SAND2017-8284C

Low-rank functional decompositions with applications to stochastic optimal control

2017 Meeting of the Linear Algebra Society
July 24-28, 2017
Ames, IA

Alex Gorodetsky

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Takeaways

Sandia National Laboratories

1. Low-rank functions $=$ representing multivariate functions with separable expressions
2. Low-rank functions are related to tensors when performing approximation with a linear *basis*
3. Low-rank functions can use *nonlinear* approximation $\rightarrow \neq$ tensors
4. ALS is not always best for finding low-rank decompositions

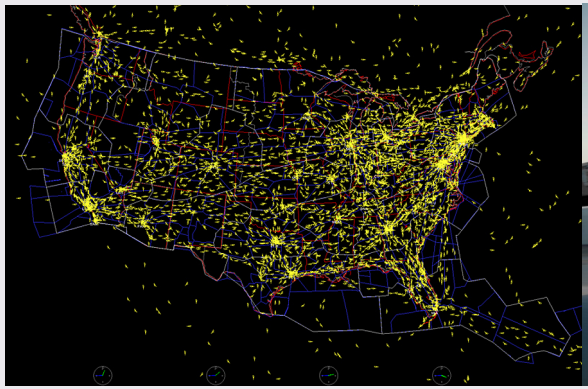
Stochastic optimal control
Challenge: representing a parametric objective function

Sandia National Laboratories

$$J(x_0) = \min_{u(t)} \mathbb{E}\left[\int g(x(t), u(t)dt)\right], \quad x(0) = x_0$$

subject to a stochastic differential equation

$$dx = F(x, u)dt + \sigma(x)dw(t), \quad x \in \mathbb{R}^d,$$



Approximating black-box functions

Sandia National Laboratories

Applications and challenges:

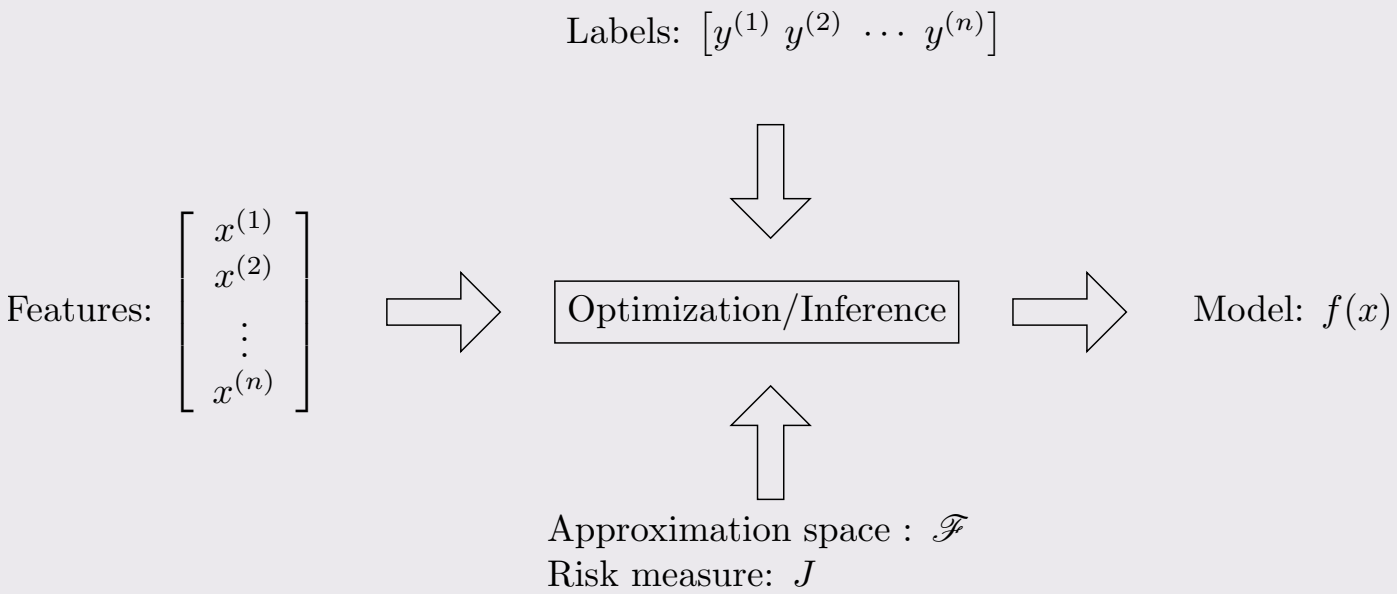
- Stochastic control: repeat applications of Bellman Operator $J^{k+1}(x) = T(J^k(x))$
  1. many approximations
  2. high-dimensional states
  3. expensive evaluations
- Data analysis: generate empirical models from data
  1. high-dimensional states
  2. large amounts of data
- Uncertainty quantification: analyze complex physical systems
  1. high-dimensional states
  2. expensive evaluations
  3. expensive analysis

Questions

1. How do we interrogate a function to accurately reconstruct it?
2. Someone else interrogated the function, how do I interpret the results?

Scalable representations of functions

Sandia National Laboratories



- Parametric approaches for $\mathscr{F}$:
  - Good: statistical properties, stability
  - Bad: curse-of-dimensionality on basis, expressivity
- Nonparametric approaches for $\mathscr{F}$:
  - Good: expressivity
  - Bad: excess risk scales exponentially with dimensionality, expensive

Common parametric approaches

Sandia National Laboratories

- Approximation space with linear basis $\left\{\phi^{(i)}\right\}_{i=1}^{P}$

$$\Psi = \left[\phi^{(1)}(\boldsymbol{x}) \ \ldots \ \phi^{(P)}(\boldsymbol{x})\right]$$

- Many algorithmic setups
  - Ridge regression:

$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_2$$

  - $L1$ regression:

$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_1$$

  - $L0$ regression:

$$\min \|\Psi\boldsymbol{\theta} - \boldsymbol{y}\| + \lambda\|\boldsymbol{\theta}\|_0$$

- Constructing Ψ is problematic in high-dimensions
  - Total order expansions: exponential growth of terms
  - Low order expansion: limited expressivity

Typical setup
Linear setup yields curse-of-dimensionality

Sandia National Laboratories

Inputs are from a *tensor product space* and inputs are *real*

$$x_i \in \mathcal{X}_i, \quad x = (x_1, x_2, \ldots, x_d) \in \mathcal{X}, \quad \mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_d$$

Determine a basis for each dimension

$$\left(\phi_k^{(i)} : \mathcal{X}_k \to \mathbb{R}\right)_{k=1}^{n}$$

1. Global basis, e.g., orthonormal polynomials
2. Multi-resolution basis, e.g., wavelets
3. Localized or kernels

Basis for f obtained by *combinations*

$$\phi^{\boldsymbol{i}} = \phi_1^{(i_1)} \times \phi_2^{(i_2)} \times \cdots \times \phi_d^{(i_d)}, \ \boldsymbol{i} = (i_1, i_2, \ldots, i_d)$$

1. Tensor-product: all combinations $(i_k)_1^{n_k} \to n^d$
2. Some subsets: $\|\boldsymbol{i}\| < n \to \mathcal{O}(n^d)$

Tensors arise!
Linear approximation with tensor-product basis

Sandia National Laboratories

The coefficients of a tensor-product basis form a tensor

$$f(x) = \sum_{i_1=1}^{n_1} \sum_{i_2=2}^{n_2} \cdots \sum_{i_d=d}^{n_d} \boldsymbol{\theta}_{i_1 i_2 \ldots i_d} \phi_{1 i_1}(x_1) \phi_{2 i_2}(x_2) \cdots \phi_{d i_d}(x_d)$$

- Exponentially growing number of coefficients reduces feasibility
- Need dimension reduction methods
- Tensor-decompositions are a multilinear compressed representations

Tensors arise!
Linear approximation with tensor-product basis

Sandia National Laboratories

The coefficients of a tensor-product basis form a tensor

$$f(x) = \sum_{i_1=1}^{n_1} \sum_{i_2=2}^{n_2} \cdots \sum_{i_d=d}^{n_d} \boldsymbol{\theta}_{i_1 i_2 \ldots i_d} \phi_{1 i_1}(x_1) \phi_{2 i_2}(x_2) \cdots \phi_{d i_d}(x_d)$$

- Exponentially growing number of coefficients reduces feasibility
- Need dimension reduction methods
- Tensor-decompositions are a multilinear compressed representations

Various decompositions:

→ Doostan et al., 2013, Mathelin 2014, Chevreuil et al., 2015., Bigoni and Marzouk, 2016

Tensors arise!
Linear approximation with tensor-product basis

Sandia National Laboratories

The coefficients of a tensor-product basis form a tensor

$$f(x) = \sum_{i_1=1}^{n_1} \sum_{i_2=2}^{n_2} \cdots \sum_{i_d=d}^{n_d} \boldsymbol{\theta}_{i_1 i_2 \ldots i_d} \phi_{1 i_1}(x_1) \phi_{2 i_2}(x_2) \cdots \phi_{d i_d}(x_d)$$

- Exponentially growing number of coefficients reduces feasibility
- Need dimension reduction methods
- Tensor-decompositions are a multilinear compressed representations

Various decompositions:

1. CP: $\boldsymbol{\theta} = \sum_{i=1}^{R} \boldsymbol{a}_{1i} \circ \boldsymbol{a}_{2i} \circ \cdots \circ \boldsymbol{a}_{di}$

→ Doostan et al., 2013, Mathelin 2014, Chevreuil et al., 2015., Bigoni and Marzouk, 2016

Tensors arise!
Linear approximation with tensor-product basis

Sandia National Laboratories

The coefficients of a tensor-product basis form a tensor

$$f(x) = \sum_{i_1=1}^{n_1} \sum_{i_2=2}^{n_2} \cdots \sum_{i_d=d}^{n_d} \boldsymbol{\theta}_{i_1 i_2 \ldots i_d} \phi_{1 i_1}(x_1) \phi_{2 i_2}(x_2) \cdots \phi_{d i_d}(x_d)$$

- Exponentially growing number of coefficients reduces feasibility
- Need dimension reduction methods
- Tensor-decompositions are a multilinear compressed representations

Various decompositions:

1. CP: $\boldsymbol{\theta} = \sum_{i=1}^{R} \boldsymbol{a}_{1i} \circ \boldsymbol{a}_{2i} \circ \cdots \circ \boldsymbol{a}_{di}$
2. Tucker: $\boldsymbol{\theta} = \sum_{j_1=1}^{r_1} \sum_{j_2=2}^{r_2} \cdots \sum_{j_d=d}^{r_d} \hat{\boldsymbol{\theta}}_{j_1 j_2 \ldots j_d} \boldsymbol{a}_{1 j_1} \circ \boldsymbol{a}_{2 j_2} \circ \cdots \circ \boldsymbol{a}_{d j_d}$

→ Doostan et al., 2013, Mathelin 2014, Chevreuil et al., 2015., Bigoni and Marzouk, 2016

Tensors arise!
Linear approximation with tensor-product basis

Sandia National Laboratories

The coefficients of a tensor-product basis form a tensor

$$f(x) = \sum_{i_1=1}^{n_1} \sum_{i_2=2}^{n_2} \cdots \sum_{i_d=d}^{n_d} \boldsymbol{\theta}_{i_1 i_2 \ldots i_d} \phi_{1 i_1}(x_1) \phi_{2 i_2}(x_2) \cdots \phi_{d i_d}(x_d)$$

- Exponentially growing number of coefficients reduces feasibility
- Need dimension reduction methods
- Tensor-decompositions are a multilinear compressed representations

Various decompositions:

1. CP: $\boldsymbol{\theta} = \sum_{i=1}^{R} \boldsymbol{a}_{1i} \circ \boldsymbol{a}_{2i} \circ \cdots \circ \boldsymbol{a}_{di}$
2. Tucker: $\boldsymbol{\theta} = \sum_{j_1=1}^{r_1} \sum_{j_2=2}^{r_2} \cdots \sum_{j_d=d}^{r_d} \hat{\boldsymbol{\theta}}_{j_1 j_2 \ldots j_d} \boldsymbol{a}_{1 j_1} \circ \boldsymbol{a}_{2 j_2} \circ \cdots \circ \boldsymbol{a}_{d j_d}$
3. Tensor-train: $\boldsymbol{\theta}_{i_1 i_2 \ldots i_d} = \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \cdots \sum_{j_d=1}^{r_d} \boldsymbol{a}_{1 j_1 i_1} \boldsymbol{a}_{2 j_1 j_2 i_2} \cdots \boldsymbol{a}_{d j_d i_d}$

→ Doostan et al., 2013, Mathelin 2014, Chevreuil et al., 2015., Bigoni and Marzouk, 2016

Issues and solutions
Functional tensor-train (FT) as a separable ansatz

Sandia National Laboratories

- Would like to *adapt* to local and global structure
- Would like computationally *efficient*, and *flexible* approximation format
- Algorithms for decomposing functions into FT format

$$f(x_1, x_2, \ldots, x_d) = \sum_{i_0=1}^{r_0} \sum_{i_1=1}^{r_1} \cdots \sum_{i_d=1}^{r_d} f_1^{(i_0 i_1)}(x_1) f_2^{(i_1 i_2)}(x_2) \ldots f_d^{(i_{d-1} i_d)}(x_d)$$
$$= \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\ldots\mathcal{F}_d(x_d)$$

$$\underbrace{\begin{bmatrix} f_k^{(11)}(x_k) & \cdot & f_k^{(1 r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1} 1)}(x_k) & \cdot & f_k^{(r_{k-1} r_k)}(x_k) \end{bmatrix}}_{\mathcal{F}_k(x_k)}$$

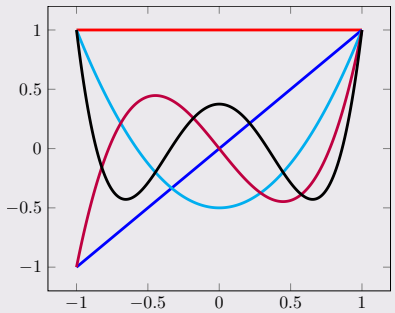
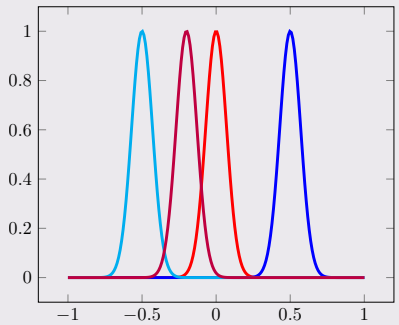
$$f(x_1, x_2, :, x_4) =$$

→ Oseledets 2013, **G.** et al., A continuous extension of the tensor-train 2016

Linear and nonlinear parameterizations

Sandia National Laboratories

Each $f_k^{(i_{k-1}i_k)}$ can be represented *independently*

	Linear	Nonlinear
		
Evaluation	$\sum_{\ell=1}^{n_{kij}} \boldsymbol{a}_{kij\ell}\phi_{k\ell}^{(ij)}(x_k)$	$\sum_{\ell=1}^{n_{kij}} \exp\left((x_k - \boldsymbol{a}_{kij\ell})^2\right)$
Gradient	$\phi_{k\ell}^{(ij)}(x_k)$	$-2\left(x_k - \boldsymbol{a}_{kij\ell}\right)\exp\left((x_k - \boldsymbol{a}_{kij\ell})^2\right)$

- Linear: gradient independent of parameter
  $\rightarrow$ faster optimization, lower expressivity
- Nonlinear: gradient dependent on parameters
  $\rightarrow$ slower optimization, higher expressivity

Special case: recovering existing models
Or — converting to low-rank tensors

Sandia National Laboratories

Modeling choice

- *linear* parameterization univariate functions
- *identitical basis* expansions within a core, i.e., $n_{kij} = n_k$ and $\phi_{k\ell}^{(ij)} = \phi_{k\ell}$

- Define a TT-core $\boldsymbol{\mathcal{F}}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$ such that

$$\boldsymbol{\mathcal{F}}_k[:, \ell, :] = \begin{bmatrix} \boldsymbol{a}_{k11\ell} & \cdot & \boldsymbol{a}_{k1r_k\ell} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a}_{kr_{k-1}1\ell} & \cdot & \boldsymbol{a}_{kr_{k-1}r_k\ell} \end{bmatrix}$$

- TT-core and FT-core related according to

$$\mathcal{F}_k(x_k) = \sum_{\ell=1}^{n_k} \boldsymbol{\mathcal{F}}_k[:, \ell, :] \phi_{k\ell}(x_k).$$

- TT-approximation of coefficients of a tensor-product of basis functions

$$f(x_1, \ldots, x_d) = \sum_{\ell_1=1}^{n_1} \cdots \sum_{\ell_d=1}^{n_d} \boldsymbol{\mathcal{F}}_1[:, \ell_1, :] \cdots \boldsymbol{\mathcal{F}}_d[:, \ell_d, :] \phi_{1\ell_1}(x_1) \cdots \phi_{d\ell_d}(x_d).$$

Two problem setups

Sandia National Laboratories

Compression with experimental design

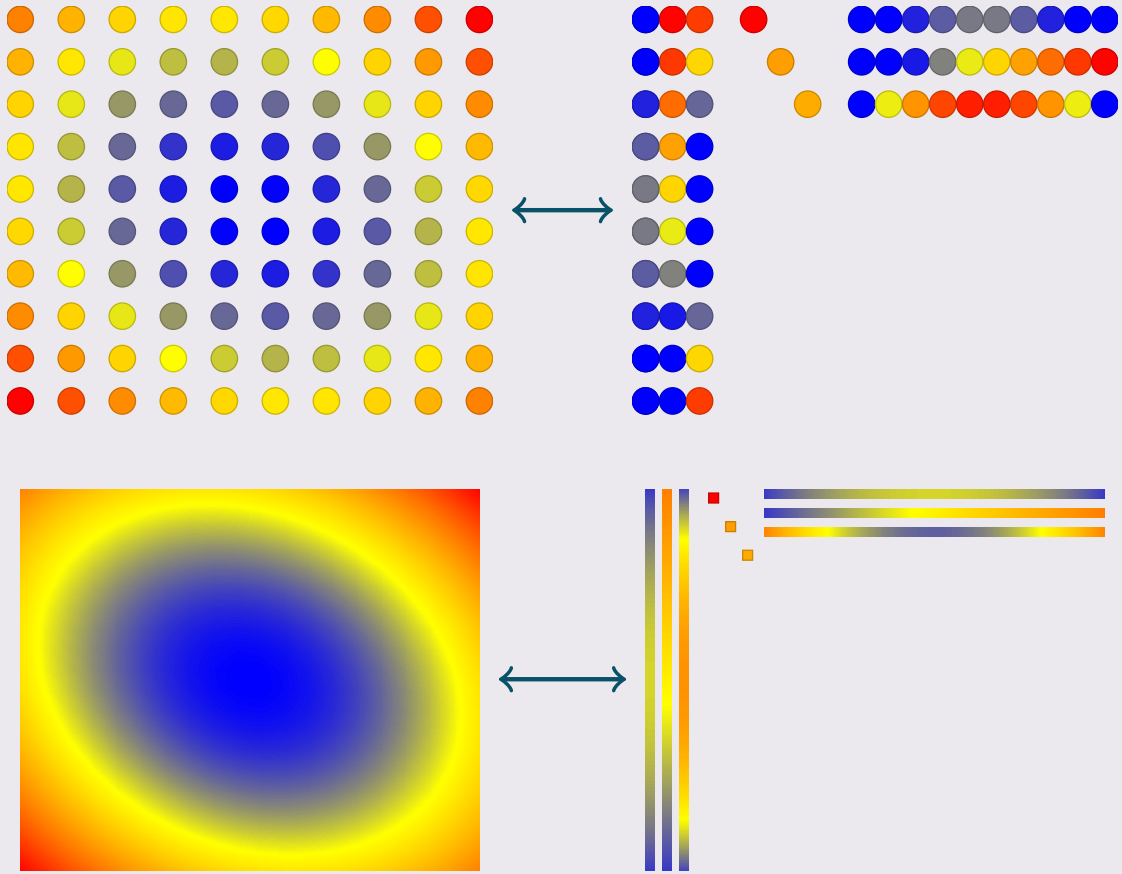
- Have ability to generate samples from f and would like to generate a compressed representation
- **Algorithm:** Cross-approximation with maximum volume
  - Univariate functions chosen from fibers.
  - Algebraically the same as the discrete cross-approximation of Oseledets 2010.
  - However, it is a continuous analogue in the spirt of Chebfun using continuous matrix factorizations (Townsend 2014)

Supervised learning

- We are given data and we parameterized each univariate function $f_k^{(i_{k-1}i_k)}$.
- **Algorithm:** solve an optimization problem for the coefficients
  - Least-squares for real-valued data
  - Cross-entropy for classification
  - Other differentiable objectives...

Compression

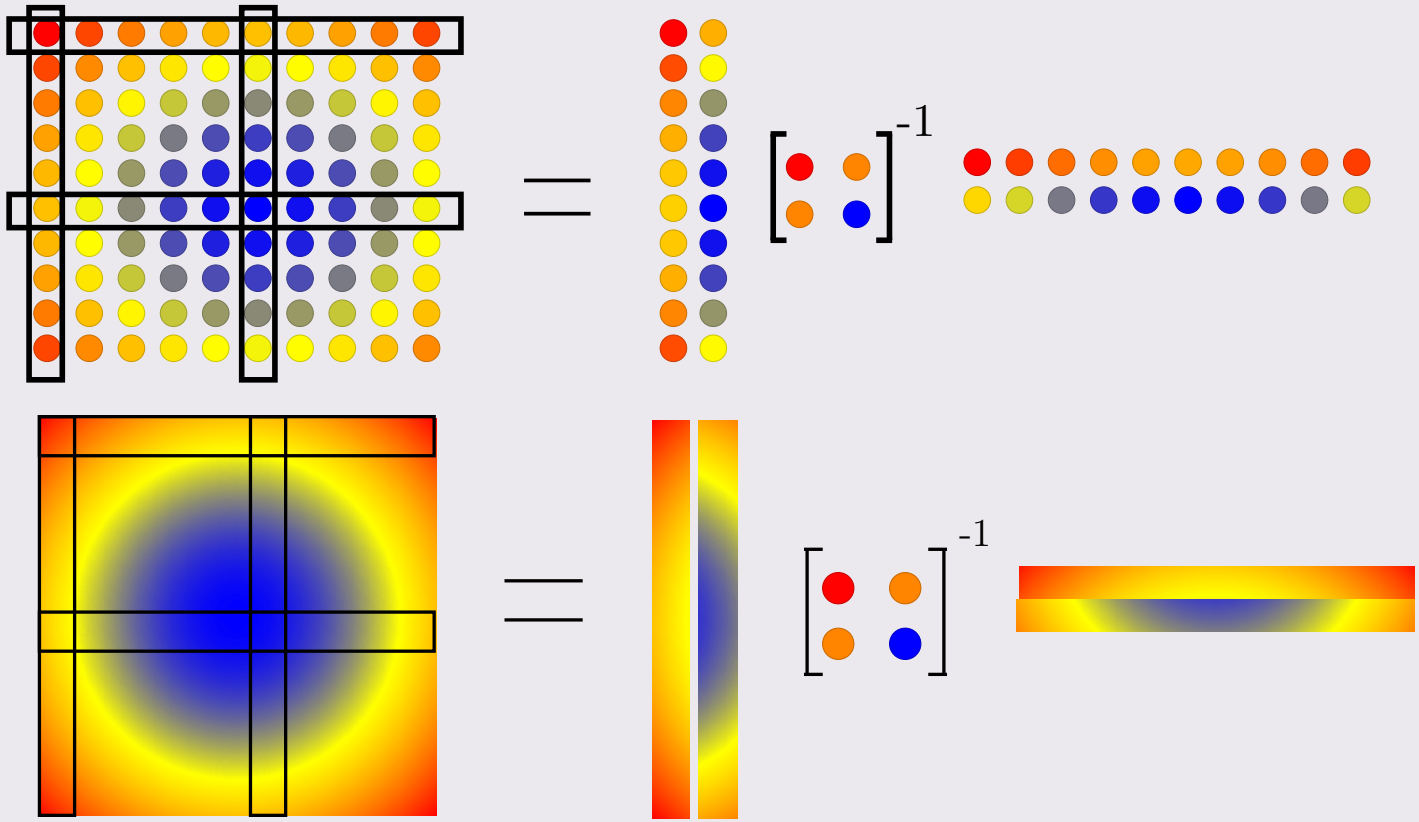
Compression of bivariate functions



- Matrix SVD - sum of outer products of vectors
- Functional SVD - sum of products of *univariate functions*

Replace SVD with a CUR/Skeleton decompositior

Sandia National Laboratories



- Compression with SVD expensive, evaluates all elements $\mathcal{O}(N^2)$
- Replace SVD with *another* (suboptimal) factorization: CUR
- Theorem: if $\exists$ exact rank r (f)SVD then $\exists$ exact rank r (f)CUR

Representation of CUR

Sandia National Laboratories

1. Fix "rows" $[x_1\ x_2]$ and "columns" $[y_1\ y_2]$

$$f(x,y) = \begin{bmatrix} f(x,y_1) & f(x,y_2) \end{bmatrix} \mathbf{G} \begin{bmatrix} f(x_1,y) \\ f(x_2,y) \end{bmatrix}$$

2. Approximate *univariate* row and column – e.g., in some basis

$$f(x,y) = \begin{bmatrix} \sum_{i=1}^{P} a_i\phi_i(x) & \sum_{i=1}^{P} b_i\phi_i(x) \end{bmatrix} \mathbf{G} \begin{bmatrix} \sum_{i=1}^{P} c_i\psi_i(y) \\ \sum_{i=1}^{P} d_i\psi_i(y) \end{bmatrix}$$

3. Store parameters of *each* row and column and **G** matrix $\mathcal{O}(Pr + r^2)$

Representation of CUR

Sandia National Laboratories

1. Fix "rows" $[x_1\ x_2]$ and "columns" $[y_1\ y_2]$

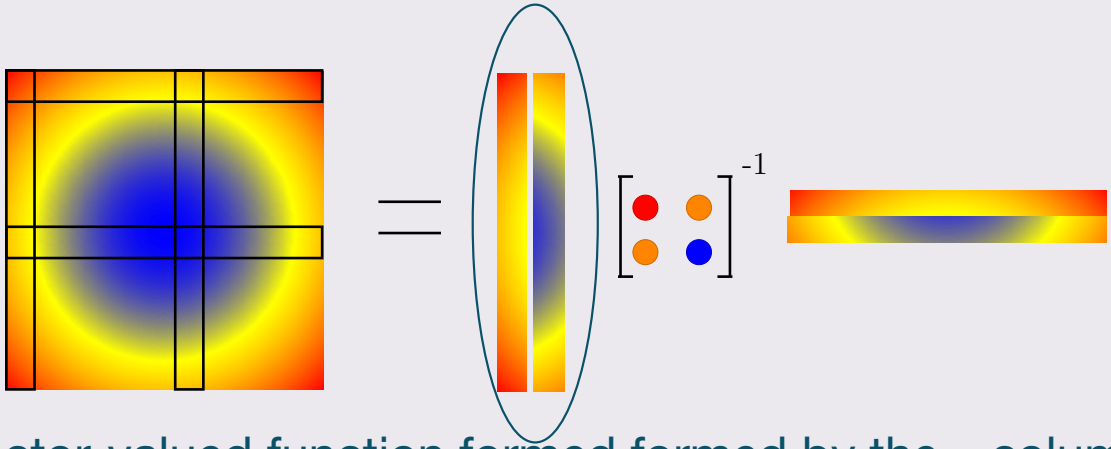
$$f(x,y) = \begin{bmatrix} f(x,y_1) & f(x,y_2) \end{bmatrix} \mathbf{G} \begin{bmatrix} f(x_1,y) \\ f(x_2,y) \end{bmatrix}$$

2. Approximate *univariate* row and column – e.g., in some basis

$$f(x,y) = \begin{bmatrix} \sum_{i=1}^{P_1} a_i \phi_i(x;\alpha) & \sum_{i=1}^{P_2} b_i \phi_i(x;\beta) \end{bmatrix} \mathbf{G} \begin{bmatrix} \sum_{i=1}^{P} c_i \psi_i(y) \\ \sum_{i=1}^{P} d_i \psi_i(y) \end{bmatrix}$$

3. Store parameters of *each* row and column and **G** matrix $\mathcal{O}(Pr + r^2)$

MAXVOL - cross approximation algorithm

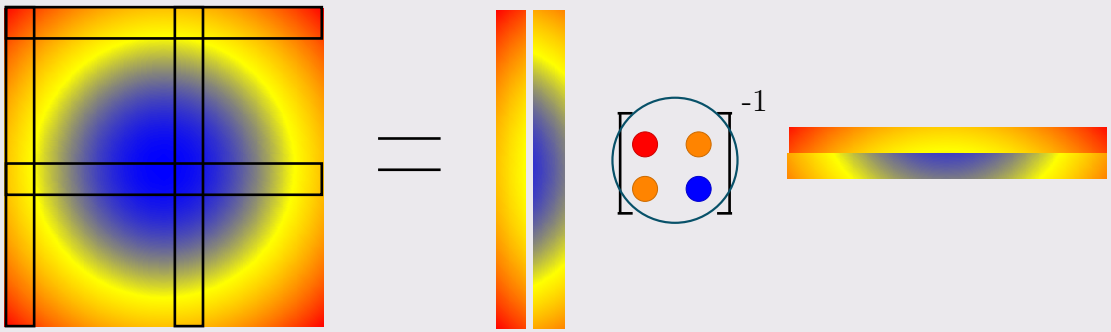


- Consider a vector-valued function formed formed by the r columns
- Let $r = 2$

$$\underbrace{\begin{bmatrix} f(x, y_1) & f(x, y_2) \end{bmatrix}}_{\text{vector-valued function}}$$

MAXVOL - cross approximation algorithm

Sandia National Laboratories



- Consider a vector-valued function formed formed by the r columns
- Let $r = 2$

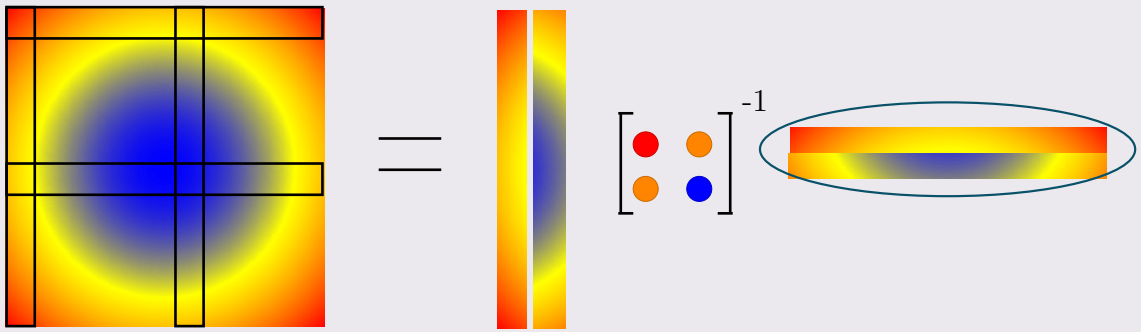
$$\underbrace{\begin{bmatrix} f(x, y_1) & f(x, y_2) \end{bmatrix}}_{\text{vector-valued function}}$$

- Seek values $[x_1, x_2]$ such that

$$\det\left(\begin{bmatrix} f(x_1, y_1) & f(x_1, y_2) \\ f(x_2, y_2) & f(x_2, y_2) \end{bmatrix}\right) \to \max$$

MAXVOL - cross approximation algorithm

Sandia National Laboratories



- Consider a vector-valued function formed formed by the r columns
- Let $r = 2$

$$\underbrace{\begin{bmatrix} f(x, y_1) & f(x, y_2) \end{bmatrix}}_{\text{vector-valued function}}$$

- Seek values $[x_1, x_2]$ such that

$$\det\left(\begin{bmatrix} f(x_1, y_1) & f(x_1, y_2) \\ f(x_2, y_2) & f(x_2, y_2) \end{bmatrix}\right) \to \max$$

- Fix $[x_1, x_2]$ and find new values $[y_1, y_2]$.

Maximizing the determinant

Sandia National Laboratories

- We extend the MAXVOL algorithm from [Oseledets 2010] to the continuous case
- Idea: find a dominant submatrix; its volume is not much smaller than that of the maximum-volume submatrix
- Dominant submatrix has property that

$$\underbrace{\begin{bmatrix} b_1(x) & b_2(x) \end{bmatrix}}_{B(x)} = \underbrace{\begin{bmatrix} f(x,y_1) & f(x,y_2) \end{bmatrix}}_{F(x)} \underbrace{\begin{bmatrix} f(x_1,y_1) & f(x_1,y_2) \\ f(x_2,y_2) & f(x_2,y_2) \end{bmatrix}^{-1}}_{\mathbf{G}}$$

$$\max_{i\in[1,\ldots,r],x} |b_i(x)| \leq 1$$

MAXVOL sub-optimality

Sandia National Laboratories

- SVD provides best rank r approximation

$$\|f(x,y) - f_{SVD}^{(r)}\|_{L^2}^2 = \sum_{i=r+1}^{\infty} \sigma_i^2$$

MAXVOL sub-optimality

Sandia National Laboratories

- SVD provides best rank r approximation

$$\|f(x,y) - f_{SVD}^{(r)}\|_{L^2}^2 = \sum_{i=r+1}^{\infty} \sigma_i^2$$

- Matrices: error bound relative to SVD singular values [Goreinov et. al. 1997,2001]

$$\|\boldsymbol{f} - \boldsymbol{f}_{CUR}^{(r)}\|_{\text{max}} = (r+1)\sigma_{r+1}$$

MAXVOL sub-optimality

Sandia National Laboratories

- SVD provides best rank r approximation

$$\|f(x,y) - f_{SVD}^{(r)}\|_{L^2}^2 = \sum_{i=r+1}^{\infty} \sigma_i^2$$

- Matrices: error bound relative to SVD singular values [Goreinov et. al. 1997,2001]

$$\|\boldsymbol{f} - \boldsymbol{f}_{CUR}^{(r)}\|_{\text{max}} = (r+1)\sigma_{r+1}$$

- Functions: error bound relative to functional SVD singular values [**G.**, Karaman, Marzouk 2015]

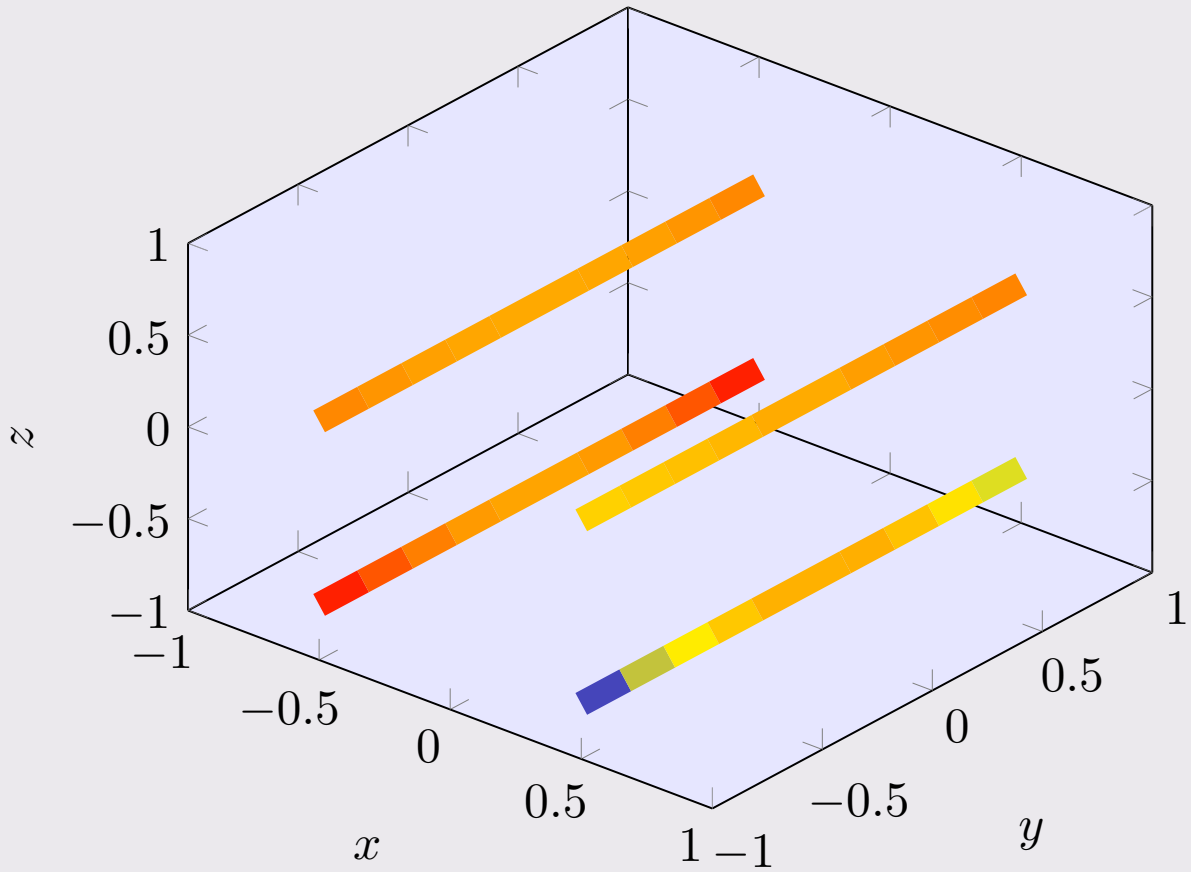
$$\|f(x,y) - f_{CUR}^{(r)}(x,y)\|_{\text{max}} \leq C(r+1)^{1+2\epsilon}\sigma_{r+1}$$

for $\epsilon > \frac{1}{2}$

High-dimensional cross approximation

Sandia National Laboratories

- Dimension sweeping algorithm
- Example – seeking a rank 2 $f(x, y, z)$
- Determine optimal y values from fixed x and z values
  - $[x_1, x_2] = [-0.5, 0.5]$ $[z_1, z_2] = [-0.7, 0.3]$



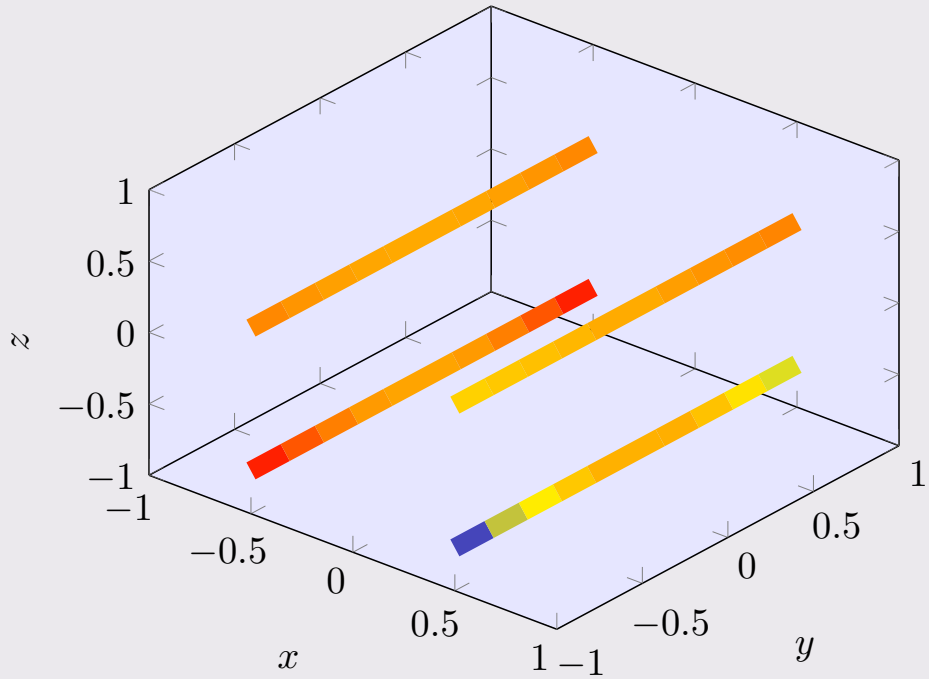
High-dimensional cross approximation

Sandia National Laboratories

- Dimension sweeping algorithm
- Example – seeking a rank 2 $f(x, y, z)$
- Determine optimal y values from fixed x and z values
  - $[x_1, x_2] = [-0.5, 0.5] \quad [z_1, z_2] = [-0.7, 0.3]$

1. Generate approximation of

$$\underbrace{\begin{bmatrix} f(x_1, y, z_1) & f(x_1, y, z_2) \\ f(x_2, y, z_1) & f(x_2, y, z_2) \end{bmatrix}}_{\text{matrix-valued function!}}$$



High-dimensional cross approximation

Sandia National Laboratories

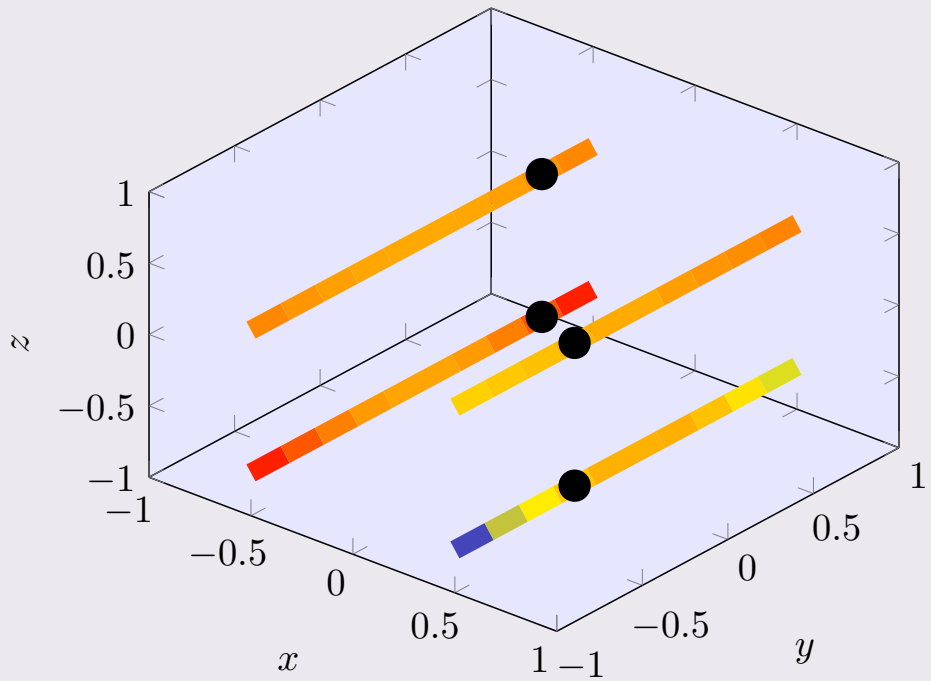
- Dimension sweeping algorithm
- Example – seeking a rank 2 $f(x, y, z)$
- Determine optimal y values from fixed x and z values
  - $[x_1, x_2] = [-0.5, 0.5] \quad [z_1, z_2] = [-0.7, 0.3]$

1. Generate approximation of

$$\underbrace{\begin{bmatrix} f(x_1, y, z_1) & f(x_1, y, z_2) \\ f(x_2, y, z_1) & f(x_2, y, z_2) \end{bmatrix}}_{\text{matrix-valued function!}}$$

2. Seek values $[y_1, y_2]$

$$\det\left(\begin{bmatrix} f(x_1, y_1, z_1) & f(x_1, y_1, z_2) \\ f(x_2, y_2, z_1) & f(x_2, y_2, z_2) \end{bmatrix}\right) \longrightarrow \max$$



High-dimensional cross approximation

Sandia National Laboratories

- Dimension sweeping algorithm
- Example – seeking a rank 2 $f(x, y, z)$
- Determine optimal y values from fixed x and z values
  - $[x_1, x_2] = [-0.5, 0.5] \quad [z_1, z_2] = [-0.7, 0.3]$

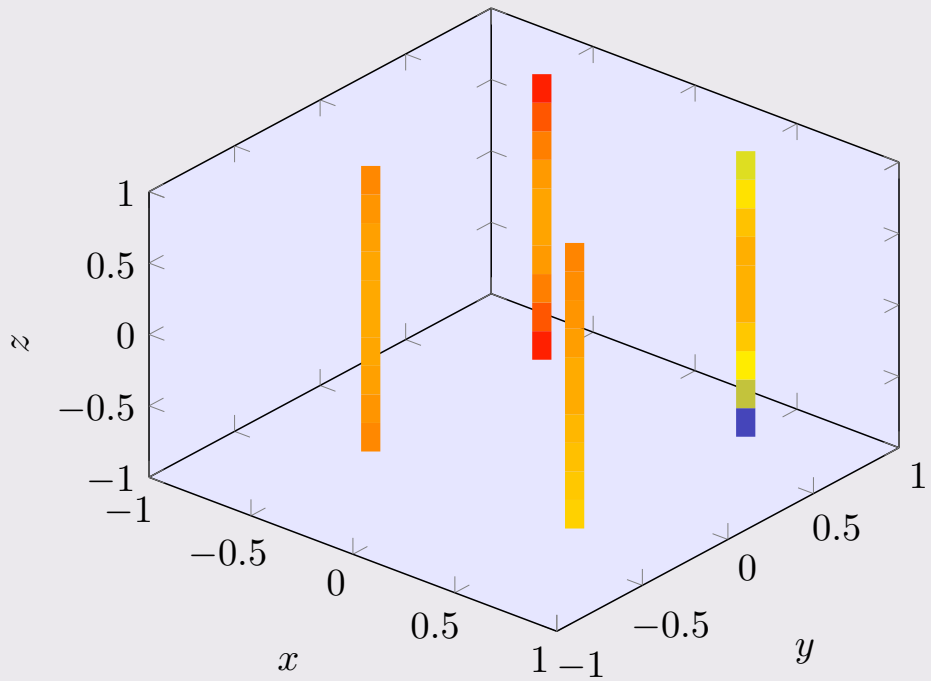
1. Generate approximation of

$$\underbrace{\begin{bmatrix} f(x_1, y, z_1) & f(x_1, y, z_2) \\ f(x_2, y, z_1) & f(x_2, y, z_2) \end{bmatrix}}_{\text{matrix-valued function!}}$$

2. Seek values $[y_1, y_2]$

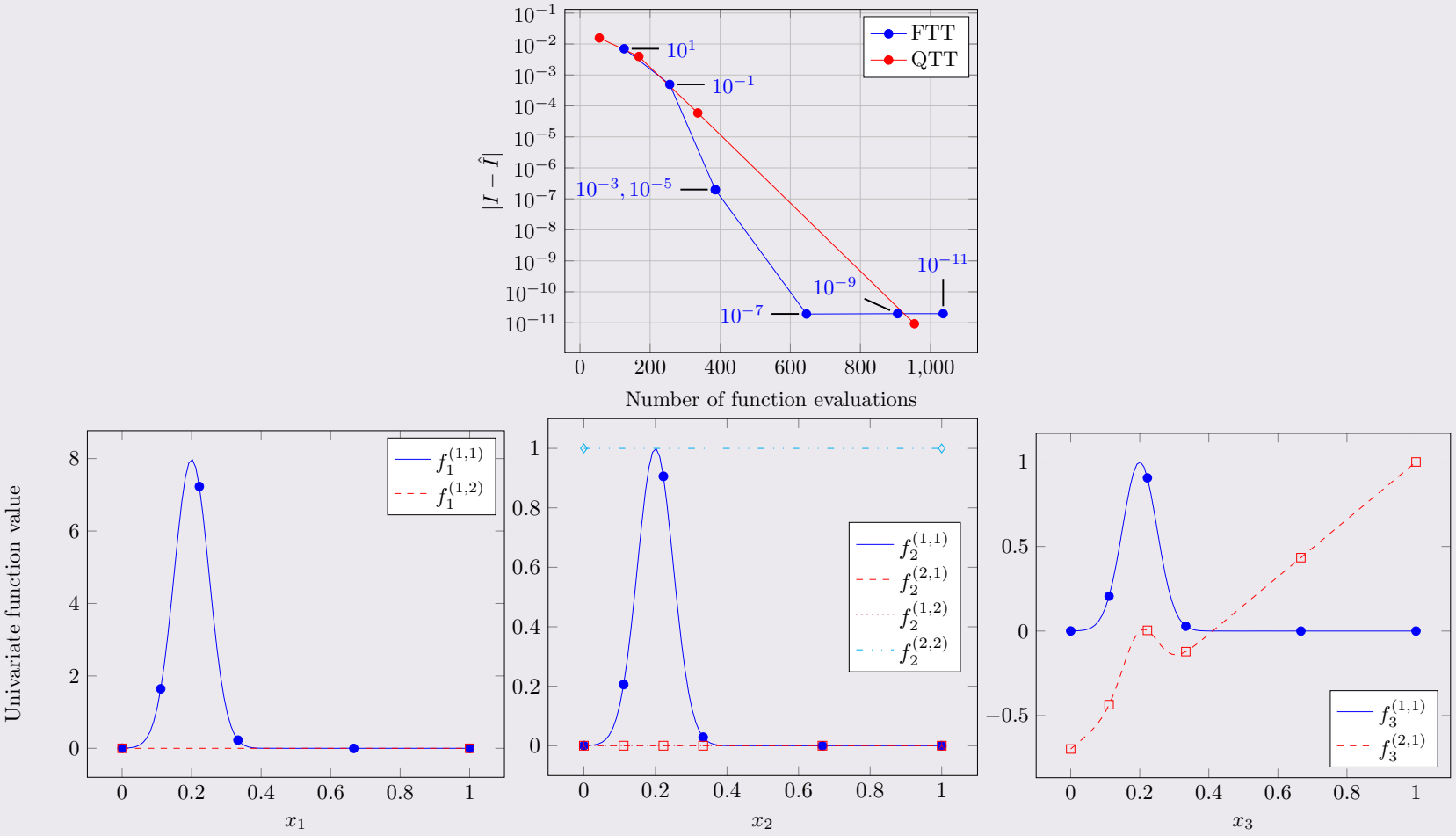
$$\det\left(\begin{bmatrix} f(x_1, y_1, z_1) & f(x_1, y_1, z_2) \\ f(x_2, y_2, z_1) & f(x_2, y_2, z_2) \end{bmatrix}\right) \longrightarrow \max$$

3. Fix $[x_1, x_2]$ and $[y_1, y_2]$ find new values $[z_1, z_2]$.



Comparison with QTT and DMRG - I
Benefits of adaptivity

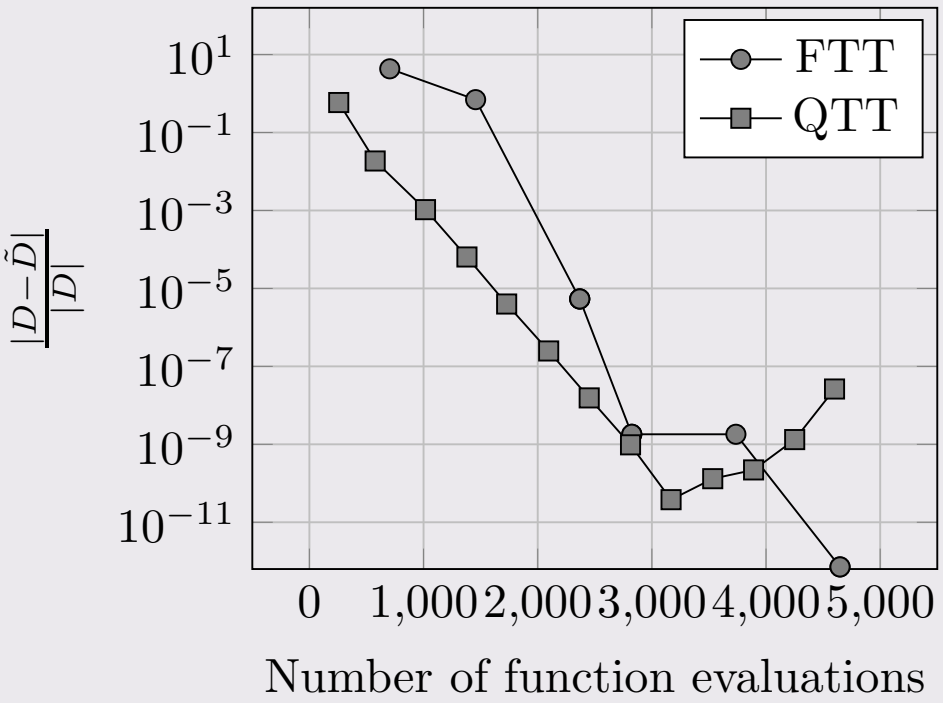
Sandia National Laboratories



Comparison with QTT and DMRG - II
Benefits of flexibility in representation

Sandia National Laboratories

Analytic derivative vs. finite difference approximation



Supervised Learning

Constrained least squares

Sandia National Laboratories

$$f^* = \arg\min_{f \in \mathscr{F}_{\boldsymbol{r}}} \quad \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - f\left(x^{(i)} \right) \right)^2$$

$$\mathscr{F}_{\boldsymbol{r}} = \left\{ f : \mathcal{X} \to \mathbb{R} \;\middle|\; \begin{array}{l} f = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\ldots\mathcal{F}_d(x_d) \\ \mathcal{F}_1 : \mathcal{X}_1 \to \mathbb{R}^{1 \times r_1} \\ \mathcal{F}_k : \mathcal{X}_k \to \mathbb{R}^{r_{k-1} \times r_k},\ 2 \leq k < d \\ \mathcal{F}_d : \mathcal{X}_d \to \mathbb{R}^{r_{d-1} \times 1} \end{array} \right\}$$

- Function space $\mathscr{F}_{\boldsymbol{r}}$ includes functions with ranks $< \boldsymbol{r}$.
  - Obtained by constraining certain blocks of $\mathcal{F}_k$ to be zero.
- Gradient based procedure for optimization
  - Alternating least squares
  - Quasi-Newton methods
  - Riemannian manifold

Gradient computation for single core

Sandia National Laboratories

Partial Derivative: $\frac{\partial f(x;\boldsymbol{a})}{\partial \boldsymbol{a_i}} = \mathcal{F}_{<k}(x_{<k}) \frac{\partial_i \mathcal{F}_k(x_k)}{\partial \boldsymbol{a_i}} \mathcal{F}_{>k}(x_{>k})$

Evaluation of left cores: $\mathcal{F}_{<k}(x_{<k}) = \mathcal{F}_1(x_1) \cdots \mathcal{F}_{k-1}(x_{k-1})$

Evaluation of right cores: $\mathcal{F}_{>k}(x_{>k}) = \mathcal{F}_{k+1}(x_{k+1}) \cdots \mathcal{F}_d(x_d)$

Gradient computation for single core

Sandia National Laboratories

Partial Derivative: $\frac{\partial f(x;\boldsymbol{a})}{\partial \boldsymbol{a_i}} = \mathcal{F}_{<k}(x_{<k}) \frac{\partial_i \mathcal{F}_k(x_k)}{\partial \boldsymbol{a_i}} \mathcal{F}_{>k}(x_{>k})$

Evaluation of left cores: $\mathcal{F}_{<k}(x_{<k}) = \mathcal{F}_1(x_1) \cdots \mathcal{F}_{k-1}(x_{k-1})$

Evaluation of right cores: $\mathcal{F}_{>k}(x_{>k}) = \mathcal{F}_{k+1}(x_{k+1}) \cdots \mathcal{F}_d(x_d)$

Assumption

Each univariate function in each FT-core is independently parameterized.

Proposition: Partial derivative of core $\mathcal{F}_k$

The partial derivative of core $\mathcal{F}_k$ with respect to $\boldsymbol{a}_{kij\ell}$ is a *sparse* matrix

$$\mathbf{G}[\alpha, \beta] = \begin{cases} \frac{\partial f_k^{(ij)}(x_k)}{\partial \boldsymbol{a}_{kij\ell}} & \text{if } \alpha = i, \beta = j \\ 0 & \text{otherwise} \end{cases}$$

for $\alpha = 1, \ldots, r_{k-1}$, and $\beta = 1, \ldots r_k$.

Gradient computation for single core

Sandia National Laboratories

Partial Derivative: $\frac{\partial f(x;\boldsymbol{a})}{\partial \boldsymbol{a_i}} = \mathcal{F}_{<k}(x_{<k}) \frac{\partial_i \mathcal{F}_k(x_k)}{\partial \boldsymbol{a_i}} \mathcal{F}_{>k}(x_{>k})$

Evaluation of left cores: $\mathcal{F}_{<k}(x_{<k}) = \mathcal{F}_1(x_1) \cdots \mathcal{F}_{k-1}(x_{k-1})$

Evaluation of right cores: $\mathcal{F}_{>k}(x_{>k}) = \mathcal{F}_{k+1}(x_{k+1}) \cdots \mathcal{F}_d(x_d)$

Assumption

Each univariate function in each FT-core is independently parameterized.

Proposition: Partial derivative of core $\mathcal{F}_k$

The partial derivative of core $\mathcal{F}_k$ with respect to $\boldsymbol{a}_{kij\ell}$ is a *sparse* matrix

$$\mathbf{G}[\alpha, \beta] = \begin{cases} \frac{\partial f_k^{(ij)}(x_k)}{\partial \boldsymbol{a}_{kij\ell}} & \text{if } \alpha = i, \beta = j \\ 0 & \text{otherwise} \end{cases}$$

for $\alpha = 1, \ldots, r_{k-1}$, and $\beta = 1, \ldots r_k$.

Given $\mathcal{F}_{<k}(x_{<k})$, $\mathcal{F}_{>k}(x_{>k})$, and $\frac{\partial f_k^{(ij)}(x_k)}{\partial \boldsymbol{a}_{kij\ell}}$, derivative requires $\rightarrow \mathcal{O}(1)$ ops.

Computational complexity

Sandia National Laboratories

- Let $\mathbf{\Gamma} = \mathcal{F}_{<k}(x_{<k})$ and $\mathbf{\Theta} = \mathcal{F}_{>k}(x_{>k})$.
- Then $\partial_{\boldsymbol{i}} f(x; \boldsymbol{a}) = \mathbf{\Gamma}^T \partial_{\boldsymbol{i}} \mathcal{F}_k(x_k) \mathbf{\Theta}$
- Let $G(n)$ denote the number of ops. to evaluate the gradient of $f_k^{(ij)}$ with respect to *all* parameters.

Complexity

Computing, $\partial_{\boldsymbol{i}} f(x; \boldsymbol{a})$ for all $\boldsymbol{i}$ within the kth core requires

$$\mathcal{O}\left(\left(G(n) + n\right) r_{k-1} r_k\right)$$

operations.

FT evaluation and gradient algorithm
Forward sweep

Sandia National Laboratories

$$f(x_1, x_2, \ldots, x_d) = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\cdots \begin{bmatrix} f_k^{(11)}(x_k) & \cdots & f_k^{(1r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1}1)}(x_k) & \cdots & f_k^{(r_{k-1}r_k)}(x_k) \end{bmatrix} \cdots \mathcal{F}_d(x_d)$$

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage 1	Eval	$\mathbf{\Gamma} = \mathbf{F}_1 = \mathcal{F}(x_1)$	$E(n)r_1$	r_1
	Eval all	$\partial f_{1ij\ell} = \frac{\partial f_1^{(ij)}}{\partial \boldsymbol{a}_{1ij\ell}}$	$G(n)r_1$	nr_1

FT evaluation and gradient algorithm
Forward sweep

Sandia National Laboratories

$$f(x_1, x_2, \ldots, x_d) = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\cdots \begin{bmatrix} f_k^{(11)}(x_k) & \cdots & f_k^{(1r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1}1)}(x_k) & \cdots & f_k^{(r_{k-1}r_k)}(x_k) \end{bmatrix} \cdots \mathcal{F}_d(x_d)$$

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage 1	Eval	$\mathbf{\Gamma} = \mathbf{F}_1 = \mathcal{F}(x_1)$	$E(n)r_1$	r_1
	Eval all	$\partial f_{1ij\ell} = \frac{\partial f_1^{(ij)}}{\partial \boldsymbol{a}_{1ij\ell}}$	$G(n)r_1$	nr_1
Stage 2	Eval	$\mathbf{F}_2 = \mathcal{F}(x_2)$	$E(n)r_1r_2$	r_1r_2
	Up. part. deriv	$\partial f_{2ij\ell} = \mathbf{\Gamma}[i]\frac{\partial f_2^{(ij)}}{\partial \boldsymbol{a}_{2ij\ell}}$	$G(n)r_1r_2$	nr_1r_2
	Update	$\mathbf{\Gamma} \leftarrow \mathbf{\Gamma}\mathbf{F}_2$	r_1r_2	-

FT evaluation and gradient algorithm
Forward sweep

Sandia National Laboratories

$$f(x_1, x_2, \ldots, x_d) = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\cdots \begin{bmatrix} f_k^{(11)}(x_k) & \cdots & f_k^{(1r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1}1)}(x_k) & \cdots & f_k^{(r_{k-1}r_k)}(x_k) \end{bmatrix} \cdots \mathcal{F}_d(x_d)$$

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage 1	Eval	$\mathbf{\Gamma} = \mathbf{F}_1 = \mathcal{F}(x_1)$	$E(n)r_1$	r_1
	Eval all	$\partial f_{1ij\ell} = \frac{\partial f_1^{(ij)}}{\partial \boldsymbol{a}_{1ij\ell}}$	$G(n)r_1$	nr_1
Stage 2	Eval	$\mathbf{F}_2 = \mathcal{F}(x_2)$	$E(n)r_1r_2$	r_1r_2
	Up. part. deriv	$\partial f_{2ij\ell} = \mathbf{\Gamma}[i]\frac{\partial f_2^{(ij)}}{\partial \boldsymbol{a}_{2ij\ell}}$	$G(n)r_1r_2$	nr_1r_2
	Update	$\mathbf{\Gamma} \leftarrow \mathbf{\Gamma}\mathbf{F}_2$	r_1r_2	-
Stage k	Eval	$\mathbf{F}_k = \mathcal{F}(x_k)$	$E(n)r_{k-1}r_k$	$r_{k-1}r_k$
	Up. part. deriv	$\partial f_{kij\ell} = \mathbf{\Gamma}[i]\frac{\partial f_k^{(ij)}}{\partial \boldsymbol{a}_{kij\ell}}$	$G(n)r_{k-1}r_k$	$nr_{k-1}r_k$
	Update	$\mathbf{\Gamma} \leftarrow \mathbf{\Gamma}\mathbf{F}_k$	$r_{k-1}r_k$	-

FT evaluation and gradient algorithm
Forward sweep

Sandia National Laboratories

$$f(x_1, x_2, \ldots, x_d) = \mathcal{F}_1(x_1)\mathcal{F}_2(x_2)\cdots \begin{bmatrix} f_k^{(11)}(x_k) & \cdots & f_k^{(1r_k)}(x_k) \\ \vdots & \ddots & \vdots \\ f_k^{(r_{k-1}1)}(x_k) & \cdots & f_k^{(r_{k-1}r_k)}(x_k) \end{bmatrix} \cdots \mathcal{F}_d(x_d)$$

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage 1	Eval	$\mathbf{\Gamma} = \mathbf{F}_1 = \mathcal{F}(x_1)$	$E(n)r_1$	r_1
	Eval all	$\partial f_{1ij\ell} = \frac{\partial f_1^{(ij)}}{\partial \boldsymbol{a}_{1ij\ell}}$	$G(n)r_1$	nr_1
Stage 2	Eval	$\mathbf{F}_2 = \mathcal{F}(x_2)$	$E(n)r_1r_2$	r_1r_2
	Up. part. deriv	$\partial f_{2ij\ell} = \mathbf{\Gamma}[i]\frac{\partial f_2^{(ij)}}{\partial \boldsymbol{a}_{2ij\ell}}$	$G(n)r_1r_2$	nr_1r_2
	Update	$\mathbf{\Gamma} \leftarrow \mathbf{\Gamma}\mathbf{F}_2$	r_1r_2	-
Stage k	Eval	$\mathbf{F}_k = \mathcal{F}(x_k)$	$E(n)r_{k-1}r_k$	$r_{k-1}r_k$
	Up. part. deriv	$\partial f_{kij\ell} = \mathbf{\Gamma}[i]\frac{\partial f_k^{(ij)}}{\partial \boldsymbol{a}_{kij\ell}}$	$G(n)r_{k-1}r_k$	$nr_{k-1}r_k$
	Update	$\mathbf{\Gamma} \leftarrow \mathbf{\Gamma}\mathbf{F}_k$	$r_{k-1}r_k$	-
Stage d	Eval	$\mathbf{F}_d = \mathcal{F}(x_d)$	$E(n)r_{d-1}$	r_{d-1}
	Part. deriv	$\partial f_{dij\ell} = \mathbf{\Gamma}[i]\frac{\partial f_d^{(ij)}}{\partial \boldsymbol{a}_{dij\ell}}$	$G(n)r_{d-1}$	nr_{d-1}
	Final evaluation	$f(x) = \mathbf{\Gamma}\mathbf{F}_d$	r_{d-1}	-

FT evaluation and gradient algorithm
Backward sweep

Sandia National Laboratories

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage $d-1$	Initialize	$\boldsymbol{\Theta} = \mathbf{F}_d$	-	-
	Part deriv	$\partial f_{(d-1)ij\ell} = \partial f_{(d-1)ij\ell}\boldsymbol{\Theta}[j]$	r_{d-1}	-

FT evaluation and gradient algorithm
Backward sweep

Sandia National Laboratories

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage $d-1$	Initialize	$\boldsymbol{\Theta} = \mathbf{F}_d$	-	-
	Part deriv	$\partial f_{(d-1)ij\ell} = \partial f_{(d-1)ij\ell}\boldsymbol{\Theta}[j]$	r_{d-1}	-
Stage k	Up. right product	$\boldsymbol{\Theta} \leftarrow \mathbf{F}_{k+1}\boldsymbol{\Theta}$	$r_k r_{k+1}$	-
	Up. part. deriv	$\partial f_{kij\ell} = \partial f_{kij\ell}\boldsymbol{\Theta}[j]$	$r_{k-1} r_k$	-

FT evaluation and gradient algorithm
Backward sweep

Sandia National Laboratories

Stage	Task	Operation	Alg. Complexity	Storage Complexity
Stage $d-1$	Initialize	$\boldsymbol{\Theta} = \mathbf{F}_d$	-	-
	Part deriv	$\partial f_{(d-1)ij\ell} = \partial f_{(d-1)ij\ell}\boldsymbol{\Theta}[j]$	r_{d-1}	-
Stage k	Up. right product	$\boldsymbol{\Theta} \leftarrow \mathbf{F}_{k+1}\boldsymbol{\Theta}$	$r_k r_{k+1}$	-
	Up. part. deriv	$\partial f_{kij\ell} = \partial f_{kij\ell}\boldsymbol{\Theta}[j]$	$r_{k-1} r_k$	-
Stage 1	Up. part. deriv	$\partial f_{1ij\ell} = \partial f_{1ij\ell}\boldsymbol{\Theta}[j]$	r_1	-

Alg. complexity: $\mathcal{O}\left(dr^2\left(G(n)+E(n)\right)\right)$

Real data

Sandia National Laboratories

- Kernel expansion for each univariate function
- CV over number of basis functions
- Compare with other parametric approaches
- Data and results for non-FT algs from Kandasamy and Yu (2016).

Data Set (d)	FT	RR	LASSO	LAR
Housing (12)	**0.416836**	95.60708	0.44515	0.84410
Galaxy (20)	**0.002675**	0.13902	0.02392	1.02315
Skillcraft (18)	**0.556605**	0.70910	0.66496	1.0048
CCPP (59)	**0.073086**	0.07641	0.07395	1.04527
Airfoil (41)	**0.42956**	0.53187	0.51986	1.00910
Speech (21)	**0.02571**	0.07392	0.07303	0.73916

Use compression within dynamic programming

Sandia National Laboratories

- Value iteration – fixed point interation
  - Iterate the Bellman operator:

$$T_h(J) = \min_{u_h(x) \in \mathcal{U}^h} \left[g(x, u_h(x)) + \gamma \sum_{y \in \mathcal{X}^h} p(x, y | u_h(x)) J(y) \right].$$

$$J^{k+1} = T_h(J^{k+1})$$

- Policy iteration – generate a sequence of policies
  1. Update policy

$$\mu_{k+1} = T_h(J_{\mu_k})$$

  2. Update Cost: Solve

$$J_{\mu_{k+1}} = T_h(J_{\mu_{k+1}})$$

- Multigrid
  - Observation: coarse grid solutions converge quickly
  - Algorithm: Solve problem on sequentially finer grids

Convergence result for the approximation

Sandia National Laboratories

Theorem

Let J^{k+1} denote a sequence of functions obtained through the iteration

$$J^{(k+1)} = \texttt{FT-cross}\left(T_h(J^{(k+1)}), \epsilon\right).$$

If $T_h(J)$ is a contraction mapping and the sequence $J^{(k+1)}$ has bounded ranks then FT-based VI convergences according to

$$\lim_{k\to\infty} \|J^{(k)} - J_h^*\| \leq \frac{\varepsilon\rho}{1-\gamma},$$

- Convergence is in the norm for which $T_h(J)$ is a contraction mapping
- Bounded ranks ensures that the cross-approximation algorithm can obtain an approximation to a tolerance of ϵ

Underactuated perching glider
Cory 2008

Sandia National Laboratories

$$J^*(z) = \min_{u(t)} \mathbb{E}\Big[\int_0^T \bar{x}^T \mathbf{Q}\bar{x} + 0.1u^2 dt + \bar{x}(T)^T \mathbf{Q}_f \bar{x}(T)\Big]$$

subject to nonlinear ODEs

$$\begin{aligned} m\ddot{x} &= -f_w s_\theta - f_e s_{\theta+\phi} + \sigma dw \\ m\ddot{y} &= f_w c_\theta + f_e c_{\theta+\phi} - mg + \sigma dw \\ I\ddot{\theta} &= -f_w l_w - f_e(l c_\phi + l_e) + \sigma dw \end{aligned}$$

$$\mathbf{x}_w = [x - l_w c_\theta, y - l_w s_\theta],$$

$$\mathbf{x}_w = [\dot{x} + l_w \dot{\theta} s_\theta, \dot{y} - l_w \dot{\theta} c_\theta]$$

$$\mathbf{x}_e = [x - l c_\theta - l_e, c_{\theta+\phi}, y - l s_\theta - l_e s_{\theta+\phi}]$$

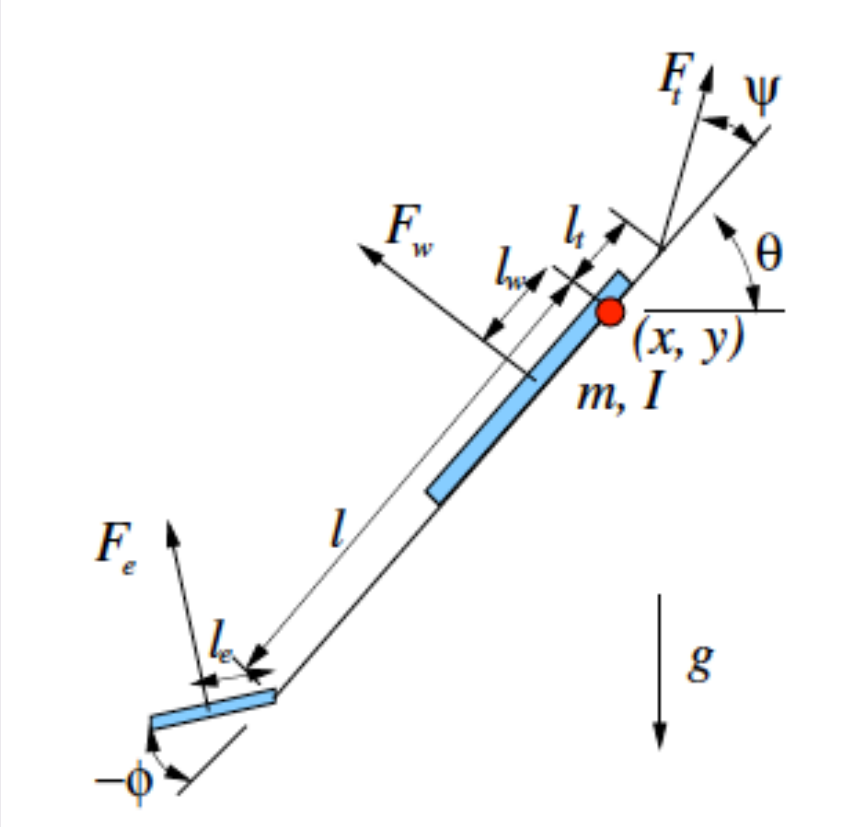
$$\dot{\mathbf{x}}_e = [\dot{x} + l\dot{\theta} s_\theta + l_e(\dot{\theta} + u) s_{\theta+\phi}, \dot{y} - l\dot{\theta} c_\theta - l_e(\dot{\theta} + u) c_{\theta+\phi}]$$

$$\alpha_w = \theta - \tan^{-1}(\dot{y}_w, \dot{x}_w),$$

$$\alpha_e = \theta + \phi - \tan^{-1}(\dot{y}_e, \dot{x}_e)$$

$$f_w = \rho S_w |\dot{\mathbf{x}}_w|^2 \sin(\alpha_w),$$

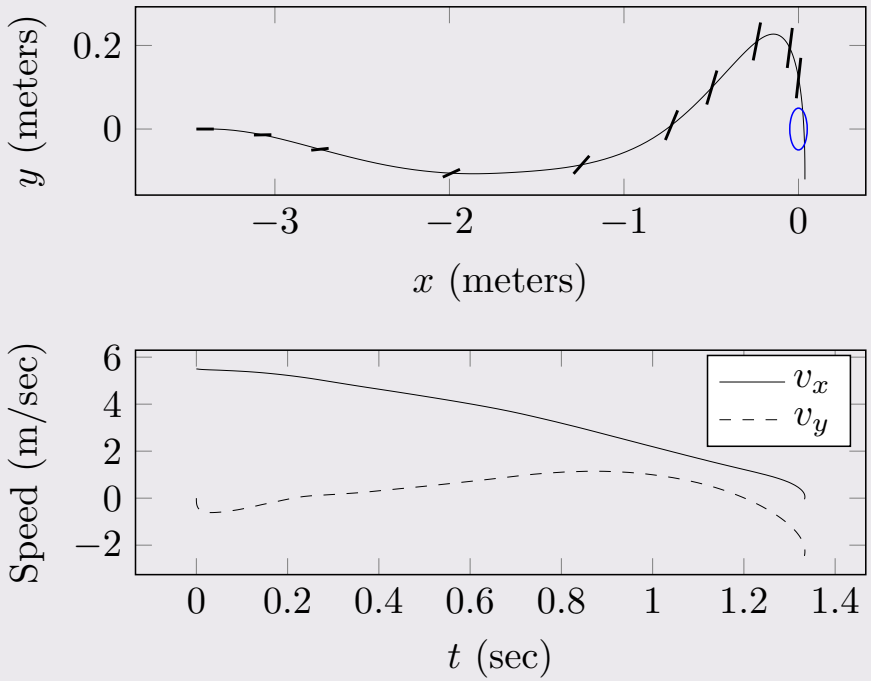
$$f_e = \rho S_e |\dot{\mathbf{x}}_e|^2 \sin(\alpha_e)$$

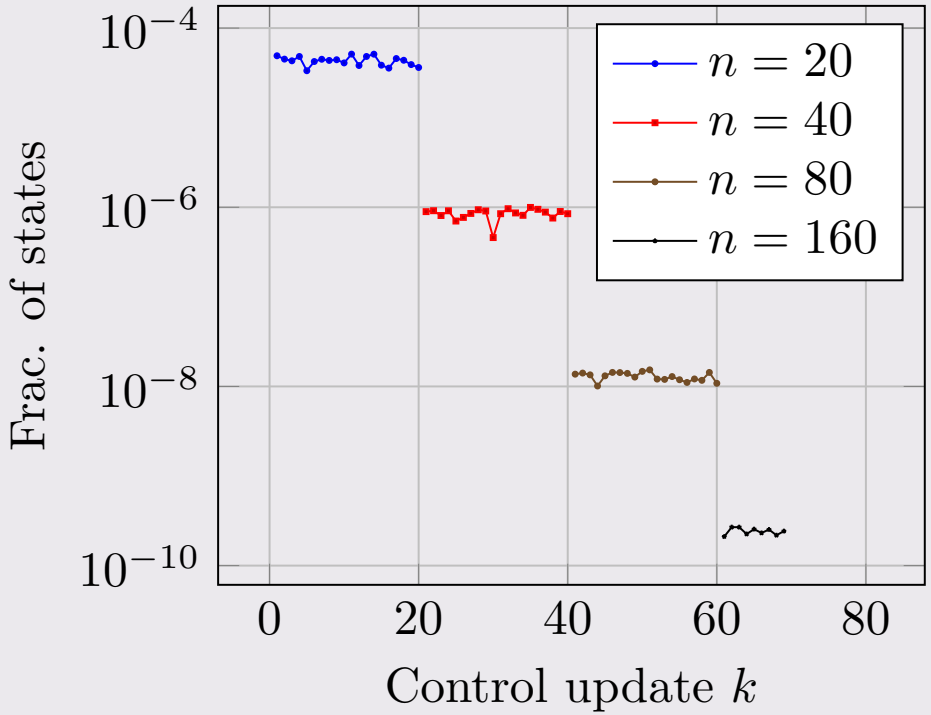


Simulation and compression

Sandia National Laboratories

- One-way multigrid [Chow and Tsitsiklis 1991]

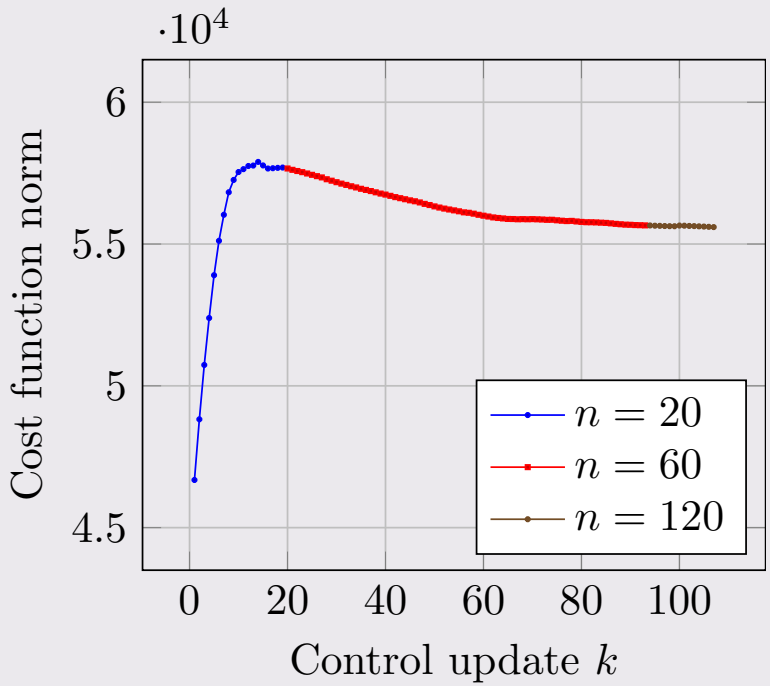


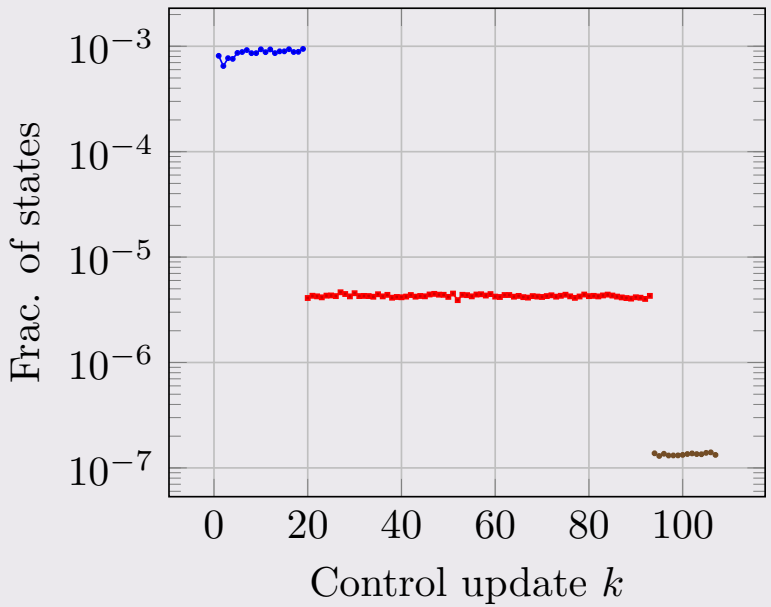


Experimental quadcopter

Sandia National Laboratories

- Goal: maneuver a quadcopter quickly through a window
- Dynamical system has six states and three controls, non-affine control
- Motion capture cameras for state estimation
- Optimal cost function computed offline (1 MB), full controller would require 24 TB





Conclusions

Sandia National Laboratories

Summary:

- Tensor-based compression mitigates curse of dimensionality in stochastic optimal control problems
- Works with nonlinear and non-affine-control systems
- Up to four orders of magnitude reduction
- Extensions to real-time systems and state estimation

Related papers: (available on `alexgorodetsky.com` and Arxiv)

- Gorodetsky A., Karaman, S., and Marzouk Y. M. Function-train: a continuous analogue of the tensor-train decomposition. (2016)
- Gorodetsky A., Karaman, S., and Marzouk Y. M. High-dimensional stochastic optimal control using continuous tensor decompositions. (2016)

Code:

- Compressed Continuous Computation (C^3) Library: `github.com/goroda/Compressed-continuous-computation`
- Compressed stochastic optimal control (C^3SC): `github.com/goroda/c3sc`

Acknowledgements

Collaborators:

- John Alora
- John Jakeman
- Sertac Karaman
- Youssef Marzouk
- Ezra Tal

I am currently funded by the John von Neumann Postdoctoral Research Fellowship supported by the DOE Applied Mathematics program in the Office of Advanced Scientific Computing Research.

Backup Slides

Advantages of the FT format

Sandia National Laboratories

- Nonlinear parameterizations: can capture local effects
- Sparse representations:
  - Example: Rank-2 additive function

$$f(x_1, \ldots, x_d) = f_1(x_1) + \cdots + f_d(x_d)$$

  - FT format:

$$f(x_1, x_2, \ldots, x_d) = [f_1(x_1)\ 1] \begin{bmatrix} 1 & 0 \\ f_2(x_2) & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 \\ f_d(x_d) \end{bmatrix}$$

  - TT storage requirement: $4p(d-2)$ floats
  - FT storage requirement: $d(p+3)-4$ floats
  - In the limit, FT requires almost 4 times less storage
  - Difference is more striking for higher order interactions

MAXVOL algorithm

Sandia National Laboratories

1. Find $\boldsymbol{x} = [x_1, x_2]$ such that cross matrix is nonsingular
2. Form $B(x) = F(x)\mathbf{G}$
3. Find $i^*, x^* = \arg \max_{i \in [1,\ldots,r], x} [b_i(x)]$
   - If basis functions of b_i are orthonormal polynomials then solve eigenvalue problem!
   - If basis functions of b_i are piecewise polynomials then find extremum of each piece
   - If some other representation, can use brute force and discretize to find maximum
4. $b_{\text{max}} = b_{i^*}(x^*)$
5. If $|b_{\text{max}}| \geq 1 \rightarrow \boldsymbol{x}[i^*] = x^*$ (switch "rows")
6. Go to step 2.

Rank adaptation

- Use FT rounding and CV
- Increase ranks until either
  - Rounding lowers all ranks
    $\rightarrow$ data not informative enough
  - CV error increases
    $\rightarrow$ avoid overfitting
- Rounding threshold is a parameter
  - Similar to regularization
  - Relation to other approaches?