Review of Spectral Tensor-Train Decomposition

Matthew Reynolds[1]

[1]Department of Aerospace Engineering Sciences, CU Boulder

UQ Group meeting
Oct 24th, 2016

Outline

Outline

Separated representations

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

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

Bad idea! The number of coefficients $c_{i_1 \dots i_d}$ 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,

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

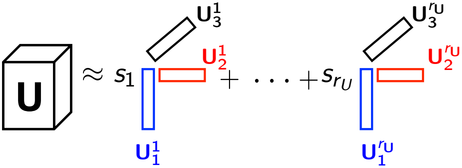
- The number of terms, r, is called the separation rank is assumed to be *small*.
- Given a separated representation,

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

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. $\left\| \mathbf{U}_j^l \right\|_2 = 1$.

Applications of CTDs and separated representations

- psychometrics
- chemometrics
- multivariate regression and machine learning
- uncertainty quantification
- ... and many more[1]

[1] Kolda and Bader (2009)

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	$\Vert \mathbf{U} \Vert_F^2 = \langle \mathbf{U}, \mathbf{U} \rangle$	$\mathcal{O}(d \cdot r_{\mathbf{U}}^2 \cdot N)$
"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 SRs 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 $r_{\mathbb{A}} r_{\mathbf{U}}$.

Rank reduction

Since many operations in separated form increase the rank, we require a rank reduction algorithm. Specifically, given $\mathbf{G}$,

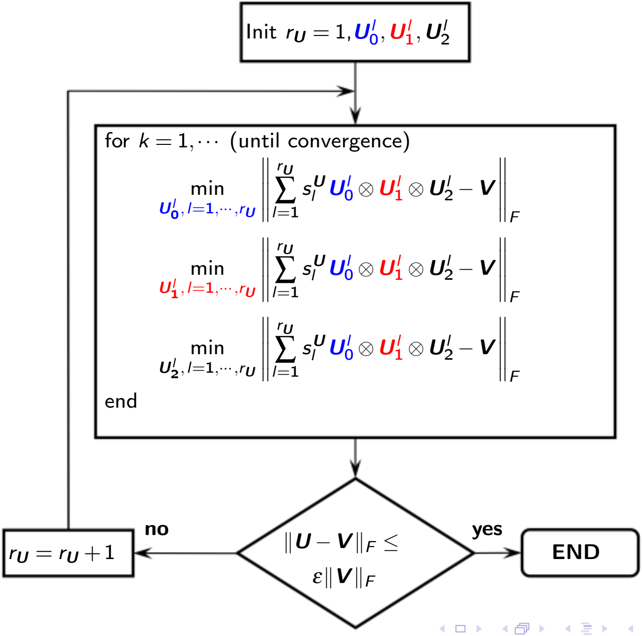
$$\mathbf{G} = \sum_{l=1}^{r_{\mathbf{G}}} s_l^{\mathbf{G}} \mathbf{G}_1^l \circ \cdots \circ \mathbf{G}_d^l,$$

and an acceptable error ε, we attempt to find a representation

$$\mathbf{F} = \sum_{\tilde{l}=1}^{r_{\mathbf{F}}} s_{\tilde{l}}^{\mathbf{F}} \mathbf{F}_1^{\tilde{l}} \circ \cdots \circ \mathbf{F}_d^{\tilde{l}}$$

with $r_{\mathbf{F}} < r_{\mathbf{G}}$, such that $\|\mathbf{F} - \mathbf{G}\| / \|\mathbf{G}\| < \varepsilon$.

ALS for three directions



Outline

CTD limitations and the Tucker decomposition

- Unfortunately, unlike the SVD, the best rank r CTD approximation to a tensor does not always exist (space of rank r tensors isn't closed).
- Computations of CTDs based on ALS algorithms don't always find a global minimum in terms of approximation error.
- A different tensor decomposition, the Tucker decomposition,

$$\mathbf{U} = \sum_{l_1=1}^{r_1} \cdots \sum_{l_d=1}^{r_d} s_{l_1,\ldots,l_d} \mathbf{U}_1^{l_1} \otimes \cdots \otimes \mathbf{U}_d^{l_d}$$

  does not suffer from the "closure problem."
- Curse of Dimensionality again: number of $s_{l_1,\ldots l_d}$ grows exponentially with d.

Hierarchical Tucker and Tensor-Train

- A variant of the Tucker decomposition, the Heirarchical Tucker decomposition, addresses the Curse of Dimensionality problem.
- Tensor Train (TT) is a particular type of Heirarchical Tucker decomposition, and is defined as follows:

Definition

Let $\mathbf{U} \in \mathbb{R}^{N_1,\ldots,N_d}$ have entries $\mathbf{U}(i_i,\ldots,i_d)$. The tensor-train rank $\mathbf{r} = (r_0,\ldots,r_d)$ approximation of $\mathbf{U}$, $\mathbf{U}_{TT} \in \mathbb{R}^{N_1,\ldots,N_d}$, is defined as

$$
\begin{aligned}
\mathbf{U}(i_1,\ldots,i_d) &= \mathbf{U}_{TT}(i_i,\ldots,i_d) + \varepsilon_{TT}(i_i,\ldots,i_d) \\
&= \sum_{\alpha_0,\ldots,\alpha_d}^{\mathbf{r}} G_1(\alpha_0, i_1, \alpha_1)\ldots G_d(\alpha_{d-1}, i_d, \alpha_d) \\
&\qquad + \varepsilon_{TT}(i_i,\ldots,i_d),
\end{aligned}
$$

where ε_{TT} is the residual term and $r_0 = r_d = 1$.

Cost comparison: Tensor-Train vs. CTD

Comparison of operations, assuming all directional samples are all $\mathcal{O}(N)$ and all ranks r_k are $\mathcal{O}(r)$.

Operation	**TT cost**	**CTD cost**
Storage	$\mathcal{O}(d \cdot r^2 \cdot N)$	$\mathcal{O}(d \cdot r \cdot N)$
Inner product	$\mathcal{O}(d \cdot r^4 \cdot N)$	$\mathcal{O}(d \cdot r^2 \cdot N)$
Frobenius norm	$\mathcal{O}(d \cdot r^4 \cdot N)$	$\mathcal{O}(d \cdot r^2 \cdot N)$
"Matrix-vector" multiply	$\mathcal{O}(d \cdot r^4 \cdot N^2)$	$\mathcal{O}(d \cdot r^2 \cdot N^2)$

Note: if $r_k = \mathcal{O}(r)\ \forall k$, then the inner product and Frobenius norm can be computed from the TT format in $\mathcal{O}\left(d \cdot r^3 \cdot N\right)$.

Properties of Tensor-Train

- There exists an exact TT representation ($\varepsilon_{TT} = 0$) for which

$$r_k = rank\left(\mathbf{U}_k\right),\ \forall k \in \{1, \ldots, d\},$$

where $\mathbf{U}_k$ is the "k-th unfolding of $\mathbf{U}$," i.e.

$$\mathbf{U}_k = reshape\left(\mathbf{U}, \prod_{s=1}^{k} N_s, \prod_{s=k+1}^{d} N_s\right).$$

- If $r_k \leq rank\left(\mathbf{U}_k\right)$, a best TT rank $\mathbf{r}$ approximation $\mathbf{U}^{best}$ always exists.
- The TT-SVD algorithm [Oseledets, 2011] outputs an approximation to $\mathbf{U}^{best}$ s.t.

$$\left\|\mathbf{U}_{TT} - \mathbf{U}\right\|_F \leq \sqrt{d-1}\left\|\mathbf{U}^{best} - \mathbf{U}\right\|_F.$$

Properties of Tensor-Train cont'd

- If the truncation tolerance for the SVD of each unfolding is set to $\delta = \varepsilon / \sqrt{d-1} \|\mathbf{U}\|_F$,

$$\|\mathbf{U}_{TT} - \mathbf{U}\|_F \leq \varepsilon \|\mathbf{U}\|_F .$$

- Complexity of the TT-SVD, assuming $r_k = r$ and $N_k = N$: $\mathscr{O}\left(rN^d\right)$.
- Curse of Dimensionality, yet again!
- A method for constructing TT approximations with only a small number of "function evaluations" is required.

TT-DMRG-cross

The TT-DMRG-cross approach solves,

$$\min_{G_1 \ldots G_d} \left\| \mathbf{U} - \mathbf{U}_{TT} \right\|,$$

by optimizing over two cores, G_k and G_{k+1}, at the same time. The important cores

$$W_k\left(i_k, i_{k+1}\right) = G_k\left(i_k\right) G_{k+1}\left(i_{k+1}\right),$$

are found using the maximum volume principle and G_k, G_{k+1} are recovered via the SVD.

Remarks

This is accomplished by selecting the most important *planes* $\mathbf{U}\left(i_1, \ldots, i_{k-1}, :, :, i_{k+2}, \ldots, i_d\right)$ in the d-dimensional space. This process is rank revealing.

Outline

Functional approximations in separated form

A method for generating separated representations from scattered data was introduced by [Beylkin et. al 2009]. To form the separated representation,

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

a sequence of one dimensional problems are solved. To form the one-dimensional subproblems compute,

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

for the sample indices j and rank terms l and minimize the residual

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

where y_j is the j-th data value.

Functional approximations in separated form continued

To solve the minimization problem (assuming g's are polynomials of fixed order) first form the standard Vandermonde matrix,

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

then form the following block matrix,

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

where

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

and solve the linear system,

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

for the coefficients of $g_1^l(x_1)$, $l = 1, \ldots r$.

Outline

Sketch of the functional approximation algorithms using TT

The idea behind using tensor-train to generate functional representations is to,

1. Construct a tensor product grid from Gaussian quadratures.
2. Evaluate the weighted target function f at the quadrature points; this forms a tensor $\mathbf{B} = f(\mathbf{X})\sqrt{\mathbf{W}}$.
3. Approximate the tensor $\mathbf{B}$ with a tensor train approximation,

$$\|\mathbf{B} - \mathbf{B}_{TT}\|_F \leq \varepsilon \|\mathbf{B}\|_F,$$

   for this step use either TT-SVD, TT-cross, or TT-DMRG-cross.
4. The TT cores of $\mathbf{B}_{TT}$ are used as a discretized version of the cores of the functional tensor-train, automatically leading to the following approximation of f,

$$\|f - f_{TT}\|_{L^2_\mu} \lesssim \varepsilon \|f\|_{L^2_\mu}.$$

Sketch of the functional approximation algorithms using TT

From the functional tensor-train approximation of f, f_{TT}, there are two options:

- Project f_{TT} onto a polynomial basis.
- Use f_{TT} to generate a Lagrange interpolation from candidate points.

To get to these options let us first understand the connection between $\mathbf{B}_{TT}$ and f_{TT}. f_{TT} is defined as,

$$f_{TT}(\mathbf{x}) := \sum_{\alpha_0,\ldots,\alpha_d}^{\mathbf{r}} \gamma_1(\alpha_0, x_1, \alpha_1)\ldots\gamma_d(\alpha_{d-1}, x_d, \alpha_d).$$

Relationship between the DTT and FTT

$$f_{TT}(\mathbf{x}) := \sum_{\alpha_0,\ldots,\alpha_d}^{\mathbf{r}} \gamma_1(\alpha_0, x_1, \alpha_1) \ldots \gamma_d(\alpha_{d-1}, x_d, \alpha_d).$$

And recalling that the tensor $\mathbf{B}$, approximated by $\mathbf{B}_{TT}$, was generated from weighted samples of f,

$$\mathbf{A}_{TT} = \mathbf{B}_{TT}/\sqrt{\mathbf{W}} = \sum_{\alpha_0,\ldots,\alpha_d}^{\mathbf{r}} G_1(\alpha_0, i_1, \alpha_1) \ldots G_d(\alpha_{d-1}, i_d, \alpha_d)$$

is f_{TT} evaluated on the tensor product grid of quadrature nodes. From these samples it is straightforward to,

- compute projections onto polynomial bases.
- use linear or Lagrange interpolation.

Projection onto polynomial bases

To compute the projection $P_{\mathbf{N}} f_{TT} = \sum_{\mathbf{i}=0}^{\mathbf{N}} \tilde{c}_{\mathbf{i}} \Phi_{\mathbf{i}}$ evaluate,

$$\tilde{c}_{\mathbf{i}} = \int_{\mathbf{I}} f_{TT}(\mathbf{x}) \Phi_{\mathbf{i}}(\mathbf{x}) \, d\mu(\mathbf{x}) = \sum_{\alpha_0,\ldots,\alpha_d=1}^{\mathbf{r}} \beta_1(\alpha_0, i_1, \alpha_1) \ldots \beta_d(\alpha_{d-1}, i_d, \alpha_d),$$

where

$$\beta_n(\alpha_{n-1}, i_n, \alpha_n) = \int_{I_n} \gamma(\alpha_{n-1}, x_n, \alpha_n) \phi_{i_n}(x_n) \, d\mu_n(x_n).$$

Discretizing this integral via Gaussian quadrature yields,

$$\beta_n(\alpha_{n-1}, i_n, \alpha_n) \approx \hat{\beta}_n(\alpha_{n-1}, i_n, \alpha_n) = \sum_{j=0}^{N_n} \gamma_n\left(\alpha_{n-1}, x_n^{(j)}, \alpha_n\right) w_n^{(j)},$$

where $\left(x_n^{(j)}, w_n^{(j)}\right)$, $j = 0, \ldots, N_n$, are Gaussian quadrature nodes and weights.

Procedure 1 `FTT-projection-construction`

Input: Function $f : \mathbf{I} \to \mathbb{R}$; measure $\mu = \prod_{n=1}^{d} \mu_n$; integers $\mathbf{N} = \{N_n\}_{n=1}^{d}$ denoting the polynomial degrees of approximation; univariate basis functions $\left\{ \{\phi_{i_n,n}\}_{i_n=0}^{N_n} \right\}_{n=1}^{d}$ orthogonal with respect to μ_n; `DMRG-cross` approximation tolerance ε.

Output: $\mathcal{C}_{TT}(i_1, \ldots, i_d) = \sum_{\alpha_0,\ldots,\alpha_d=1}^{\mathbf{r}} \hat{\beta}_1(\alpha_0, i_1, \alpha_1) \cdots \hat{\beta}_d(\alpha_{d-1}, i_d, \alpha_d)$, the TT decomposition of the tensor of expansion coefficients.

Determine the univariate quadrature nodes and weights in each dimension, $\{(\mathbf{x}_n, \mathbf{w}_n)\}_{n=1}^{d}$, where $\mathbf{x}_n = \{x_n^{(i)}\}_{i=0}^{N_n}$ and $\mathbf{w}_n = \{w_n^{(i)}\}_{i=0}^{N_n}$

Construct the ε–accurate approximation $\mathcal{B}_{TT}$ of $h(\mathcal{X}_{\mathrm{i}}) = f(\mathcal{X}_{\mathrm{i}})\sqrt{\mathcal{W}_{\mathrm{i}}}$ using `TT-DMRG-cross`

Recover the approximation of $f(\mathcal{X})$ as $\mathcal{A}_{TT}^{w} = \mathcal{B}_{TT}/\sqrt{\mathcal{W}}$, with cores $\{G_n\}_{n=1}^{d}$ and associated TT ranks $\mathbf{r}$

for $n := 1$ to d **do**

    for $i_n := 0$ to N_n **do**

        for all $(\alpha_{n-1}, \alpha_n) \in [0, r_{n-1}] \times [0, r_n]$ **do**

            $\hat{\beta}_n(\alpha_{n-1}, i_n, \alpha_n) = \sum_{j=0}^{N_n} G_n(\alpha_{n-1}, j, \alpha_n)\phi_{i_n,n}(x_n^{(j)})w_n^{(j)}$

        end for

    end for

end for

return $\left\{\hat{\beta}_n\right\}_{n=1}^{d}$

Procedure 2 `FTT-projection-evaluation`

Input: Cores $\{\hat{\beta}_n(\alpha_{n-1}, i_n, \alpha_n)\}_{n=1}^{d}$ obtained through `FTT-projection-construction`; N^y evaluation points $\mathbf{y}^{(i)} := \{y_1^{(i)}, \ldots, y_d^{(i)}\} \in \mathbf{I}$, $i \in [1, N^y]$, collected in the $N^y \times d$ matrix $\mathbf{Y} := \{\mathbf{y}_1, \ldots, \mathbf{y}_d\}$.

Output: Polynomial approximation $\widetilde{P}_{\mathbf{N}} f_{TT}(\mathbf{Y})$ of $f(\mathbf{Y})$.

for $n := 1$ to d **do**
    for $i := 1$ to N^y **do**
        for all $(\alpha_{n-1}, \alpha_n) \in [0, r_{n-1}] \times [0, r_n]$ **do**
            $\hat{G}_n(\alpha_{n-1}, i, \alpha_n) = \sum_{j=0}^{N_n} \hat{\beta}_n(\alpha_{n-1}, j, \alpha_n)\phi_{j,n}(y_n^{(i)})$
        end for
    end for
end for
$\boldsymbol{\mathcal{B}}_{TT}(i_1, \ldots, i_d) = \sum_{\alpha_0, \ldots, \alpha_d = 1}^{\mathbf{r}} \hat{G}_1(\alpha_0, i_1, \alpha_1) \cdots \hat{G}_d(\alpha_{d-1}, i_d, \alpha_d)$
return $\widetilde{P}_{\mathbf{N}} f_{TT}(\mathbf{Y}) := \{\boldsymbol{\mathcal{B}}_{TT}(i, \ldots, i)\}_{i=1}^{N^y}$

Interpolation

Define the hat function,

$$e_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{if } x_{i-1} \leq x \leq x_i \; and \; x \geq a, \\ \frac{x - x_{i+1}}{x_i - x_{i+1}} & \text{if } x_i \leq x \leq x_{i+1} \; and \; x \leq b, \\ 0 & \text{otherwise.} \end{cases}$$

The multilinear interpolation operator is defined as,

$$I_N f(\mathbf{x}) = \sum_{\mathbf{i}=0}^{\mathbf{N}} \hat{c}_{\mathbf{i}} e_{\mathbf{i}}(\mathbf{x}), \quad \hat{c}_{\mathbf{i}} = f(\mathbf{x}_{\mathbf{i}}),$$

where $\{\mathbf{x}_{\mathbf{i}}\}_{\mathbf{i}=0}^{\mathbf{N}} = \left\{x_i^1\right\}_{i=0}^{N_1} \times \cdots \times \left\{x_i^d\right\}_{i=0}^{N_d}$ is a tensor grid of points.

Procedure 3 `FTT-interpolation-evaluation`

Input: Tensor of interpolation points $\boldsymbol{\mathcal{X}} = \times_{n=1}^{d} \mathbf{x}_n$, where $\mathbf{x}_n = \{x_n^{(i)}\}_{i=1}^{N_n^x} \subseteq I_n$; ε-accurate approximation $\boldsymbol{\mathcal{A}}_{TT}^{w}$ (in general) or $\boldsymbol{\mathcal{A}}_{TT}$ (uniform μ, linear interpolation, equispaced points) of $f(\boldsymbol{\mathcal{X}})$ obtained by `TT-DMRG-cross`, with cores $\{G_n\}_{n=1}^{d}$ and TT ranks $\mathbf{r}$; evaluation points $\mathbf{y}^{(i)} := \{y_1^{(i)}, \ldots, y_d^{(i)}\} \in \mathbf{I}$, $i \in [1, N^y]$, collected in the $N^y \times d$ matrix $\mathbf{Y} := \{\mathbf{y}_1, \ldots, \mathbf{y}_d\}$.

Output: Interpolated approximation $I_{\mathbf{N}} f_{TT}(\mathbf{Y})$ or $\Pi_{\mathbf{N}} f_{TT}(\mathbf{Y})$ of $f(\mathbf{Y})$.

Construct list $\left\{L^{(i)}\right\}_{i=1}^{d}$ of $N^y \times N_i^x$ (linear or Lagrange) interpolation matrices from $\mathbf{x}_i$ to $\mathbf{y}_i$
for $n := 1$ to d **do**
    for all $(\alpha_{n-1}, \alpha_n) \in [0, r_{n-1}] \times [0, r_n]$ **do**
        $\hat{G}_n(\alpha_{n-1}, :, \alpha_n) = L^{(n)} G_n(\alpha_{n-1}, :, \alpha_n)$
    end for
end for
$\boldsymbol{\mathcal{B}}_{TT}(i_1, \ldots, i_d) = \sum_{\alpha_0, \ldots, \alpha_d = 1}^{\mathbf{r}} \hat{G}_1(\alpha_0, i_1, \alpha_1) \cdots \hat{G}_d(\alpha_{d-1}, i_d, \alpha_d)$
return $I_{\mathbf{N}} f_{TT}(\mathbf{Y}) := \{\boldsymbol{\mathcal{B}}_{TT}(i, \ldots, i)\}_{i=1}^{N^y}$

Outline

Numerical examples: Genz functions

$$f_1(\mathbf{x}) = \cos\left(2\pi w_1 + \sum_{i=1}^{d} c_i x_i\right), \qquad f_2(\mathbf{x}) = \prod_{i=1}^{d} \left(c_i^{-2} + (x_i + w_i)^2\right)^{-1},$$

$$f_3(\mathbf{x}) = \left(1 + \sum_{i=1}^{d} c_i x_i\right)^{-(d+1)}, \qquad f_4(\mathbf{x}) = \exp\left(-\sum_{i=1}^{d} c_i^2 (x_i - w_i)^2\right),$$

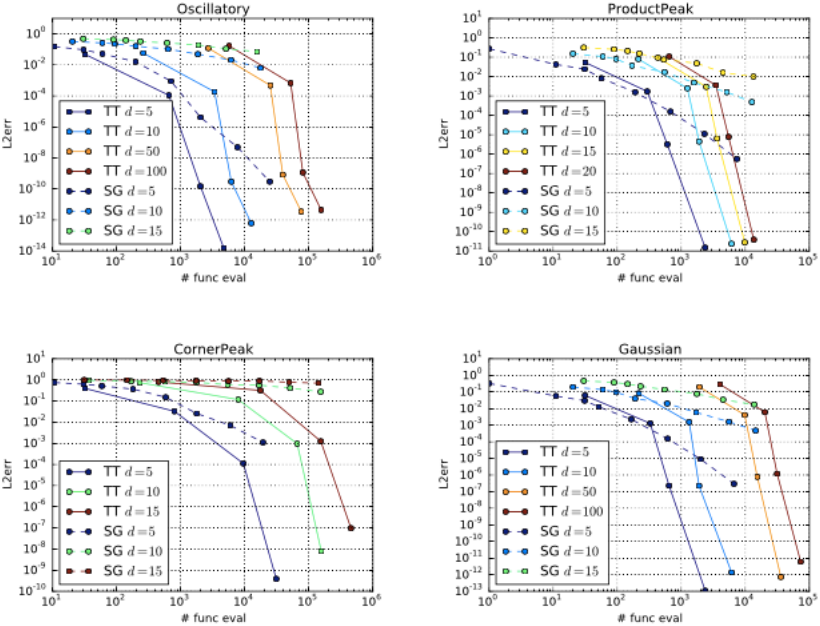
$$f_5(\mathbf{x}) = \exp\left(-\sum_{i=1}^{d} c_i^2 |x_i - w_i|\right), \qquad f_6(\mathbf{x}) = \begin{cases} 0 & \text{if } x_1 > w_1 \text{ or } x_2 > w_2, \\ \exp\left(\sum_{i=1}^{d} c_i x_i\right) & \text{otherwise,} \end{cases}$$

- $\mathbf{w}$ are drawn uniformly from $[0,1]$.
- Classically, $\mathbf{c}$ are drawn uniformly from $[0,1]$ and then are normalized. In this paper the normalization is not performed.
- Relative L^2 errors are computed via,

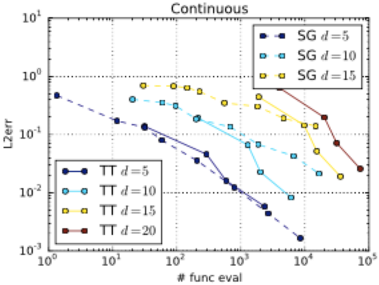
$$e_{rel} := \| f - \mathscr{L} f_{TT} \|_{L^2_\mu(\mathbf{I})} / \| f \|_{L^2_\mu(\mathbf{I})},$$

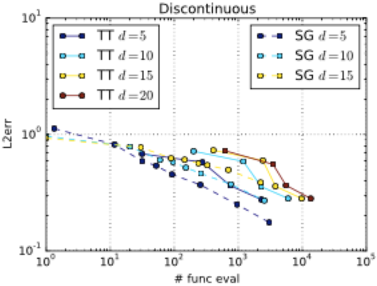
where $\mathscr{L}$ is either the projection or interpolation operator. Integrals were evaluated via Monte Carlo, with samples drawn until relative error of e_{rel} was less than 10^{-2}.

L2 error vs. # of function evaluations

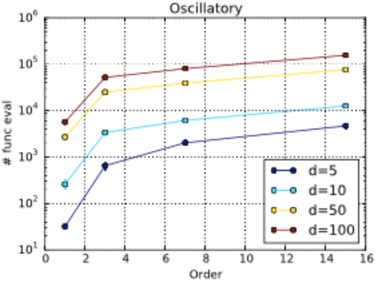


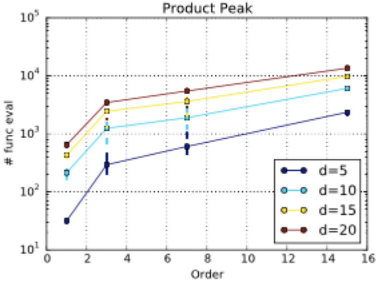
L2 error vs. # of function evaluations cont'd

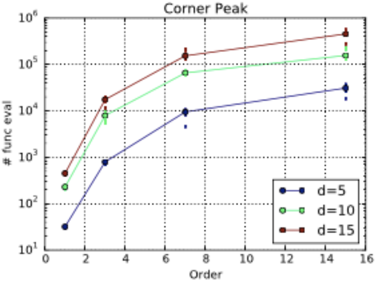


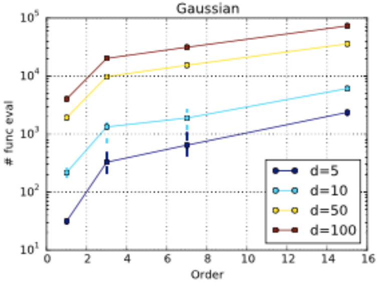


of function evaluations vs. order

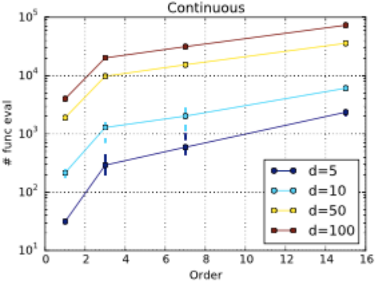


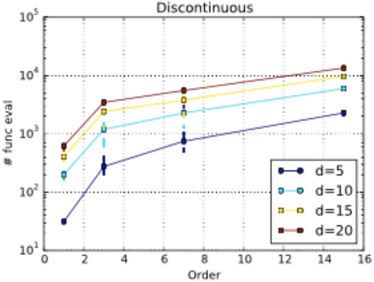




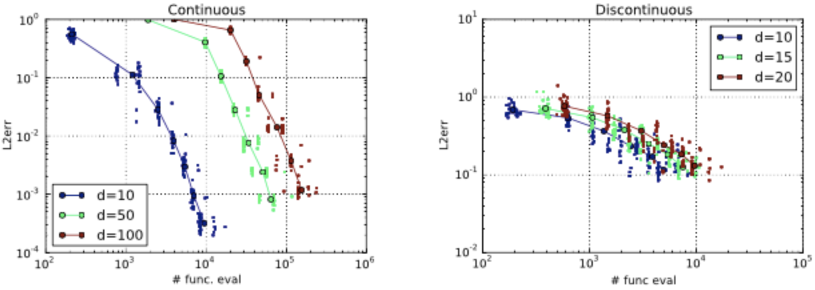


of function evaluations vs. order cont'd





Linear interpolation of Genz functions



Elliptic PDE with random coefficient

$$
\begin{aligned}
-\nabla\cdot(\kappa(\mathbf{x},\omega)\nabla u(\mathbf{x},\omega)) &= 1, \quad \text{in } \Gamma\times\Omega, \\
u(\mathbf{x},\omega) &= 0, \quad \text{on } \delta\Gamma\times\Omega,
\end{aligned}
$$

defined on the unit square $\Gamma=(0,1)\times(0,1)$ with boundary $\partial\Gamma$. The diffusion coefficient $\kappa(\mathbf{x},\mathbf{z})$ is a log-normal random field defined on the probability space (Ω,Σ,μ) by

$$
\kappa(\mathbf{x},\mathbf{z}) = \exp\left(g\left(\mathbf{x},\omega\right)\right), \;\; g\left(\mathbf{x},\omega\right) \sim \mathcal{N}\left(\mathbf{0}, C_g\left(\mathbf{x},\mathbf{x}'\right)\right).
$$

g is characterized by the covariance kernel

$$
C_g\left(\mathbf{x},\mathbf{x}'\right) = \int_\Omega g\left(\mathbf{x},\omega\right) g\left(\mathbf{x}',\omega\right) d\mu\left(\omega\right) = \sigma^2 \exp\left(-\frac{\left\|\mathbf{x}-\mathbf{x}'\right\|^2}{2l^2}\right)
$$

Elliptic PDE with random coefficient cont'd

where $l > 0$ is the spatial correlation length. The random field is decomposed via the KL expansion,

$$g(\mathbf{x}, \omega) = \sum_{i=1}^{\infty} \sqrt{\zeta_k} \chi_i(\mathbf{x}) Y_i(\omega),$$

where $Y_i \sim \mathcal{N}(0,1)$ and $\{\lambda_i, \chi_i(\mathbf{x})\}_{i=1}^{\infty}$ are eigenpairs of

$$\int_{\Gamma} C_g(\mathbf{x}, \mathbf{x}') \chi_i(\mathbf{x}') d\mathbf{x}' = \lambda_i \chi_i(\mathbf{x}).$$

In this example $d = 12$ s.t. $\sum_{i=1}^{d} \lambda_i \geq .95\sigma^2$, $l = 0.25$, $\sigma^2 = 0.1$, and we approximate $u(\mathbf{x}_0, \mathbf{y})$ at $\mathbf{x}_0 = (0.75, 0.25)$.

Elliptic PDE with random coefficient cont'd

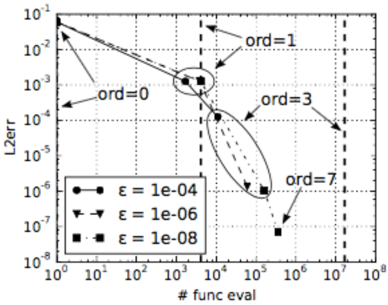


Figure 1: Convergence of the FTT-projection of hermite polynomial basis functions of orders 0, 1, 3, 7 for different target accuracies.

Tensor toolbox

Open source STT approximation algorithm:
http://pypi.python.org/pypi/TensorToolbox/