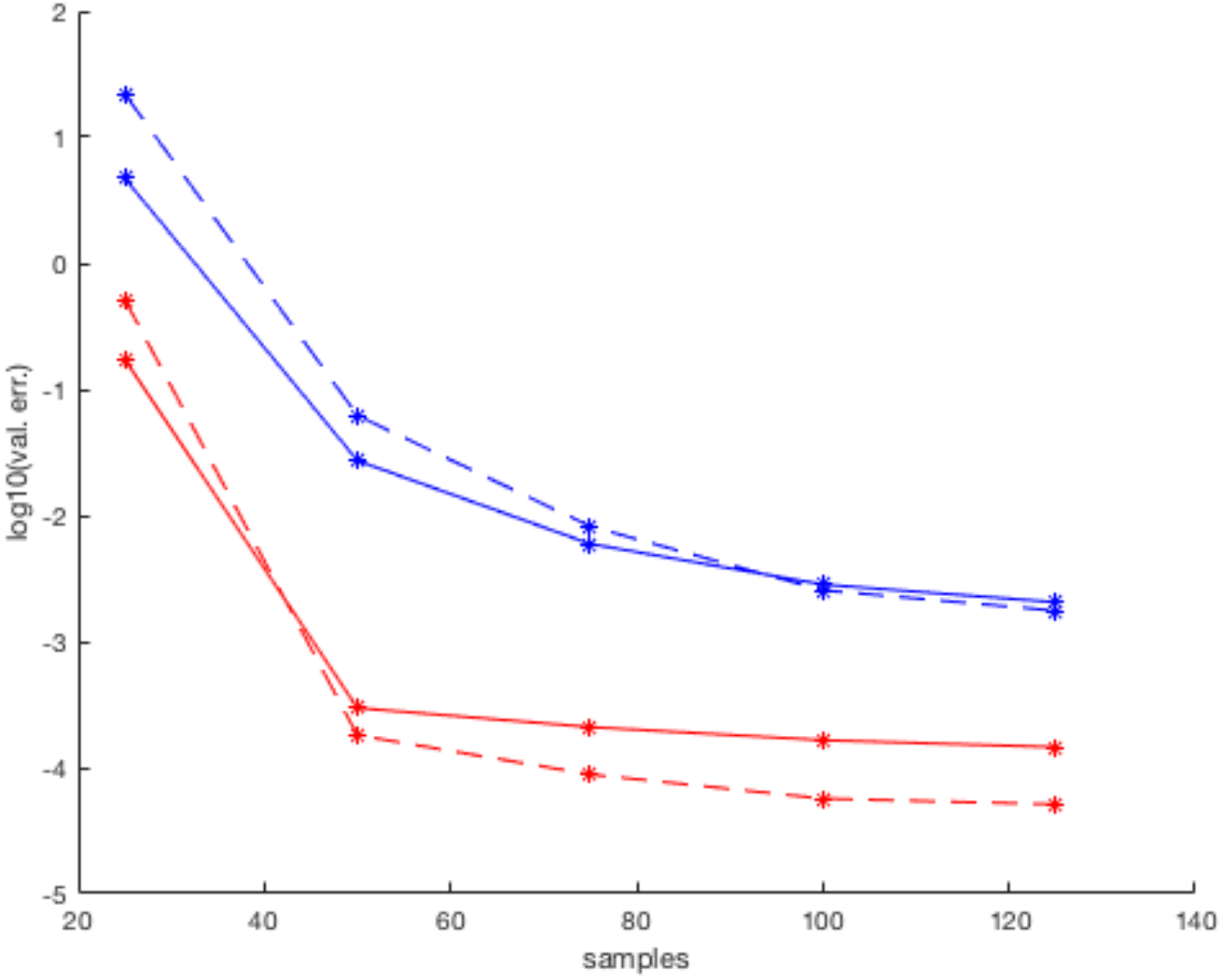
[6] Based of an example in Doostan et al. 13.

Hermite function example cont'd

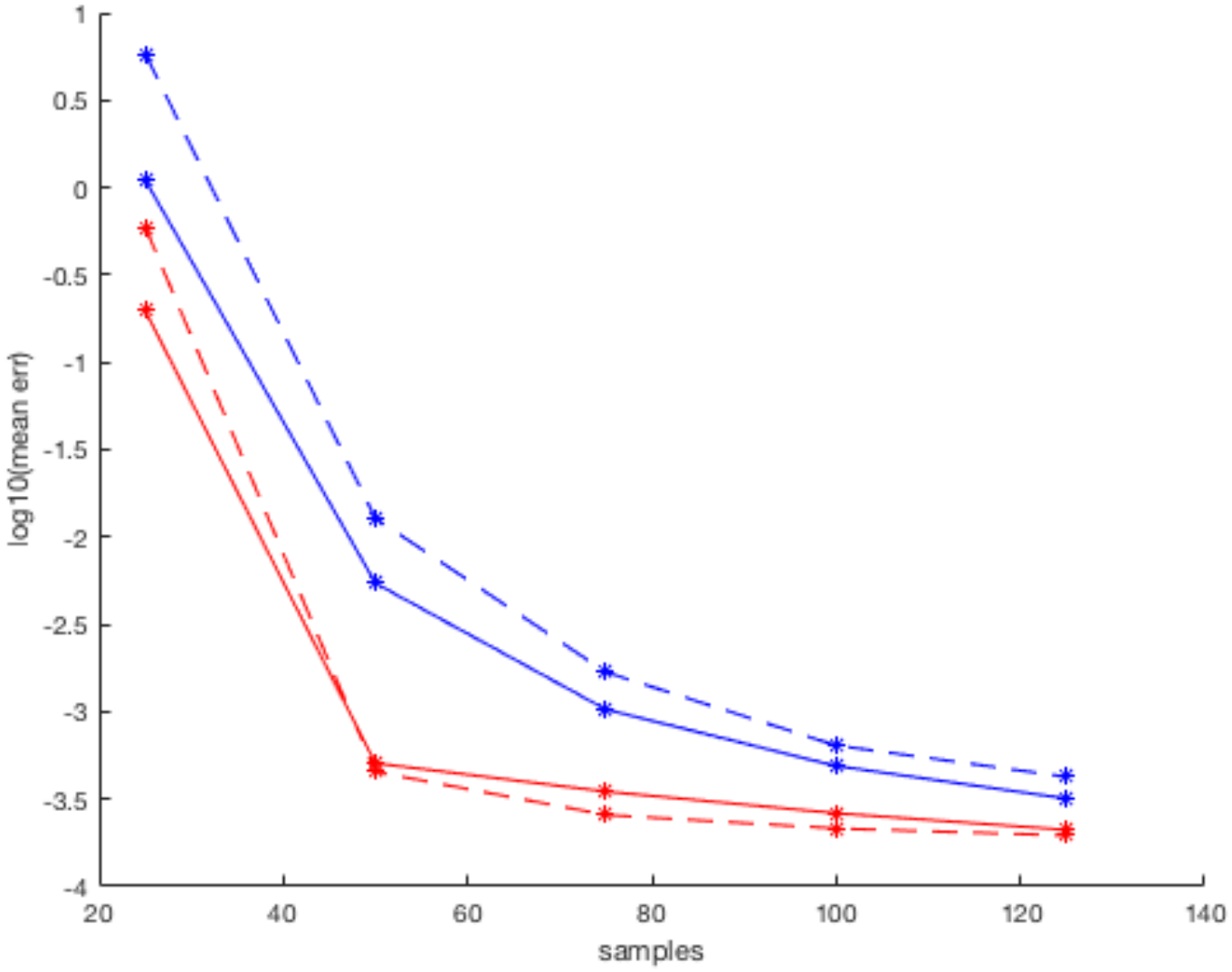
To study this example we used the following experimental setup

- Compute a set of 10000 samples with $\{y_1, y_2\}$ chosen either from the standard or asymptotic sampling strategy for Hermite polynomials.
- The last 1000 samples will be used for model validation in all tests.
- Take randomly selected subsets (without replacement) from the first 9000 experiments.
- For all experiments, we compute the validation error, errors of the computed mean and variance, and the condition number.
- Look at the mean and standard deviations of these errors and condition numbers. Statistics computed from 100 tests.
- The separated representations constructed in these tests have rank 4 and use Hermite polynomials of order at most 3.

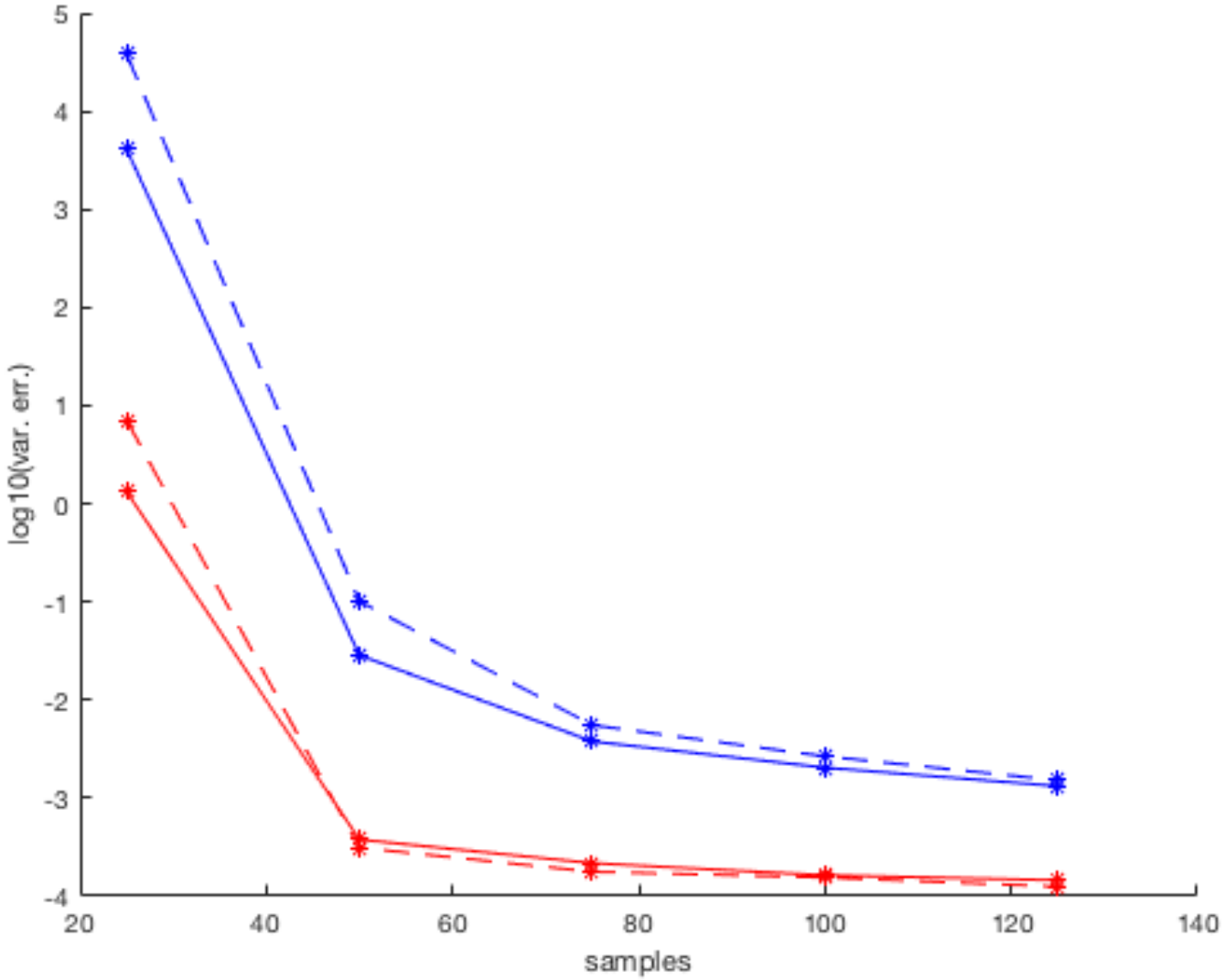
Validation error



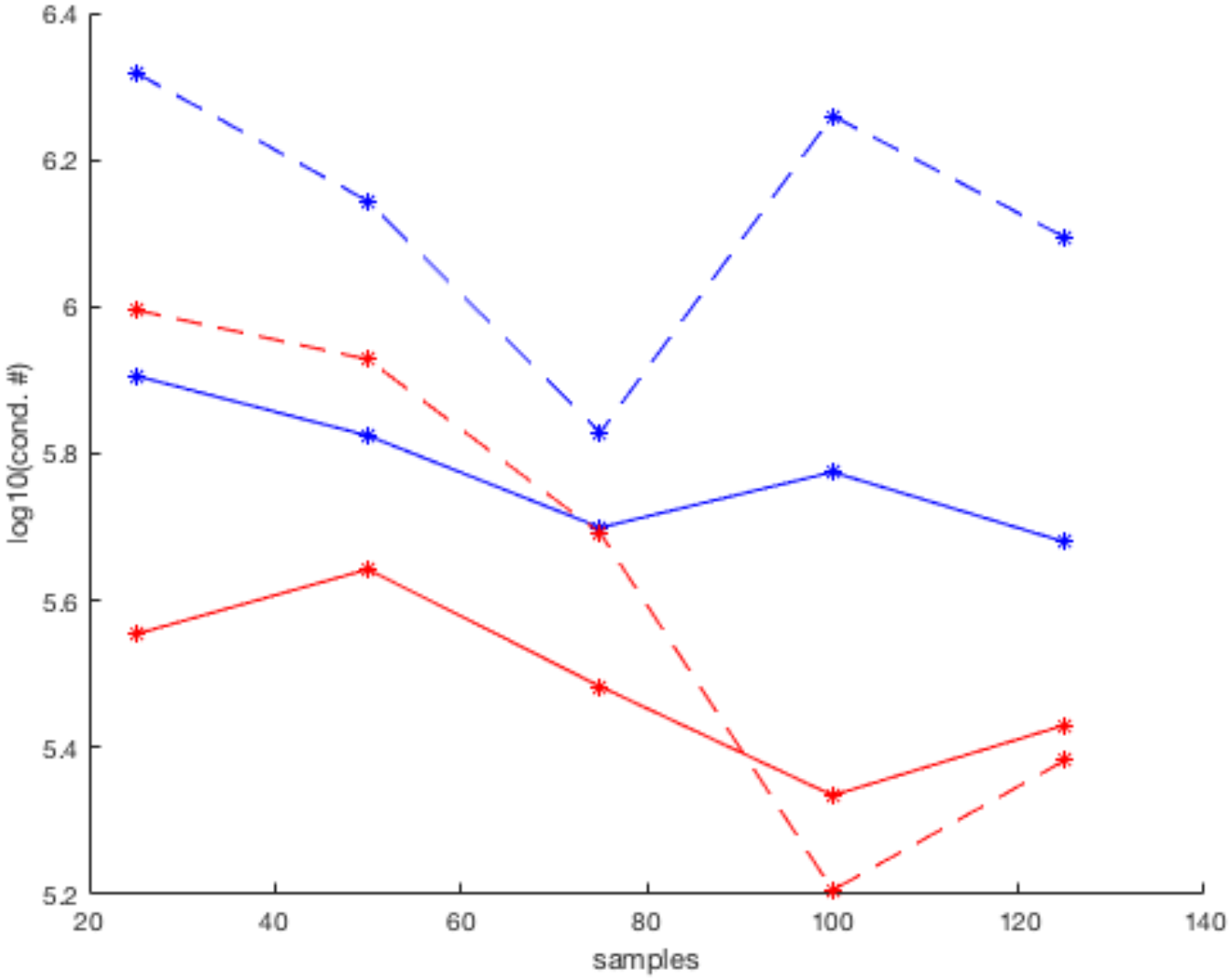
Mean error


1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
-3.5
-4
log10(mean err)
20
40
60
80
100
120
140
samples

Variance error



Condition number



Thank you!