Low-rank cross approximation approach for reducing stochastic collocation models

Sergey Dolgov

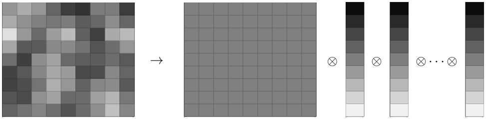
joint work with Robert Scheichl

UNIVERSITY OF BATH

Model Order Reduction Workshop
Durham, August 12, 2017

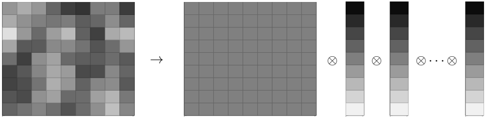
Stochastic partial differential equation

- Uncertainty quantification (UQ)
  - Subsurface flow
  - Calibration
  - Fluid dynamics



Stochastic partial differential equation

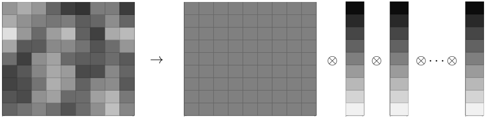
- Uncertainty quantification (UQ)
  - Subsurface flow
  - Calibration
  - Fluid dynamics



- Example: $-\nabla_{\mathbf{x}} \kappa(\mathbf{x}, \theta_1, \dots, \theta_d) \nabla_{\mathbf{x}} u = f$

Stochastic partial differential equation

- Uncertainty quantification (UQ)
  - Subsurface flow
  - Calibration
  - Fluid dynamics



- Example: $-\nabla_{\mathbf{x}}\kappa(\mathbf{x}, \theta_1, \ldots, \theta_d)\nabla_{\mathbf{x}} u = f$

- Many uncertain quantities $\rightarrow$ many dimensions
- Discretize: n DOFs for each θ_k $\rightarrow$ n^d elements in total.

High-dimensional problem $\rightarrow$ low-dimensional output

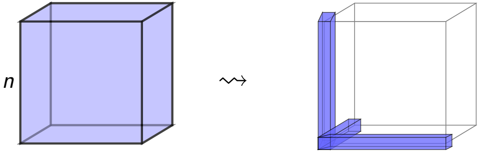
However, the output of interest is **low**-dimensional.

- Eliminate space $Q(\boldsymbol{\theta}) = \mathcal{Q}\left[u(x, \boldsymbol{\theta})\right]$.
- Eliminate parameters by computing
  - moments $\mathbb{E}Q^p$,
  - distribution function/event probabilities $P(Q < \xi)$,
  - quantiles $\xi : P(Q < \xi) > 0.95$

Approximate the solution by a **low-dimensional representation**.

Mathematical insights for data compression

Low-rank tensor decomposition $\Leftrightarrow$ separation of variables:



- Approximate: $\underbrace{u(i_1,\ldots,i_d)}_{\text{tensor}} \approx \underbrace{\sum_{\alpha} u_{\alpha}^{(1)}(i_1)u_{\alpha}^{(2)}(i_2)\cdots u_{\alpha}^{(d)}(i_d)}_{\text{tensor product decomposition}}.$

Goals:

- Store and integrate u $\qquad \mathcal{O}(dn)$ cost **instead of $\mathcal{O}(n^d)$.**
- Solve equations $Au = f$ $\qquad \mathcal{O}(dn^2)$ cost **instead of $\mathcal{O}(n^{2d})$**.

Stochastic PDEs: solution methods

	Cost vs. accuracy	Use of structure
Monte Carlo/Quasi MC	$-$	$+$
Sparse grids (collocation)	$\pm$	$+$
Low-rank decompositions	$+$	$-$

Stochastic PDEs: solution methods

	Cost vs. accuracy	Use of structure
Monte Carlo/Quasi MC	$-$	$+$
Sparse grids (collocation)	$\pm$	$+$
Low-rank decompositions	$+$	?

?

- Stochastic PDE $\rightarrow$ **block-diagonal** linear system.
- Generic low-rank algorithms discard the sparsity.
- Can we fix that?

Tensor decompositions: the two workhorses

Goals:

- Store and integrate u

  How to construct u directly?

- Solve equations $Au = f$

  In general?

  How to leverage sparsity?

Tensor decompositions: the two workhorses

Goals:

- Store and integrate u

  How to construct u directly?

Alternating Least Squares

Cross interpolation

- Solve equations $Au = f$

  In general?

  How to leverage sparsity?

my talk

Tensor decompositions: the two workhorses

Goals:

- Store and integrate u

  How to construct u directly?

Alternating Least Squares
Cross interpolation

- Solve equations $Au = f$

  In general?

my talk

  How to leverage sparsity?

The story starts in two dimensions...

2D: low-rank matrices

- Discrete **Separation of variables**:

$$\begin{bmatrix} u_{1,1} & \cdots & u_{1,n} \\ \vdots & & \vdots \\ u_{n,1} & \cdots & u_{n,n} \end{bmatrix} = \sum_{\alpha=1}^{r} \begin{bmatrix} v_{1,\alpha} \\ \vdots \\ v_{n,\alpha} \end{bmatrix} \begin{bmatrix} w_{\alpha,1} & \cdots & w_{\alpha,n} \end{bmatrix} + \mathcal{O}(\varepsilon).$$

- **Rank** $r \ll n$.
- $\mathtt{mem}(v) + \mathtt{mem}(w) = 2nr \ll n^2 = \mathtt{mem}(u)$.
- **Singular Value Decomposition** $\rightarrow$ optimal approximation:

$$\|U - VW^*\|_F^2 \rightarrow \min_{V,W}$$

Cross approximation methods

Singular Value Decomposition is not always good:

- impossible to start from full arrays
- analytical low-rank forms may not exist

Cross algorithms: reconstruct a low-rank form from **a few entries**.

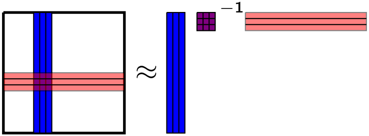
Cross interpolation

- Recall SVD: minimization of the error $\|U - VW^*\|_F^2$.
- **Interpolate** instead:

$$U(\mathcal{I},:) = V(\mathcal{I},:)W^*, \qquad U(:,\mathcal{J}) = VW^*(:,\mathcal{J})$$

for some **index sets** $\mathcal{I},\mathcal{J} \subset \{1,\ldots,n\}$.

- Equivalent to **cross** decomposition:



How to **find** index sets?

Cross approximation: alternating iteration

Practically realizable strategy: assume **initial guess** $U \approx VW^\top$.

1. $\mathcal{J} = \texttt{pivots}(W) \quad \rightarrow \quad V = U(:, \mathcal{J})$.
2. $\mathcal{I} = \texttt{pivots}(V) \quad \rightarrow \quad W = U(\mathcal{I}, :)$.
3. repeat...

V, W are $n \times r$ matrices $\Rightarrow$ `pivots` are feasible (LU, **Maxvol**[1])

Cost: $2nr$ samples $+ \mathcal{O}(nr^2)$ other flops per iteration.

[1] Goreinov, Tyrtyshnikov '01

Cross approximation algorithm

Improvements:

- $V = \mathrm{qr}(V)$ numerical stability.
- $V = \begin{bmatrix} V & Z \end{bmatrix}$ rank update.

Use cross approximation to construct **low-rank PDE coefficients**.

Similar algorithms exist: ACA[2], (D)EIM[3].

---

[2] [Bebendorf]

[3] [Maday, Chaturantabut/Sorensen] This week: ask Chris?

Stochastic PDE $\to$ block-diagonal matrix

- We are solving $-\nabla\kappa(x,\theta)\nabla u = f$
- n Finite Elements for x, m collocation points for θ.

$$\int \kappa(x,\theta_j)\nabla\psi_i(x)\cdot\nabla u(x,\theta_j)\;dx = \int \psi_i(x) f(x,\theta_j) dx$$

Independent equations over x for different θ.

$$\begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_m \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}$$

But: every block A_j is **not** diagonal.

Cross approximation and Alternating Least Squares (ALS)

Cross approximation algorithm:

+ Good for functions defined pointwise.
− Not applicable for linear systems.

Cross approximation and Alternating Least Squares (ALS)

Cross approximation algorithm:

+ Good for functions defined pointwise.

? Not applicable for linear systems.

Cross interpolation for linear systems

- Rewrite interpolation as projection:

$$E_{\mathcal{J}}^{\top}\mathrm{vec}(VW^*) = E_{\mathcal{J}}^{\top}\mathrm{vec}(U),$$

where $E_{\mathcal{J}}$ is a **submatrix of identity** at $\mathcal{J}$.

Cross interpolation for linear systems

- Rewrite interpolation as projection:

$$E_{\mathcal{J}}^{\top} \cdot I \cdot \mathrm{vec}(VW^*) = E_{\mathcal{J}}^{\top} \cdot I \cdot \mathrm{vec}(U),$$

where $E_{\mathcal{J}}$ is a **submatrix of identity** at $\mathcal{J}$.

Cross interpolation for linear systems

- Rewrite interpolation as projection:

$$E_{\mathcal{J}}^{\top} \cdot I \cdot \mathrm{vec}(VW^*) = E_{\mathcal{J}}^{\top} \cdot I \cdot \mathrm{vec}(U),$$

where $E_{\mathcal{J}}$ is a **submatrix of identity** at $\mathcal{J}$.

- Replace I by the stiffness matrix:

$$E_{\mathcal{J}}^{\top} \cdot \mathbf{A} \cdot \mathrm{vec}(VW^*) = E_{\mathcal{J}}^{\top} \cdot \mathbf{A} \cdot \mathrm{vec}(U)$$

Cross interpolation for linear systems

- Rewrite interpolation as projection:

$$E_{\mathcal{J}}^{\top} \cdot I \cdot \mathrm{vec}(VW^*) = E_{\mathcal{J}}^{\top} \cdot I \cdot \mathrm{vec}(U),$$

where $E_{\mathcal{J}}$ is a **submatrix of identity** at $\mathcal{J}$.

- Replace I by the stiffness matrix:

$$E_{\mathcal{J}}^{\top} \cdot \mathbf{A} \cdot \mathrm{vec}(VW^*) = E_{\mathcal{J}}^{\top} \mathrm{vec}(F)$$

... and $\mathbf{A} \cdot \mathrm{vec}(U)$ by the right hand side.

Cross interpolation for linear systems

- Rewrite interpolation as projection:

$$E_{\mathcal{J}}^{\top} \cdot I \cdot \mathrm{vec}(VW^*) = E_{\mathcal{J}}^{\top} \cdot I \cdot \mathrm{vec}(U),$$

where $E_{\mathcal{J}}$ is a **submatrix of identity** at $\mathcal{J}$.

- Replace I by the stiffness matrix:

$$E_{\mathcal{J}}^{\top} \cdot \mathbf{A} \cdot \mathrm{vec}(VW^*) = E_{\mathcal{J}}^{\top} \mathrm{vec}(F)$$

- Still: any benefit for UQ?

How we represent the matrix?

- **Coefficient** is low-rank:

$$\kappa(x, \theta) \approx \sum_{\beta=1}^{R} g_\beta(x) h_\beta(\theta).$$

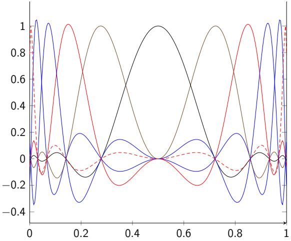
- Hence $\mathbf{A}$ is low-**Kronecker**-rank:

$$\mathbf{A} = \sum_{\beta=1}^{R} A_\beta \otimes D_\beta$$

- A_β is FEM-related, but
- $D_\beta = \mathrm{diag}(d_\beta)$ is **diagonal**.

Interpolatory representation

- VW^* is not unique $\to$ ensure $W(\mathcal{J}) = I$.



...as a by-product $V = U(:, \mathcal{J})$.

- **Distribute** the products

$$E_{\mathcal{J}}^{\top} \cdot \mathbf{A} \cdot \mathrm{vec}(VW^*) = \sum_{\beta=1}^{R} A_\beta \otimes [D_\beta(\mathcal{J}, :)W] \cdot v = \sum_{\beta=1}^{R} A_\beta \otimes \mathrm{diag}(d_\beta(\mathcal{J})) \cdot v$$

Block diagonal system, stage 1: space

The first step:

$$\{j_1, \dots, j_r\} = \texttt{pivots}(W)$$

$$\begin{bmatrix} A_{j_1} & & \\ & A_{j_2} & \\ & & A_{j_r} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_r \end{bmatrix} = \begin{bmatrix} f_{j_1} \\ f_{j_2} \\ f_{j_r} \end{bmatrix}$$

- Solve r independent deterministic problems.
  - Similar to Monte Carlo and stochastic collocation.
  - Can use specialized tools (preconditioners/software).

Block diagonal system, stage 2: parameters

A_j is not diagonal $\rightarrow$ ALS-projection:

1. Make V orthogonal $\rightarrow$ projection matrix $\mathcal{V} = V \otimes I$.
2. Solve $\left(\mathcal{V}^\top \mathbf{A} \mathcal{V}\right) w = \mathcal{V}^\top \mathbf{f}$.

- Solve m systems of size $r \times r$.
- Similar to Reduced Basis MOR (**in 2 dimensions**).
- Extensible to **many dimensions**.

Back to many variables

- What about $-\nabla\kappa(x, \theta_1, \dots, \theta_d)\nabla u = f$?

Tensor Train (TT) decomposition

- Many dimensions: Matrix Product States/Tensor Train[4]:

$$u(i_1 \dots i_d) = \sum_{\alpha_k=1}^{r_k} u^{(1)}_{\alpha_1}(i_1) \cdot u^{(2)}_{\alpha_1,\alpha_2}(i_2) \cdot u^{(3)}_{\alpha_2,\alpha_3}(i_3) \cdots u^{(d)}_{\alpha_{d-1}}(i_d)$$

- TT blocks $u^{(k)}$ are **three-dimensional**.
- Storage: $\mathcal{O}(dnr^2)$.

[4] Wilson '75, White '93, Verstraete '04, Oseledets '09

Tensor Train for stochastic PDEs

TT format of the coefficient $\rightarrow$ TT format of the matrix:

- Space $\rightarrow$ first TT block $\rightarrow$ FEM stiffness pattern
- Parameters $\rightarrow$ other TT blocks $\rightarrow$ diagonal

$$\mathbf{A} = \sum_{\beta_k=1}^{R_k} A^{(1)}_{\beta_1} \otimes \mathrm{diag}(d^{(2)}_{\beta_1,\beta_2}) \otimes \cdots \otimes \mathrm{diag}(d^{(d)}_{\beta_{d-1}})$$

Tensor Train for stochastic PDEs

The algorithm in a nutshell:

1. $\mathcal{J}_k = \texttt{pivots}\left(U_{>k}\right)$.
2. Generate and solve $\left[\mathcal{U}_{<k}{}^{\top} \mathbf{A}_{\mathcal{J}_k} \mathcal{U}_{<k}\right] u^{(k)} = \mathcal{U}_{<k}^{\top} \mathbf{f}_{\mathcal{J}_k}$.
3. $\mathcal{U}_{<k+1} = \mathcal{U}_{<k} \cdot u^{(k)}$.
4. Set $k = k+1$ or $k = k-1$ and repeat...

$$\xrightarrow{\qquad\qquad}$$

$$u^{(1)} \cdot u^{(2)} \ldots u^{(k)} \ldots u^{(d-1)} \cdot u^{(d)}$$

$$\xleftarrow{\qquad\qquad}$$

1. Problem setting
2. Low-rank UQ algorithms
   - Singular value decomposition
   - Cross approximation for matrices
   - Cross approximation for sPDEs
3. Numerical experiments

Log-normal diffusion coefficient

$$
\begin{aligned}
&-\nabla\kappa(x,\theta)\nabla u = 0 \quad \text{in} \quad (0,1)^2 \\
&u|_{x_1=0} = 1, \qquad u|_{x_1=1} = 0, \\
&\frac{\partial u}{\partial n}|_{x_2=0} = \frac{\partial u}{\partial n}|_{x_2=1} = 0.
\end{aligned}
$$

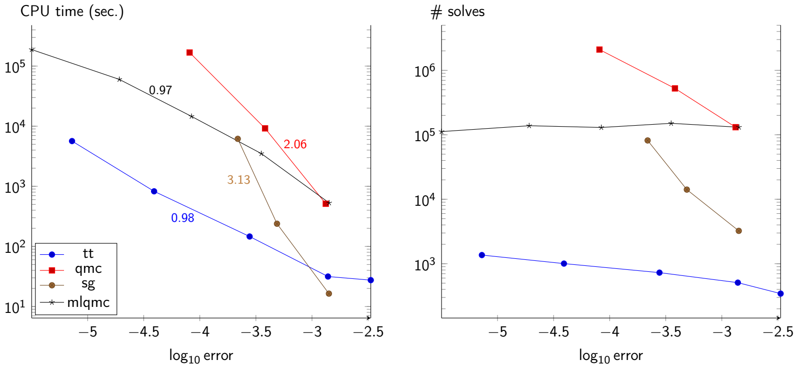
- $\kappa(x,\theta) = \exp\left(\sum_{k=1}^{d} \phi_k(x)\theta_k\right)$
- ϕ_k: Karhunen-Loeve expansion of the Matern covariance with parameters $\sigma^2 = 1$ and (different) ν.

Methods

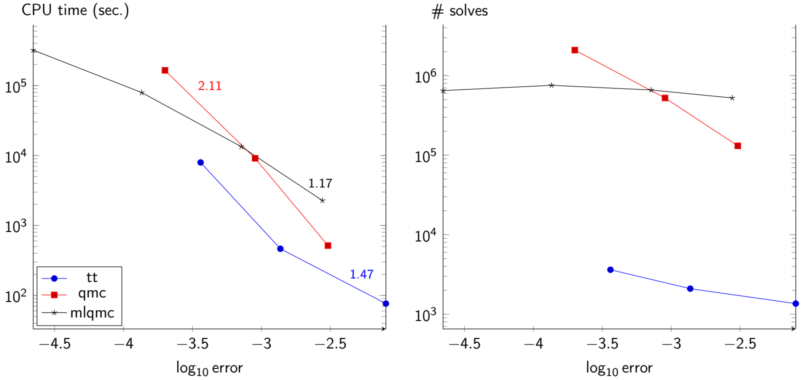
- Quasi Monte Carlo with a special lattice vector[5].
- Multilevel QMC.
- Adaptive Sparse Grids toolbox[6].
- TT Cross algorithm.

[5] `lattice-39102-1024-1048576.3600.txt` from F. Kuo
[6] Andreas Klimke '08 `http://www.ians.uni-stuttgart.de/spinterp/`

Smooth ($\nu = 4$) uniform field



Rougher ($\nu = 2.5$) normal field



Conclusion

Cross interpolation can be used as a low-rank solver...

- ...which preserves sparsity in sPDEs.
- Faster than QMC/SG if the problem has low-rank structure.
- Less efficient if the problem is "more" random.
- Reference and code: [arXiv:1707.04562]
- Future plans: inverse problems.

Conclusion

Cross interpolation can be used as a low-rank solver...

- ...which preserves sparsity in sPDEs.
- Faster than QMC/SG if the problem has low-rank structure.
- Less efficient if the problem is "more" random.

- Reference and code: [arXiv:1707.04562]

- Future plans: inverse problems.

Thank you for your attention!