# Accelerating Hierarchical Stochastic Collocation Methods for PDEs with Random Coefficients

Peter Jantsch

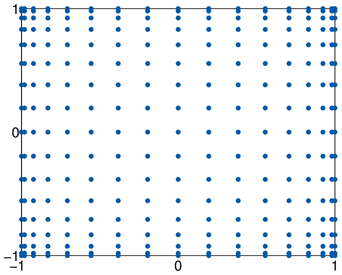
University of Tennessee, Knoxville

November 19, 2014

Joint work with **Diego Galindo** (ORNL), **Guannan Zhang** (ORNL), **Clayton Webster** (ORNL, UTK).

## Improving on existing methods

- Stochastic sampling methods for random PDEs are computationally expensive: each sample point corresponds to a PDE solve.
- We can try to improve single level methods by reducing the number of samples/solves: Quasi Monte Carlo and importance sampling, (anisotropic) sparse grids, adaptive grids.

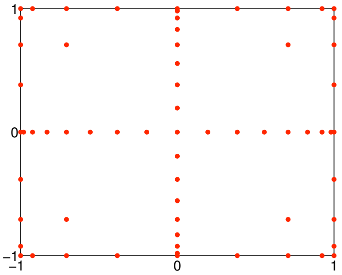


Tensor Product Grid

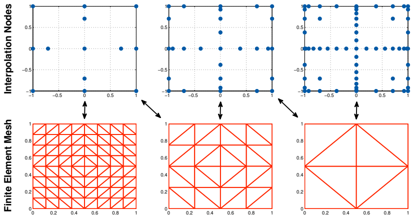
$\longleftarrow$

$\longrightarrow$

Sparse Grid

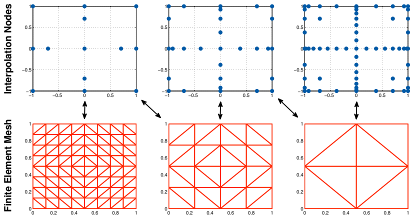


## Improving on existing methods

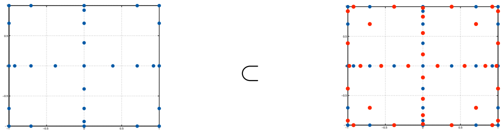


- Exploit the hierarchy in deterministic approximation: For a given accuracy, multilevel methods seek to reduce the complexity by spreading computational cost evenly across several resolutions of the spatial discretization

## Improving on existing methods



- Exploit the hierarchy in deterministic approximation: For a given accuracy, multilevel methods seek to reduce the complexity by spreading computational cost evenly across several resolutions of the spatial discretization
- **Exploit the hierarchy in stochastic approximation:** Sparse grids with nested grid points provide a natural multilevel hierarchy which we can use to accelerate each PDE solve



## Uncertainty Quantification

Uncertain input parameters: $\mathbf{y}(\omega), \ \omega \in \Omega_{\mathbb{P}}$ $\longrightarrow$ **SPDE model:** $\mathcal{L}(u, \mathbf{y}) = f$ for a.e. $x \in D \subset \mathbb{R}^d$ $\longrightarrow$ Quantity of interest: $Q[u(\cdot, \mathbf{y})]$

- The parameters $\mathbf{y}(\omega)$ may be affected by uncertainty (experimental data, incomplete description of parameters, unresolved scales, etc.)
- $\mathbf{y} : \Omega \to \Gamma \subset \mathbb{R}^N$ can be assumed to be a random vector with $N$ components, i.e., $\mathbf{y} = (y_1, \ldots, y_N)$, with joint probability density function $\rho(\mathbf{y})$

The solution $u$ is a stochastic function, $u(\cdot, \mathbf{y})$

**Goals of forward UQ**: Approximate $u$ or some statistical QoI depending on $u$, i.e.

$$\mathbb{E}[u], \ \mathbb{V}\mathrm{ar}[u], \ \mathbb{P}[u > u_0] = \mathbb{E}[\mathbb{1}_{\{u > u_0\}}]$$

with the minimal computational cost possible.

## Model Problem - Linear Elliptic SPDE

Let $(\Omega, \mathcal{F}, P)$ be a complete probability space. Find $u$ such that almost surely, i.e. for $P$-almost every $\omega \in \Omega$

$$\begin{cases} -\nabla \cdot (a(x,\omega) \cdot \nabla u(x,\omega)) & = f(x,\omega) \quad x \in D, \\ u(x,\omega) & = 0 \qquad\quad x \in \partial D. \end{cases} \tag{1}$$

We'll assume that $a, f$ are such that this problem has a unique solution represented in terms of $\mathbf{y} \in \Gamma \subset \mathbb{R}^N$, a finite dimensional random vector.

## Model Problem - Linear Elliptic SPDE

Let $(\Omega, \mathcal{F}, P)$ be a complete probability space. Find $u$ such that almost surely, i.e. for $P$-almost every $\omega \in \Omega$

$$\begin{cases} -\nabla \cdot (a(x,\omega) \cdot \nabla u(x,\omega)) = f(x,\omega) & x \in D, \\ u(x,\omega) = 0 & x \in \partial D. \end{cases} \tag{1}$$

We'll assume that $a, f$ are such that this problem has a unique solution represented in terms of $\mathbf{y} \in \Gamma \subset \mathbb{R}^N$, a finite dimensional random vector.

- Such a PDE might be used to model ground water flow or current through a random material.

## Model Problem - Linear Elliptic SPDE

Let $(\Omega, \mathcal{F}, P)$ be a complete probability space. Find $u$ such that almost surely, i.e. for $P$-almost every $\boldsymbol{\omega} \in \Omega$

$$\left\{ \begin{array}{rcll} -\nabla \cdot (a(x,\omega) \cdot \nabla u(x,\omega)) & = & f(x,\omega) & x \in D, \\ u(x,\omega) & = & 0 & x \in \partial D. \end{array} \right. \tag{1}$$

We'll assume that $a, f$ are such that this problem has a unique solution represented in terms of $\mathbf{y} \in \Gamma \subset \mathbb{R}^N$, a finite dimensional random vector.

- Such a PDE might be used to model ground water flow or current through a random material.

- The methods we'll talk about are not specific to this simple class of problems, but our methods extend, e.g., to non-linear PDEs, and more generally to random PDEs where the solution map $\mathbf{y} \mapsto u(\cdot, \mathbf{y})$ has some smoothness properties.

## Assumptions

- The random fields $a$ and $f$ depend on a finite dimensional random vector $\mathbf{y}(\omega) := [y_1(\omega), \dots, y_N(\omega)] : \Omega \to \mathbb{R}^N$.
- The image $\Gamma_n := y_n(\Omega)$ of $y_n$ is bounded for all $n \in \{1, \dots, N\}$.
- With $\Gamma = \prod_{n=1}^N \Gamma_n$, the random variables $\mathbf{y}$ have a joint probability density function $\rho(\mathbf{y}) = \prod_{n=1}^N \rho_n(y_n) \in L^\infty(\Gamma)$.
- The coefficient $a$ is uniformly elliptic in $\mathbf{y}$, i.e. there exist $0 < a_{\min}, a_{\max} < \infty$ such that almost surely

$$a_{\min} < a(x, \mathbf{y}) < a_{\max}$$

## Assumptions

- The random fields $a$ and $f$ depend on a finite dimensional random vector $\mathbf{y}(\omega) := [y_1(\omega), \dots, y_N(\omega)] : \Omega \to \mathbb{R}^N$.
- The image $\Gamma_n := y_n(\Omega)$ of $y_n$ is bounded for all $n \in \{1, \dots, N\}$.
- With $\Gamma = \prod_{n=1}^N \Gamma_n$, the random variables $\mathbf{y}$ have a joint probability density function $\rho(\mathbf{y}) = \prod_{n=1}^N \rho_n(y_n) \in L^\infty(\Gamma)$.
- The coefficient $a$ is uniformly elliptic in $\mathbf{y}$, i.e. there exist $0 < a_{\min}, a_{\max} < \infty$ such that almost surely

$$a_{\min} < a(x, \mathbf{y}) < a_{\max}$$

**Examples:**

- Piecewise random material: Let $\{D_n\}_{n=1}^N$ be a partition of $D$ then define $a(x, \mathbf{y}) = \sum_{n=1}^N \sigma_n y_n(\omega) \chi_{D_n}(x)$
- $\infty$-dimensional random field, e.g. expand $a(x, \mathbf{y}) = \sum_{n=1}^\infty \phi_n(x) \Psi_n(\mathbf{y})$ in a Karhunen-Loeve expansion and retain the first $N$ terms

## Implications

Under the Doob-Dynkin Lemma, the probability space $(\Omega, \mathcal{F}, P)$ is mapped to $(\Gamma, \mathcal{B}(\Gamma), \rho(\mathbf{y})d\mathbf{y})$, where $\mathcal{B}(\Gamma)$ is the Borel $\sigma$-algebra on $\Gamma$. Thus, the solution $u$ can be characterized in terms of the random vector $\mathbf{y}(\omega)$, i.e.

$$u(x, \omega) = u(x, \mathbf{y}(\omega)).$$

## Implications

Under the Doob-Dynkin Lemma, the probability space $(\Omega, \mathcal{F}, P)$ is mapped to $(\Gamma, \mathcal{B}(\Gamma), \rho(\mathbf{y})d\mathbf{y})$, where $\mathcal{B}(\Gamma)$ is the Borel $\sigma$-algebra on $\Gamma$. Thus, the solution $u$ can be characterized in terms of the random vector $\mathbf{y}(\omega)$, i.e.

$$u(x, \omega) = u(x, \mathbf{y}(\omega)).$$

**Weak formulation:** find $u \in L^2_\rho := L^2_\rho(\Gamma; H^1_0(D))$ s.t., $\forall v \in L^2_\rho$

$$\mathbb{E}\left[\int_D a(x, \mathbf{y}) \nabla u(x, \mathbf{y}) \cdot \nabla v(x, \mathbf{y})\, dx\right] = \mathbb{E}\left[\int_D f(x) \cdot v(x, \mathbf{y})\, dx\right]$$

**Lax-Milgram:** $\exists!\ u \in L^2_\rho$ s.t. $\|u\|_{L^2_\rho} \leq \frac{C_P}{a_{min}} \left(\int_D \mathbb{E}[f^2]\, dx\right)^{1/2}$

## Monte Carlo Method

Pure random sampling to approximate statistics of the solution:

$$\mathbb{E}(u) = \int_\Gamma u(x,\mathbf{y})\rho(\mathbf{y})\,d\mathbf{y} \approx \frac{1}{M}\sum_{j=1}^{M} u(x,\mathbf{y}_j)$$

- Most widely used in applications and high ($\sim 100$) dimensional problems
- **Pro:** Simple to implement, easily parallelizable, convergence rate $\mathcal{O}(M^{-1/2})$ is dimension independent, but...
- **Con:** Relatively slow if the solution has some smoothness wrt random parameters

## Analytic Regularity

**Theorem**: [Babuska et al '07,'10, Webster '07]

- Let $\Gamma_n^* = \prod_{\substack{j=1 \\ j\neq n}}^{N} \Gamma_j$, and let $\mathbf{y}_n^*$ denote an arbitrary element of $\Gamma_n^*$

$\exists$ constants $\lambda$ (independent of $n$), $\gamma_n \geq 0$ and regions $\Sigma_n \equiv \{z \in \mathbb{C},\ \ \mathrm{dist}(z, \Gamma_n) \leq \gamma_n\}$ in the complex plane for which

$$\max_{\mathbf{y}_n^* \in \Gamma_n^*} \max_{z \in \Sigma_n} \|\nabla u(\cdot, \mathbf{y}_n^*, z)\|_{L^2(D)} \leq \lambda.$$

That is, the solution $u(x, \mathbf{y}_n^*, y_n)$ admits an analytic extension $u(x, \mathbf{y}_n^*, z)$, $z \in \Sigma_n \subset \mathbb{C}$.

- The analyticity of the solution $u(x, \mathbf{y})$ w.r.t. each random direction $y_n$ suggests the use of multivariate **global** polynomial approximation. In cases with less regularity we might turn to the use of **local** basis functions.

Analytic polydisc regularity: [Cohen-DeVore-Schwab 2010, Chkifa-Cohen-DeVore-Schwab 2013, Tran-Trenchea-Webster 2013, Tran-Webster-Zhang 2014]

## Stochastic Polynomial Methods

- **Stochastic Galerkin:** projection technique, intrusive approach
- **Stochastic collocation:** interpolation technique, non-intrusive approach

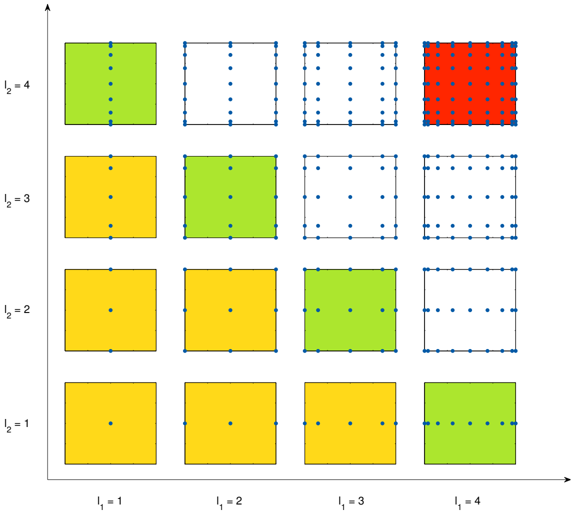
$$u_{h,M}^{SL}(x,\mathbf{y}) = \sum_{j=1}^{M} c_j(x)\psi_j(\mathbf{y})$$

**pro**: convergence can be faster than MC

**con**: curse of dimensionality

- $\psi_j$ is a linear combination of tensorized 1D global Lagrange nodal basis functions and $c_j$ are determined through $u_h(x,\mathbf{y}_j)$
- Note that the coefficients of the interpolant will be functions from some Sobolev space, i.e. $H_0^1(D)$
- Collocation bests Galerkin from a point of view of complexity, except in the simplest cases.

## Example: Level $L = 3$ Clenshaw-Curtis sparse grid



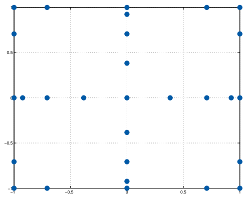
- 1D Growth Rule:

$$p(l_n) = 2^{l_n - 1} + 1, \; l > 1.$$

- Keep all grids satisfying:

$$g(\mathbf{l}) = \sum_{n=1}^{N} (l_n - 1) \leq 3.$$

$$\downarrow$$



Compare to the full tensor product grid

## Generalized Sparse Grid Interpolation

- Define $\{y_{n,j}^{(l)}\}_{j=1}^{p(l)} \subset \Gamma_n$ to be a set of $p(l)$ points in $\Gamma_n$.
- $\{\mathcal{U}_n^{p(l)}\}_{l\in\mathbb{N}} : C^0(\Gamma_n) \to \mathcal{P}_{p(l)-1}(\Gamma_n)$ is the standard one-dimensional Lagrange interpolation operator for $\{y_{n,j}^{(l)}\}_{j=1}^{p(l)} \subset \Gamma_n$.
- $\Delta_n^{p(l)} := \mathcal{U}_n^{p(l)} - \mathcal{U}_n^{p(l-1)}$.
- $g : \mathbb{N}_+^N \to \mathbb{N}$ is strictly increasing and defines the mapping between the multi-index $\mathbf{l}$ and the level $L$ used to construct the sparse grid.

## Generalized Sparse Grid Interpolation

- Define $\{y_{n,j}^{(l)}\}_{j=1}^{p(l)} \subset \Gamma_n$ to be a set of $p(l)$ points in $\Gamma_n$.
- $\{\mathcal{U}_n^{p(l)}\}_{l\in\mathbb{N}} : C^0(\Gamma_n) \to \mathcal{P}_{p(l)-1}(\Gamma_n)$ is the standard one-dimensional Lagrange interpolation operator for $\{y_{n,j}^{(l)}\}_{j=1}^{p(l)} \subset \Gamma_n$.
- $\Delta_n^{p(l)} := \mathcal{U}_n^{p(l)} - \mathcal{U}_n^{p(l-1)}$.
- $g : \mathbb{N}_+^N \to \mathbb{N}$ is strictly increasing and defines the mapping between the multi-index $\mathbf{l}$ and the level $L$ used to construct the sparse grid.

The *L-th level generalized sparse-grid approximation* of $v \in C^0(\Gamma)$ is given by

$$\mathcal{A}_L^{p,g}[v] = \sum_{g(\mathbf{l})\le L} \bigotimes_{n=1}^{N} \Delta_n^{p(l_n)}[v]$$

$$= \mathcal{A}_{L-1}^{p,g}[v] + \sum_{g(\mathbf{l})=L} \bigotimes_{n=1}^{N} \Delta_n^{p(l_n)}[v]$$

## Reordering of the basis

If we assume that the 1D point sets are nested, the approximation $\mathcal{A}_L^{p,g}[v]$ is a Lagrange interpolating polynomial, and can be rewritten (Wasilkowski, Wozniakowski, '95),

$$A_L^{p,g}[v](\mathbf{y}) = \sum_{j=1}^{M_L} v(\mathbf{y}_j) \underbrace{\sum_{\mathbf{l} \in \mathcal{J}(j,L)} \sum_{\mathbf{i} \in \{0,1\}^N} (-1)^{|\mathbf{i}|} \prod_{n=1}^{N} \psi_{k_n}^{l_n - i_n}(y_n)}_{\Psi_{L,j}(\mathbf{y})} .$$

Here $\{\mathbf{y}_j\}_{j=1}^{M_L}$ is the reordered set of the $M_L$ interpolation points involved in $\mathcal{A}_L^{p,g}$. The functions $\{\Psi_{L,j}\}_{j=1}^{M_L}$ are given by a linear combination of tensorized Lagrange polynomials.

## Reordering of the basis

If we assume that the 1D point sets are nested, the approximation $\mathcal{A}_L^{p,g}[v]$ is a Lagrange interpolating polynomial, and can be rewritten (Wasilkowski, Wozniakowski, '95),

$$A_L^{p,g}[v](\mathbf{y}) = \sum_{j=1}^{M_L} v(\mathbf{y}_j) \underbrace{\sum_{\mathbf{l}\in\mathcal{J}(j,L)} \sum_{\mathbf{i}\in\{0,1\}^N} (-1)^{|\mathbf{i}|} \prod_{n=1}^{N} \psi_{k_n}^{l_n - i_n}(y_n)}_{\Psi_{L,j}(\mathbf{y})} .$$

Here $\{\mathbf{y}_j\}_{j=1}^{M_L}$ is the reordered set of the $M_L$ interpolation points involved in $\mathcal{A}_L^{p,g}$. The functions $\{\Psi_{L,j}\}_{j=1}^{M_L}$ are given by a linear combination of tensorized Lagrange polynomials.

This provides motivation for our acceleration scheme: $\{\mathbf{y}_j\}_{j=1}^{M_L} \subset \{\mathbf{y}_j\}_{j=1}^{M_{L+1}}$, and we can reuse point evaluations from level to level

## Polynomial Spaces

Choose $g$ and $p$ to construct the SG approximation in a given polynomial space:

$$\mathcal{A}_L^{p,g}[v] = \sum_{g(\mathbf{l}) \leq L} \bigotimes_{n=1}^{N} \Delta_n^{p(l_n)}[v]$$

| **Polynomial Space** | $p(l_n)$ | $g(\mathbf{l})$ |
|---|---|---|
| **Tensor product** | $p(l_n) = l_n$ | $\max\limits_{1 \leq n \leq N}(l_n - 1)$ |
| **Total degree** | $p(l_n) = l_n$ | $\sum_{n=1}^{N}(l_n - 1)$ |
| **Hyperbolic cross** | $p(l_n) = l_n$ | $\prod_{n=1}^{N}(l_n - 1)$ |
| **Sparse Smolyak** | $p(l_n) = 2^{l_n - 1} + 1,\ l > 1$ | $\sum_{n=1}^{N}(l_n - 1)$ |
| **Anisotropic Smolyak** | $p(l_n) = 2^{l_n - 1} + 1,\ l > 1$ | $\sum_{n=1}^{N} \frac{\alpha_n}{\alpha_{\min}}(l_n - 1),\ \boldsymbol{\alpha} \in \mathbb{R}_+^N$ |

## Convergence of Sparse Grid Interpolants

We will use the sparse Smolyak construction, with Clenshaw-Curtis abscissas for $\Gamma_n = [-1, 1]$ and $p(l_n) > 1$:

$$y_{n,j}^{(l_n)} = -\cos\left(\frac{\pi(j-1)}{p(l_n)-1}\right) \quad \text{for } j = 1, \ldots, p(l_n).$$

These choices result in a nested set of multidimensional collocation points.

## Convergence of Sparse Grid Interpolants

We will use the sparse Smolyak construction, with Clenshaw-Curtis abscissas for $\Gamma_n = [-1, 1]$ and $p(l_n) > 1$:

$$y_{n,j}^{(l_n)} = -\cos\left(\frac{\pi(j-1)}{p(l_n)-1}\right) \quad \text{for } j = 1, \ldots, p(l_n).$$

These choices result in a nested set of multidimensional collocation points.

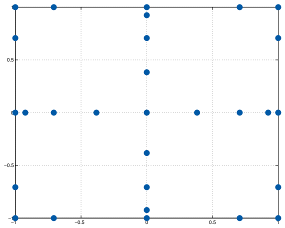
Let $\mathcal{H}_L$—a set of size $\#(\mathcal{H}_L) = M_L$, to be the set of multiD interpolation points

- **Tensor product:** $\mathcal{O}(M_L^{-r/N})$
- **Isotropic sparse grids:** $\mathcal{O}(M_L^{-r/\log(N)})$
  [Smolyak 1963; Nobile-Tempone-Webster 2008]
- **Anisotropic sparse grids:** $\mathcal{O}(M_L^{-r/G(\boldsymbol{\alpha},N)})$
  [Nobile-Tempone-Webster 2008]

## Main idea: exploit the stochastic hierarchy

When nested sparse grids are used to construct a global Lagrange interpolant, we can solve the PDE at samples points in a hierarchical order (*i.e.*, proceed level by level), and construct coarse grid solutions at intermediate steps in the construction of the full interpolant.

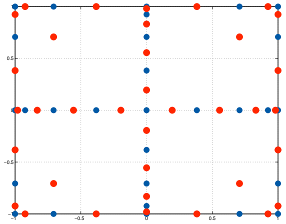
Since we have to solve a linear system at each collocation sample point, we can use these intermediate solutions to provide strong preconditioners and good initial guesses for the iterative solver.



Solve $A_j\mathbf{c}_j = \mathbf{f}_j$ at all blue points

$\longrightarrow$

Interpolate to accelerate solution



## Some Previous Work

**Gordon, Powell.** *Solving Stochastic Collocation with Algebraic Multigrid*

- Compare mean-based, full multigrid, and multigrid + recycled setup for preconditioning SC systems
- They also implement a "nearest neighbor" strategy for initial vectors

**Gunzburger, Webster, Zhang.** *Stochastic finite element methods for PDEs with random input data*

- Implement acceleration using local basis functions
- No significant additional cost for interpolation

**Relationship to multilevel:** Multilevel methods reduce the complexity of stochastic sampling methods by balancing errors across a sequence of stochastic and spatial approximations.

$$u_K^{(\mathrm{ML})} = \sum_{k=0}^{K} \mathcal{I}_{L_{K-k}}[u_{h_k} - u_{h_{k-1}}]$$

## Construction of fully discrete solution

For a prescribed accuracy $\tau > 0$, the semi-discrete solution $u_h(x, \mathbf{y}_j)$ is approximated by

$$u_h(x, \mathbf{y}_j) = \sum_{i=1}^{N_h} c_{j,i} \varphi_i(x) \approx \widetilde{u}_h(x, \mathbf{y}_j) = \sum_{i=1}^{N_h} \widetilde{c}_{j,i} \varphi_i(x),$$

where

$$\widetilde{\mathbf{c}}_j = \left(\widetilde{c}_{j,1}, \dots, \widetilde{c}_{j,N_h}\right)^T$$

is the output of the solver satisfying $\|\mathbf{c}_j - \widetilde{\mathbf{c}}_j\|_{A_j} < \tau$.

## Construction of fully discrete solution

For a prescribed accuracy $\tau > 0$, the semi-discrete solution $u_h(x, \mathbf{y}_j)$ is approximated by

$$u_h(x, \mathbf{y}_j) = \sum_{i=1}^{N_h} c_{j,i}\varphi_i(x) \approx \widetilde{u}_h(x, \mathbf{y}_j) = \sum_{i=1}^{N_h} \widetilde{c}_{j,i}\varphi_i(x),$$

where

$$\widetilde{\mathbf{c}}_j = \left(\widetilde{c}_{j,1}, \ldots, \widetilde{c}_{j,N_h}\right)^T$$

is the output of the solver satisfying $\|\mathbf{c}_j - \widetilde{\mathbf{c}}_j\|_{A_j} < \tau$. Our final approximation is given by :

$$\boxed{\widetilde{u}_{h,L}(x, \mathbf{y}) := \sum_{j=1}^{M_L} \left( \sum_{i=1}^{N_h} \widetilde{c}_{j,i}\varphi_i(x) \right) \psi_{L,j}(\mathbf{y}).}$$

## Interpolated initial vectors

Convergence of CG for $A_j\mathbf{c}_j = \mathbf{f}_j$:

$$\|\mathbf{c}_j - \mathbf{c}_j^{(k)}\|_{A_j} \leq 2\left(\frac{\sqrt{\kappa_j}-1}{\sqrt{\kappa_j}+1}\right)^k \|\mathbf{c}_j - \mathbf{c}_j^{(0)}\|_{A_j}$$

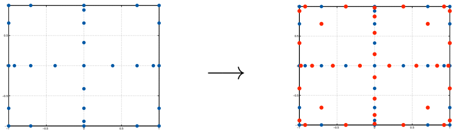
Improving the performance of the CG solver is a matter of either improving the condition number $\kappa_j$, or improving the initial guess $\mathbf{c}_j^{(0)}$.

## Interpolated initial vectors

Convergence of CG for $A_j\mathbf{c}_j = \mathbf{f}_j$:

$$\|\mathbf{c}_j - \mathbf{c}_j^{(k)}\|_{A_j} \le 2\left(\frac{\sqrt{\kappa_j}-1}{\sqrt{\kappa_j}+1}\right)^k \|\mathbf{c}_j - \mathbf{c}_j^{(0)}\|_{A_j}$$

Improving the performance of the CG solver is a matter of either improving the condition number $\kappa_j$, or improving the initial guess $\mathbf{c}_j^{(0)}$.



Specifically, assume we have solved for each the vectors $\widetilde{\mathbf{c}}_m, m = 1, \ldots, M_{L-1}$. Then for any new point $\mathbf{y}_j \in \Delta\mathcal{H}_L$, a good approximation to $\mathbf{c}_j$ is given by

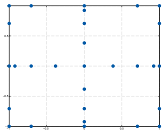
$$\mathbf{c}_j^{(0)} = \sum_{m=1}^{M_{L-1}} \widetilde{\mathbf{c}}_m \psi_{L-1,m}(\mathbf{y}_j).$$

## Interpolated initial vectors

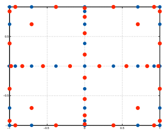
Convergence of CG for $A_j \mathbf{c}_j = \mathbf{f}_j$:

$$\|\mathbf{c}_j - \mathbf{c}_j^{(k)}\|_{A_j} \leq 2 \left( \frac{\sqrt{\kappa_j} - 1}{\sqrt{\kappa_j} + 1} \right)^k \|\mathbf{c}_j - \mathbf{c}_j^{(0)}\|_{A_j}$$

Improving the performance of the CG solver is a matter of either improving the condition number $\kappa_j$, or improving the initial guess $\mathbf{c}_j^{(0)}$.



$\longrightarrow$



Alternatively, suppose we have constructed strong preconditioners $\mathbf{P}_m, m = 1, \ldots, M_{L-1}$. Then for any new point $\mathbf{y}_j \in \Delta \mathcal{H}_L$,

$$\mathbf{P}_j = \sum_{m=1}^{M_{L-1}} \mathbf{P}_m \psi_{L-1,m}(\mathbf{y}_j).$$

The total error $e = u(x, y) - \tilde{u}_{h,M_L}(x, \mathbf{y})$ can be split into

$$\|e\| \leq \underbrace{\|u - u_h\|}_{e_1} + \underbrace{\|u_h - u_{h,M_L}\|}_{e_2} + \underbrace{\|u_{h,M_L} - \tilde{u}_{h,M_L}\|}_{e_3 \text{(solver error)}}$$

The total error $e = u(x,y) - \tilde{u}_{h,M_L}(x,\mathbf{y})$ can be split into

$$\|e\| \le \underbrace{\|u - u_h\|}_{e_1} + \underbrace{\|u_h - u_{h,M_L}\|}_{e_2} + \underbrace{\|u_{h,M_L} - \tilde{u}_{h,M_L}\|}_{e_3(\text{solver error})}$$

Sufficient conditions to achieve overall error $\le \varepsilon$:

$$\|e_1\| \le C_1\, h^{s} \le \frac{\varepsilon}{3}$$
$$\|e_2\| \le C_2(N)\, e^{-\sigma(N)L} \le \frac{\varepsilon}{3}$$
$$\|e_3\| \le C_3\, \Lambda_L\, e_{\mathrm{CG}} \le \frac{\varepsilon}{3}$$

Here $s$, $\sigma(N)$ are the convergence rates of the FEM and interpolation methods, $\Lambda_L$ is the Lebesgue constant, and

$$e_{\mathrm{CG}} = \max_{\mathbf{y}_j \in \mathcal{H}_L} \|\mathbf{c}_j - \tilde{\mathbf{c}}_j\|_{A_j}$$

## Iteration Estimate: Zero Vectors

**Theorem: [Galindo, J, Webster, Zhang]**

Given $\varepsilon > 0$, the total number of CG iterations needed to achieve an error $\| u - \tilde{u}_{h,M_L} \| < \varepsilon$ using zero initial vectors is bounded by:

$$K_{zero} \leq \alpha_1(N)\, \varepsilon^{\frac{-\ln 2}{\sigma}} \left\{ \alpha_2(N) + \alpha_3 \ln \varepsilon^{-1} \right\}^{N-1} \times \sqrt{\bar{\kappa}} \left\{ \ln \varepsilon^{-1} + \ln \Lambda_L + \alpha_4(N) \right\},$$

as $\varepsilon \to 0$, where $\bar{\kappa} = \max_{\mathbf{y} \in \mathcal{H}_L} \bar{\kappa}(\mathbf{y})$.

## Iteration Estimate: Zero Vectors

**Theorem: [Galindo, J, Webster, Zhang]**

Given $\varepsilon > 0$, the total number of CG iterations needed to achieve an error $\|u - \tilde{u}_{h,M_L}\| < \varepsilon$ using zero initial vectors is bounded by:

$$K_{zero} \leq \alpha_1(N)\, \varepsilon^{\frac{-\ln 2}{\sigma}} \left\{\alpha_2(N) + \alpha_3 \ln \varepsilon^{-1}\right\}^{N-1} \\ \times \sqrt{\bar{\kappa}} \left\{\ln \varepsilon^{-1} + \ln \Lambda_L + \alpha_4(N)\right\},$$

as $\varepsilon \to 0$, where $\bar{\kappa} = \max_{\mathbf{y} \in \mathcal{H}_L} \bar{\kappa}(\mathbf{y})$.

The first line (in blue) comes from the number of collocation nodes, which we can't affect by our algorithm. The second part comes from the convergence of the CG algorithm, which for fine grid points we can reduce by a log factor (in red).

## Iteration Estimate: Accelerated Case

**Theorem: [Galindo, J, Webster, Zhang]**

Given $\varepsilon > 0$, the total number of CG iterations needed to achieve an error $\|u - \tilde{u}_{h,M_L}\| < \varepsilon$ using the hierarchically accelerated stochastic collocation algorithm is bounded by:

$$K_{acc} \leq \alpha_1(N)\,\varepsilon^{\frac{-\ln 2}{\sigma}} \left\{\alpha_2(N) + \alpha_3 \ln \varepsilon^{-1}\right\}^{N-1} \\ \times \sqrt{\bar{\kappa}}\,\{\ln \Lambda_L + \alpha_5(N)\}\,,$$

as $\varepsilon \to 0$, where $\bar{\kappa} = \max_{\mathbf{y}\in\mathcal{H}_L} \bar{\kappa}(\mathbf{y})$.

To perform this method, we incur an addition cost of interpolation (and preconditioning).

## Remarks

- The condition number of the systems has a big effect on the complexity, but is hard to specify in general.

## Remarks

- The condition number of the systems has a big effect on the complexity, but is hard to specify in general.
- We actually observe increased % savings in iterations vs error as dimension increases. (Example 2)
  - An alternative estimate shows an iterations savings of $(2^{1/N} - 1) \log \varepsilon^{-1}$
- This method is most effective when sampling is relatively expensive, e.g. when the underlying deterministic PDE is more difficult to solve than the interpolation problem. (Example 3)

## Remarks

- The condition number of the systems has a big effect on the complexity, but is hard to specify in general.
- We actually observe increased % savings in iterations vs error as dimension increases. (Example 2)
    - An alternative estimate shows an iterations savings of $(2^{1/N} - 1) \log \varepsilon^{-1}$
- This method is most effective when sampling is relatively expensive, e.g. when the underlying deterministic PDE is more difficult to solve than the interpolation problem. (Example 3)
- The acceleration scheme should always be used in adaptive interpolation settings, or with sparse grids based on hierarchical Lagrange interpolants.

## Example 1: **Local** Basis for Sparse Grid SC

Consider the 2D Poisson equation with random diffusivity and forcing term, i.e.,

$$\begin{cases} -\nabla \cdot (a(x,\mathbf{y})\nabla u(x,\mathbf{y})) &= f(x,\mathbf{y}) \quad [0,1]^2 \times \Gamma \\ u(\mathbf{x},\mathbf{y}) &= 0 \qquad \text{on } \partial D \times \Gamma \end{cases}$$

where $a$ and $f$ are the nonlinear functions of the random vector $\mathbf{y}$ given by

$$a(x,\mathbf{y}) = 0.1 + \exp\left[y_1 \cos(\pi x_1) + y_2 \sin(\pi x_2)\right],$$

and

$$f(x,\mathbf{y}) = 10 + \exp\left[y_3 \cos(\pi x_1) + y_4 \sin(\pi x_2)\right],$$

- $y_n$, $n = 1,2,3,4$, are i. i. d. random variables following the uniform distribution $U([-1,1])$
- The quantity of interest is the mean value of the solution over $D \times \Gamma$, i.e.

$$\text{QoI} = \mathbb{E}\left[\int_D u(x,\mathbf{y})dx\right]$$

## Computational savings: Local Basis

Gunzburger, Webster, Zhang. **Stochastic finite element methods for PDEs with random input data**, 2014 *Acta Numerica*

Table: The computational savings of the local SG with hierarchical acceleration

| Basis type | Error | # SG points | hSGSC cost | hSGSC+acceleration cost | hSGSC+acceleration saving |
|---|---|---|---|---|---|
| Linear | 1.0e-2 | 377 | 13,841 | 7,497 | 45.8% |
| | 1.0e-3 | 1,893 | 81,068 | 38,670 | 52.2% |
| | 1.0e-4 | 7,777 | 376,287 | 167,832 | 55.3% |
| Quadratic | 1.0e-3 | 701 | 29,874 | 11,877 | 60.2% |
| | 1.0e-4 | 2,285 | 110,744 | 36,760 | 66.8% |
| | 1.0e-5 | 6,149 | 329,294 | 100,420 | 69.5% |
| Cubic | 1.0e-4 | 1,233 | 59,344 | 23,228 | 60.8% |
| | 1.0e-5 | 3,233 | 172,845 | 57,777 | 66.5% |
| | 1.0e-6 | 7,079 | 415,760 | 129,433 | 68.8% |

## Example 2: Global Basis w/ Error Balancing

We consider a 1D Poisson equation with random diffusivity term:

$$\begin{aligned} -\nabla \cdot (a(x,\mathbf{y})\nabla u(x,\mathbf{y})) &= 10 \text{ in } [0,1] \times \Gamma \\ u(x,\mathbf{y}) &= 0 \text{ on } \partial D \times \Gamma \end{aligned}$$

with

$$a(x,\mathbf{y}) = 1 + \exp\left\{\exp^{-1/8}(y_1 \cos \pi x + y_2 \sin \pi x + y_3 \cos 2\pi x + y_4 \sin 2\pi x)\right\}$$

## Example 2: Global Basis w/ Error Balancing

We consider a 1D Poisson equation with random diffusivity term:

$$\begin{aligned} -\nabla \cdot (a(x,\mathbf{y})\nabla u(x,\mathbf{y})) &= 10 \text{ in } [0,1] \times \Gamma \\ u(x,\mathbf{y}) &= 0 \text{ on } \partial D \times \Gamma \end{aligned}$$

with

$$a(x,\mathbf{y}) = 1 + \exp\left\{\exp^{-1/8}(y_1 \cos \pi x + y_2 \sin \pi x + y_3 \cos 2\pi x + y_4 \sin 2\pi x)\right\}$$

| Error | #SG Pts | CG iters | CG + acc | % Savings |
|---|---|---|---|---|
| $1 \times 10^{-2}$ | 137 | 29,355 | 22,219 | 24.3 |
| $5 \times 10^{-3}$ | 401 | 180,087 | 90,300 | 49.9 |
| $1 \times 10^{-3}$ | 1105 | 2,072,625 | 696,935 | 66.4 |
| $5 \times 10^{-4}$ | 2929 | 11,253,264 | 2,217,615 | 80.3 |
| $1 \times 10^{-4}$ | 7537 | 118,429,119 | 16,204,912 | 86.3 |

Table: Iterations and savings between the hierarchically acclerated SG method and the zero vector method

## Computed Cost

### A Metric for Computational Cost

**Zero Vectors:** $Cost_{\text{zero}} = C_D N_h K_{\text{zero}}$

**Acceleration:** $Cost_{\text{acc}} = C_D N_h K_{\text{acc}} + Cost_{\text{int}}$

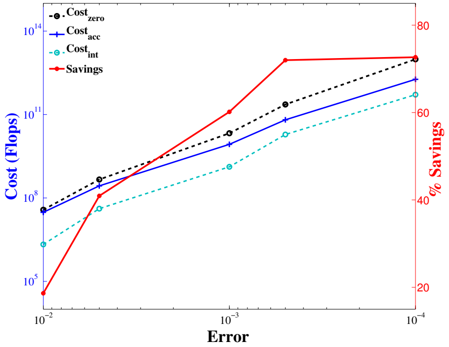


Figure: Cost (left axis) and percent savings (right axis) in flops of the hierarchically accelerated SG method versus the zero vector method.

## Example 3: Global Basis w/ Interpolated Preconditioners

Let $\mathbf{x} = (x_1, x_2)$ and consider the following linear elliptic SPDE:

$$\begin{cases} -\nabla \cdot (a(x_1, \mathbf{y}) \nabla u(\mathbf{x}, \mathbf{y})) &= \cos(x_1)\sin(x_2) \quad [0,1]^2 \times \Gamma \\ u(\mathbf{x}, \mathbf{y}) &= 0 \qquad\qquad\qquad \text{on } \partial D \times \Gamma \end{cases}$$

The diffusion coefficient is a 1d random field (varies only in $x_1$) and is $a(x_1, \mathbf{y}) = 0.5 + \exp\{\gamma(x_1, \mathbf{y})\}$, where $\gamma$ is a truncated random field with correlation length $R$ and covariance

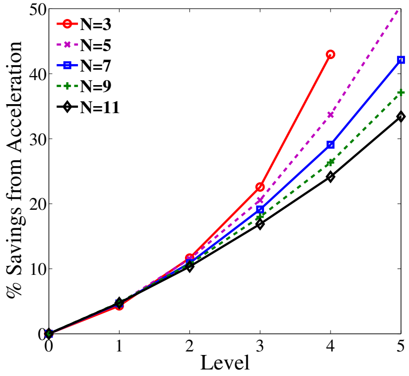
$$Cov[\gamma](x_1, \tilde{x}_1) = \exp\left(-\frac{(x_1 - \tilde{x}_1)^2}{R^2}\right), \quad \forall (x_1, \tilde{x}_1) \in [0,1]$$

$$\gamma(x_1, \mathbf{y}) = 1 + y_1 \left(\frac{\sqrt{\pi}R}{2}\right)^{1/2} + \sum_{n=2}^{N} \beta_n \, \varphi_n(x_1) \, y_n$$

$$\beta_n := \left(\sqrt{\pi}R\right)^{1/2} e^{\frac{-\left(\lfloor \frac{n}{2} \rfloor \pi R\right)^2}{8}}, \quad \varphi_n(x_1) := \begin{cases} \sin\left(\lfloor \frac{n}{2} \rfloor \pi x_1\right), & \text{if } n \text{ even}, \\ \cos\left(\lfloor \frac{n}{2} \rfloor \pi x_1\right), & \text{if } n \text{ odd} \end{cases}$$

- $\mathbb{E}[y_n] = 0$ and $\mathbb{E}[y_n y_m] = \delta_{nm}$ for $n, m \in \mathbb{N}_+$ and iid in $U(-\sqrt{3}, \sqrt{3})$

## 2D example: Iterations vs Level



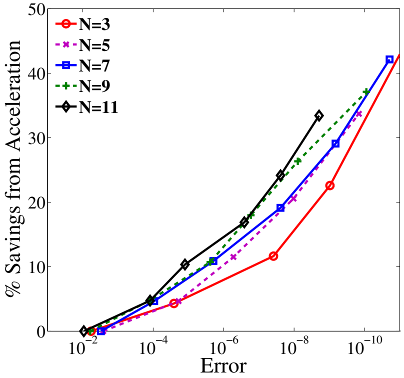
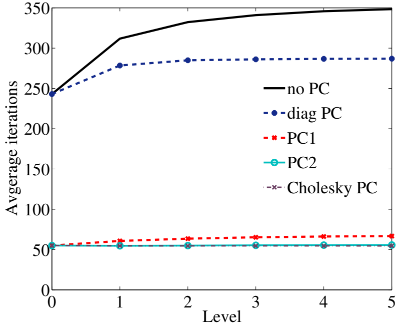


Figure: Percentage reduction in CG iterations per level (left) and vs error (right) with $N = 3, 5, 7, 9, 11$ and 13 and for correlation length $R_c = 1/64$

## Preconditioning



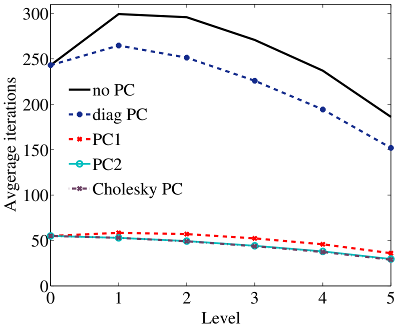


Figure: Average CG iterations per level for $N = 7$ for $L = 1/64$, with zero initial vectors (left) and with accelerated initial vectors(right), for preconditioning options: diagonal preconditioner, 1-3 level interepolated incomplete Cholesky preconditioner, and fully locally adapted incomplete Cholesky preconditioner

## Example 4: Nonlinear problem

We consider a 1D Poisson equation with random diffusivity term:

$$\begin{aligned} \nabla \cdot (a(x,\mathbf{y})\nabla u(x,\mathbf{y})) + F[u](x,\mathbf{y}) &= 10 \text{ in } [0,1] \times \Gamma \\ u(x,\mathbf{y}) &= 0 \text{ on } \partial D \times \Gamma \end{aligned}$$

with $a$ as in Example 2:

$$a(x,\mathbf{y}) = 1 + \exp\left\{\exp^{-1/8}(y_1 \cos \pi x + y_2 \sin \pi x + y_3 \cos 2\pi x + y_4 \sin 2\pi x)\right\}.$$

We'll test our method using

$$F[u] = u^2 \text{ and } F[u] = u * u'$$

## Example 4: Nonlinear problem

We consider a 1D Poisson equation with random diffusivity term:

$$\begin{aligned} \nabla \cdot (a(x,\mathbf{y})\nabla u(x,\mathbf{y})) + F[u](x,\mathbf{y}) &= 10 \text{ in } [0,1] \times \Gamma \\ u(x,\mathbf{y}) &= 0 \text{ on } \partial D \times \Gamma \end{aligned}$$

with $a$ as in Example 2:

$$a(x,\mathbf{y}) = 1 + \exp\left\{\exp^{-1/8}(y_1 \cos \pi x + y_2 \sin \pi x + y_3 \cos 2\pi x + y_4 \sin 2\pi x)\right\}.$$

We'll test our method using

$$F[u] = u^2 \text{ and } F[u] = u * u'$$

---

- For nonlinear iterative methods, a better initial guess can lead to better convergence rates!
- In this case, each iteration corresponds to a full system solve

## Example 4: Timings

| SC Level | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| $F[u] = u^5$, acc | .03018 | .113832 | .2746 | .7039 | 2.33314 |
| $F[u] = u^5$, zero | .025976 | .119256 | .339678 | .949184 | 2.61958 |
| **% Savings** | -16.2 | 4.5 | 19.2 | 25.8 | 10.9 |
| $F[u] = uu'$, acc | .027754 | .089082 | .22706 | .629451 | 2.05741 |
| $F[u] = uu'$, zero | .026527 | .090435 | .273355 | .895027 | 2.4008 |
| **% Savings** | -4.6 | 1.5 | 16.9 | 29.7 | 14.3 |

Table: Computational time in seconds for computing solution to nonlinear problem

- These timings use a fixed spatial discretization, without error balancing.
- The decrease in savings at the end is due to the increasing complexity of the interpolation problem versus that of the nonlinear PDE

## Example 4: $-\nabla(a \cdot \nabla u) + F(u) = f$

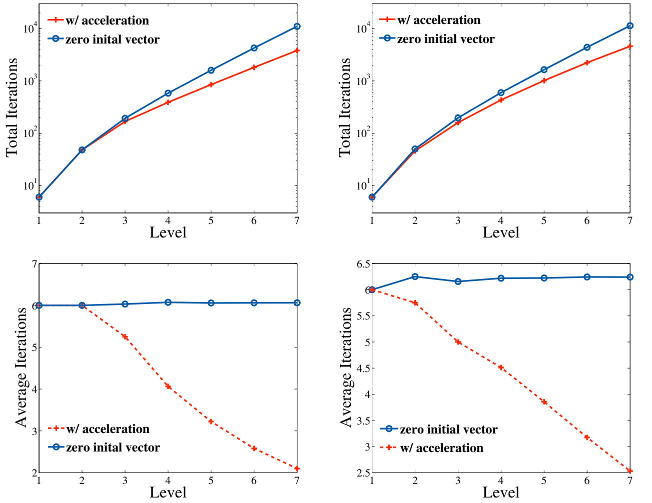


Figure: Cumulative total (top) and average per-level (bottom) number of Newton iterations with $F(u) = u * u'$ (left) and $F(u) = u^5$ (right)

## Remarks

- Acceleration works together with preconditioning to speed up solvers.
- Especially effective for more complicated or non-linear PDEs
- Improves efficiency of iterative solvers even with the additional cost of interpolation
- The natural next step would be to combine this acceleration with (spatial) multilevel methods

---

Galindo, Jantsch, Webster, Zhang
**Accelerating Hierarchical Stochastic Collocation Method for Random PDEs**, 2014 (*Submitted*)

Gunzburger, Jantsch, Teckentrup, Webster
**Multilevel Stochastic Collocation Methods for Random PDEs**, 2014 (*Submitted*)

## Future Directions

- Many people have been working on adaptively selecting the best polynomial subspaces for approximation of a given analytic function. (Gerstner, Griebel, 2003; Nobile, Tempone, Tamellini 2014)
- These methods work best when applied on hierarchical sparse grids.
- Nested grids provide a framework for adaptive approximation, and now acceleration.