OAK RIDGE National Laboratory
COMPUTATIONAL & APPLIED MATHEMATICS

Uncertainty quantification & approximation theory for parameterized (stochastic) PDEs

Part V: High-dimensional sparse grids via global Lagrange polynomials

Clayton G. Webster[†⋆]

Hoang Tran[⋆], Guannan Zhang[⋆]

[†]Department of Mathematics, University of Tennessee

[⋆]Department of Computational & Applied Mathematics (CAM)
Oak Ridge National Laboratory

Supporting agencies: **DOE** (ASCR, BES), **DOD** (AFOSR, DARPA), **NSF** (CM)

`equinox.ornl.gov` `formulate.ornl.gov` `tasmanian.ornl.gov`

Part IV Outline

High-dimensional sparse grids via global Lagrange polynomials

1. Multi-dimensional interpolation
2. Generalized global sparse grid interpolation
3. Example: sparse grid SCFEM for a parameterized stochastic PDE
4. Reducing the computational cost of multivariate interpolation

Generalized multivariate interpolation

Stochastic collocation FEM: general setting

1. Choose a set of points $\{\boldsymbol{y}_k \in \Gamma\}_{k=1}^{m_p}$ according to the measure $\varrho(\boldsymbol{y})d(\boldsymbol{y}) = \prod_{n=1}^{d} \varrho_n(y_n)d(y_n)$.
2. For each k solve the FE solution $u_k(x) = u(\boldsymbol{y}_k, x)$, given $a_k(x) = a(\boldsymbol{y}_k, x)$ and $f_k(x) = f(\boldsymbol{y}_k, x)$.
3. Interpolate the sampled values:

$$\mathcal{I}_{\Lambda_p}[u] = \sum_{k=1}^{m_p} u_k(x)\ell_k(\boldsymbol{y}) \in \mathbb{P}_{\Lambda_p}(\mathcal{U}) \otimes \mathcal{V}_h,$$

yielding the fully discrete SC approximation in, where $\ell_k \in \mathbb{P}_{\Lambda_p}(\mathcal{U})$ are suitable combinations of **global** (Lagrange) interpolants.

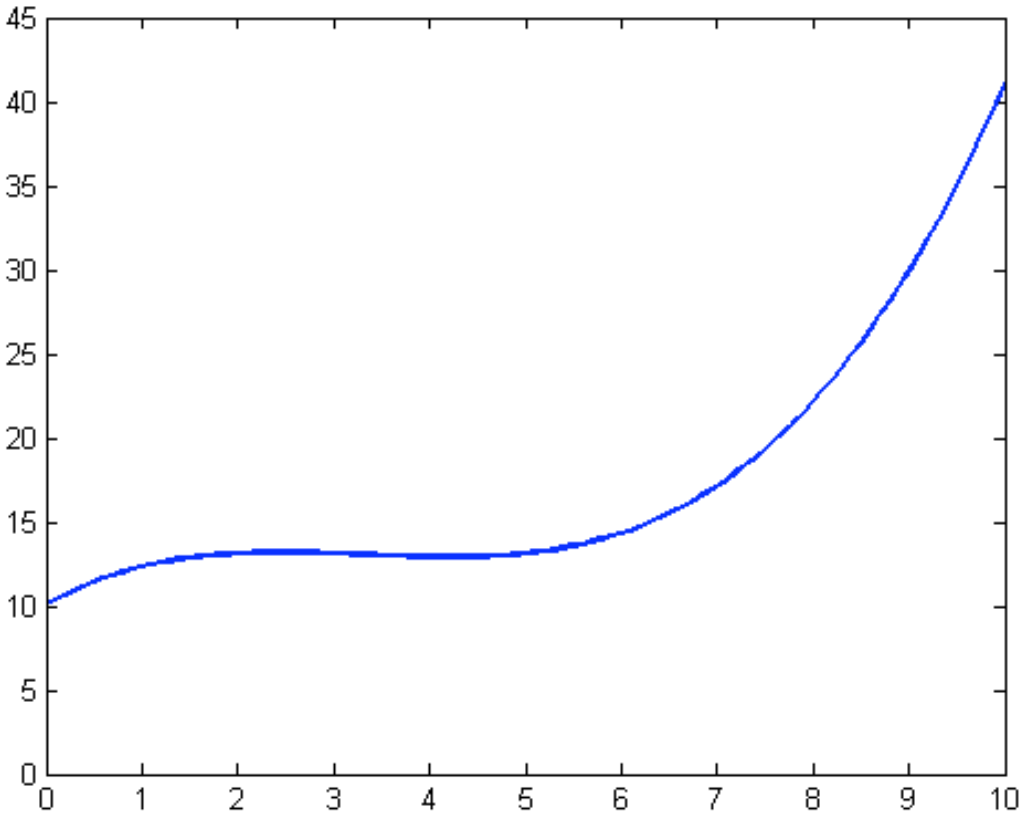
Compute a quantity of interest, e.g., $\mathbb{E}[u](x)$

$$\mathbb{E}[u](x) \approx \int_\Gamma \mathcal{I}_{\Lambda_p}[u](\cdot, \boldsymbol{y})\varrho(\boldsymbol{y})d\boldsymbol{y} = \sum_{k=1}^{m_p} u_k(x) \underbrace{\int_\Gamma \ell_k(\boldsymbol{y})\varrho(\boldsymbol{y})dy}_{\text{precomputed weights}} = \sum_{k=1}^{m_p} u_k(x)w_k$$

Quantities of interest

Interpolatory quadrature

A simple function to integrate

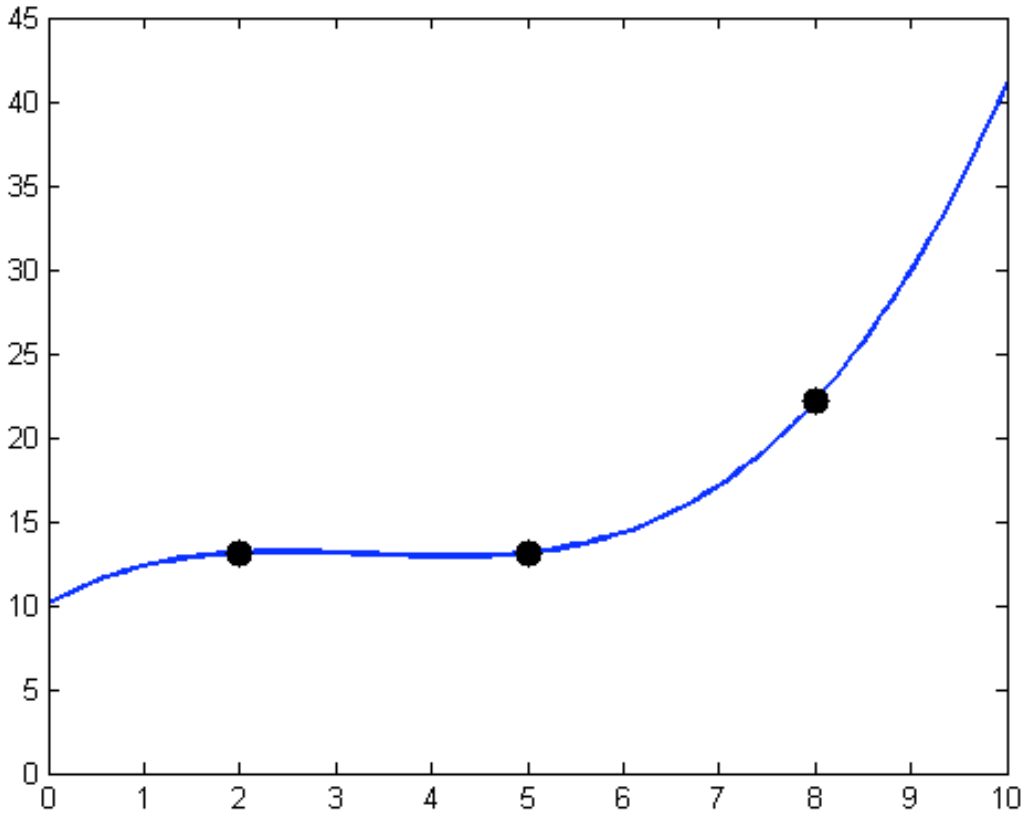


$u(y)$ is a given function

Quantities of interest

Interpolatory quadrature

Selected function values

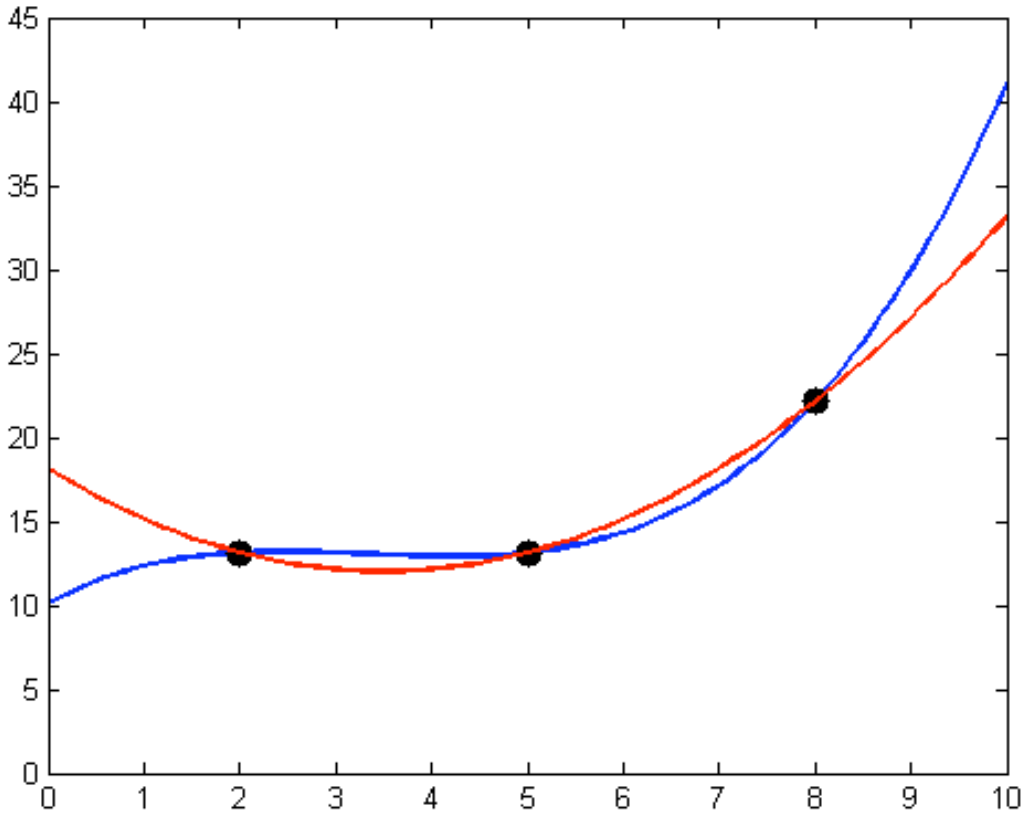


Evaluate $u(y)$ at M values $\{u(y_1), u(y_2), \ldots, u(y_M)\}$

Quantities of interest

Interpolatory quadrature

The interpolant

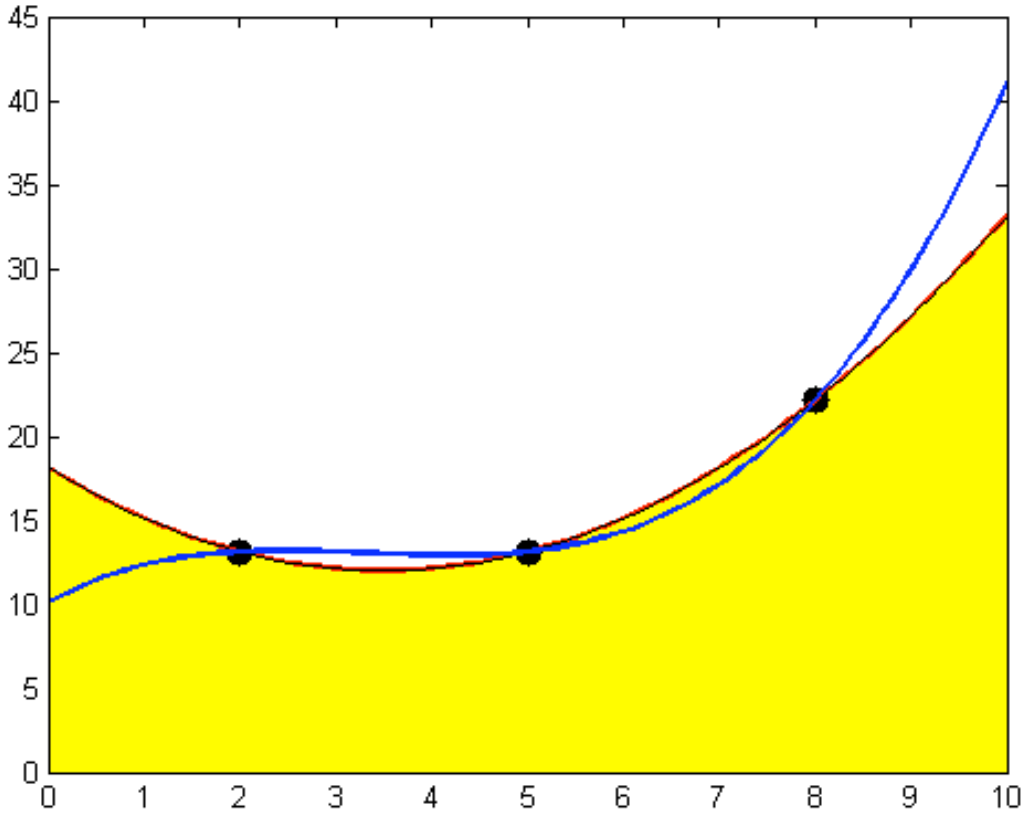


Determine the approximate polynomial u_p

Quantities of interest

Interpolatory quadrature

The area under the curve



The QoI = Integrating the approximating polynomial EXACTLY

Multivariate tensor product interpolation

The simplest SCFEM

Basic idea is to construct the point set for each variable y_n: $H^{m(l_n)}$

- choose the level l_n of interpolation in the nth direction
- set the number of points used by the l_nth interpolant, denoted $m(l_n)$
- define the set $\left\{y_n^1, y_n^2, \ldots, y_n^{m(l_n)}\right\}$ of 1d interpolating points:

  according to the measure $\varrho(y_n)dy_n$, e.g. Gauss-Hermite (Normal), Gauss-Legendre, Clenshaw-Curtis (Uniform), etc.
- The total number of tensor interpolation nodes is: $m_{\text{TP}} = m(l_1)m(l_2)\ldots m(l_d)$
- $\boldsymbol{y_k} = \left(y_1^{k_1}, y_2^{k_2}, \ldots, y_N^{k_N}\right)$, where $\boldsymbol{k} \in \text{TP} \equiv \left\{\boldsymbol{k} \in \mathbb{N}_+^d \ : \ k_n < m(l_n)\right\}$

The tensor product (TP) Lagrange-interpolant is defined by:

$$u_{\text{TP}} = \sum_{\boldsymbol{k}\in\text{TP}}^{m_{\text{TP}}} u_{\boldsymbol{k}}(x)\ell_{\boldsymbol{k}}(\boldsymbol{y}), \quad \text{with } \ell_{\boldsymbol{k}}(\boldsymbol{y}) = \prod_{n=1}^{d} \prod_{s=1, s\neq k_n}^{m(l_n)} \frac{y_n - y_n^s}{y_n^{k_n} - y_n^s}$$

Multivariate tensor product interpolation

The simplest SCFEM

Basic idea is to construct the point set for each variable y_n: $H^{m(l_n)}$

- choose the level l_n of interpolation in the nth direction
- set the number of points used by the l_nth interpolant, denoted $m(l_n)$
- define the set $\left\{y_n^1, y_n^2, \ldots, y_n^{m(l_n)}\right\}$ of 1d interpolating points:

  according to the measure $\varrho(y_n)dy_n$, e.g. Gauss-Hermite (Normal), Gauss-Legendre, Clenshaw-Curtis (Uniform), etc.
- The total number of tensor interpolation nodes is: $m_{\text{TP}} = m(l_1)m(l_2)\ldots m(l_d)$
- $\boldsymbol{y_k} = \left(y_1^{k_1}, y_2^{k_2}, \ldots, y_N^{k_N}\right)$, where $\boldsymbol{k} \in \text{TP} \equiv \left\{\boldsymbol{k} \in \mathbb{N}_+^d \ : \ k_n < m(l_n)\right\}$

The tensor product (TP) Lagrange-interpolant is defined by:

$$u_{\text{TP}} = \sum_{\boldsymbol{k}\in\text{TP}}^{m_{\text{TP}}} u_{\boldsymbol{k}}(x)\ell_{\boldsymbol{k}}(\boldsymbol{y}), \quad \text{with } \ell_{\boldsymbol{k}}(\boldsymbol{y}) = \prod_{n=1}^{d} \prod_{s=1, s\neq k_n}^{m(l_n)} \frac{y_n - y_n^s}{y_n^{k_n} - y_n^s}$$

Generalized tensor product interpolation

Let $\mathcal{I}_n^{m(l_n)}$ be the l_nth level interpolant in the direction y_n using $m(l_n)$ points:

$$\mathcal{I}_n^{m(l_n)}[u](y_n) = \sum_{k=1}^{m(l_n)} u(y_n^k)\ell_n^k(y_n), \quad \{y_n^1, \ldots, y_n^{m(l_n)}\} \in \mathcal{U}_n$$

- $\mathcal{I}_n^{m(l_n)} : C^0(\mathcal{U}_n) \to \mathcal{P}_{m(l_n)-1}(\mathcal{U}_n), \quad \mathcal{I}_n^0[u] = 0 \; \forall u \in C^0(\mathcal{U}_n)$
- The degree in the y_n direction is $p_n = m(l_n) - 1$

The generalized tensor product approximation is given by

$$\mathcal{I}_{\Lambda_L^{\mathsf{TP}}}[u] = u_L^{\mathsf{TP}}(\boldsymbol{y}) = \bigotimes_{n=1}^{d} \mathcal{I}_n^{m(l_n)}[u](\boldsymbol{y}), \quad \Lambda_L^{\mathsf{TP}} = \{\boldsymbol{l} \in \mathbb{N}_+^d : \max_n \alpha_n l_n \leq L\}$$

- the interpolation requires $m_{\mathsf{TP}} = \prod_{n=1}^{d} m(l_n)$ function evaluations (In this case, solutions of the PDE)

Generalized tensor product interpolation

Let $\mathcal{I}_n^{m(l_n)}$ be the l_nth level interpolant in the direction y_n using $m(l_n)$ points:

$$\mathcal{I}_n^{m(l_n)}[u](y_n) = \sum_{k=1}^{m(l_n)} u(y_n^k)\ell_n^k(y_n), \quad \{y_n^1, \ldots, y_n^{m(l_n)}\} \in \mathcal{U}_n$$

- $\mathcal{I}_n^{m(l_n)} : C^0(\mathcal{U}_n) \to \mathcal{P}_{m(l_n)-1}(\mathcal{U}_n), \quad \mathcal{I}_n^0[u] = 0 \; \forall u \in C^0(\mathcal{U}_n)$
- The degree in the y_n direction is $p_n = m(l_n) - 1$

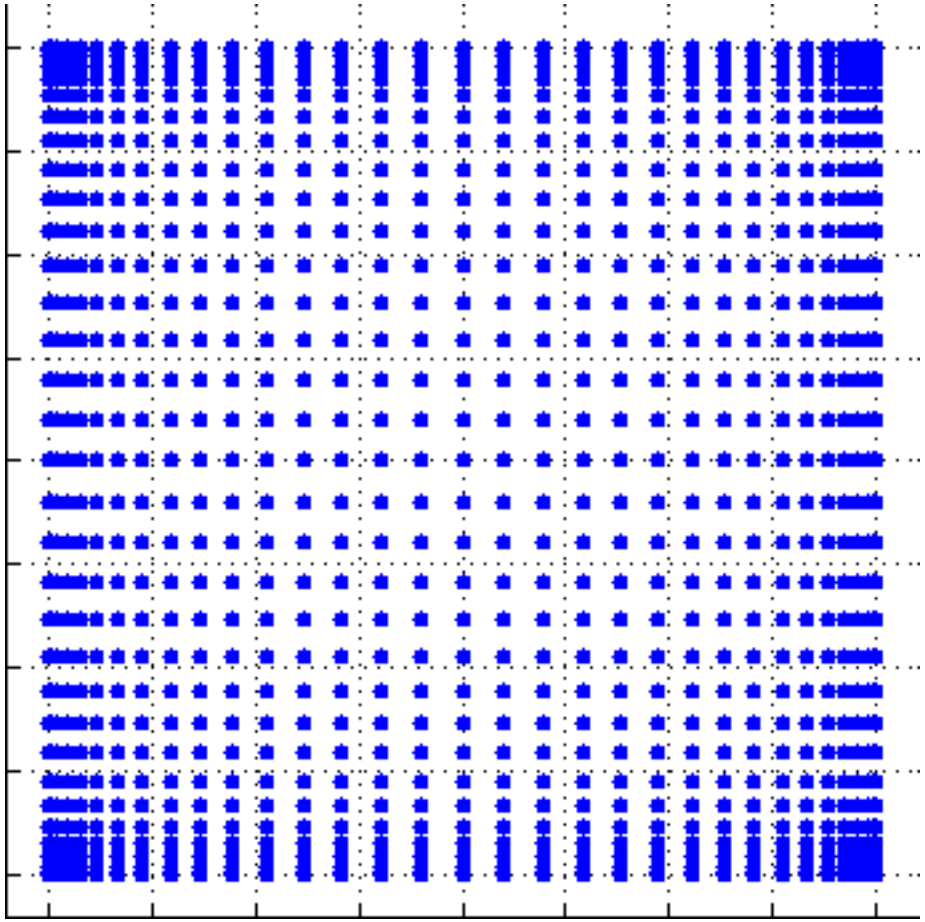
The generalized tensor product approximation is given by

$$\mathcal{I}_{\Lambda_L^{\mathsf{TP}}}[u] = u_L^{\mathsf{TP}}(\boldsymbol{y}) = \bigotimes_{n=1}^{d} \mathcal{I}_n^{m(l_n)}[u](\boldsymbol{y}), \quad \Lambda_L^{\mathsf{TP}} = \{\boldsymbol{l} \in \mathbb{N}_+^d : \max_n \alpha_n l_n \leq L\}$$

- the interpolation requires $m_{\mathsf{TP}} = \prod_{n=1}^d m(l_n)$ function evaluations
  (In this case, solutions of the PDE)

Tensor product grid for $p = 32$

Isotropic grid $\max(p_1, p_2) \leq 32$



Isotropic TP SC grid constructed from C-C points for $(y_1, y_2) \in U(-1, 1)$

Choices for interpolation

Based on 1-d interpolation formulas

Clenshaw-Curtis abscissas ($\mathcal{U}_n$ bounded):

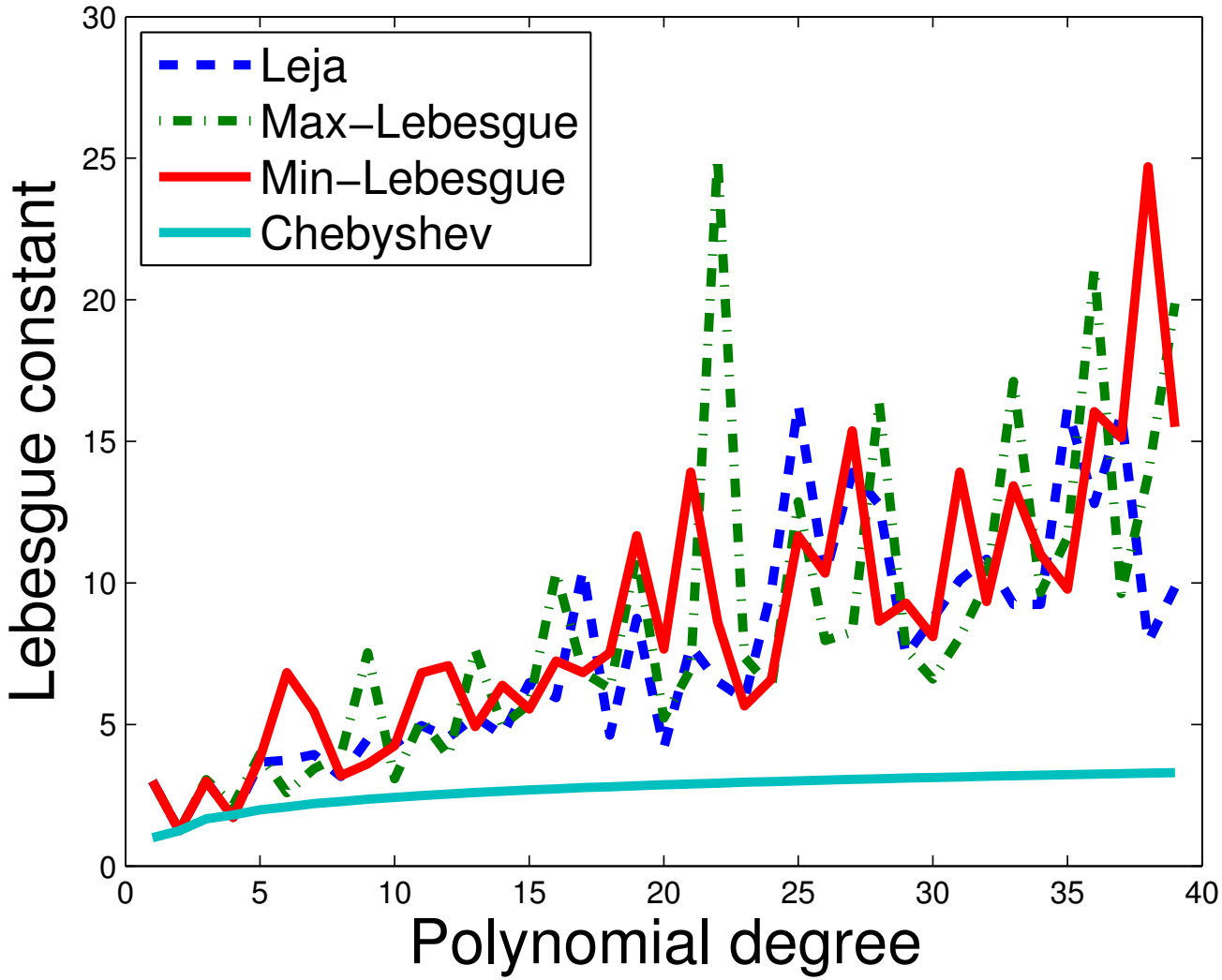
- $\left\{y_n^k\right\}_{k=1}^{m(l_n)}$: extrema of Chebyshev polynomials
- optimal for uniform convergence in $\mathcal{U}_n$
- if $m(l_n) = 2^{i_n-1}+1$ lead to nested sets, i.e. $H_n^{m(l_n)} \subset H_n^{m(l_n+1)}$

Gaussian abscissas ($\mathcal{U}_n$ bounded or unbounded): Assume, either

- y_n independent, i.e. $\varrho(\boldsymbol{y}) = \prod_{n=1}^d \varrho_n(y_n)$, or
- construct an auxiliary joint PDF $\hat{\varrho}(\boldsymbol{y}) = \prod_{n=1}^d \hat{\varrho}_n(y_n)$ such that $\|\varrho/\hat{\varrho}\|_{L^\infty(\Gamma)} < \infty$ and small enough.
- $\left\{y_n^k\right\}_{k=1}^{m(l_n)}$: zeros of orthogonal polynomials with respect to $\hat{\varrho}$
  e.g. abscissas become roots of Gauss-Legendre, -Hermite, -Jacobi, -Laguerre polynomials corresponding to uniform, normal, beta, exponential distributions, respectively
- optimal for L^2_ϱ convergence

Leja sequences ($\mathcal{U}_n$ bounded or unbounded): constructed from the extrema of a residual function

Lebesgue constants for various rules



Generalized sparse grid interpolation

[Nobile, Tempone, W., SINUM (2008); Gunzburger, W., Zhang, Acta Num. (2014)]

- Recall that $\mathcal{I}_n^{m(l_n)}$ be the l_nth level interpolant in direction y_n using $m(l_n)$ points

$$\mathcal{I}_n^{m(l_n)} : C^0(\mathcal{U}_n) \to \mathcal{P}_{m(l_n)-1}(\mathcal{U}_n), \quad \mathcal{I}_n^0[u] = 0 \; \forall u \in C^0(\mathcal{U}_n)$$

- The nth difference operator: $\Delta_n^{m(l_n)}[u] = \mathcal{I}_n^{m(l_n)}[u] - \mathcal{I}_n^{m(l_n-1)}[u]$
- The hierarchical surplus: $\Delta^m[u](\boldsymbol{y}) = \bigotimes_{n=1}^{d} \Delta_n^{m(l_n)}[u](\boldsymbol{y})$
- Let $\boldsymbol{l} = (l_1, \ldots, l_n) \in \mathbb{N}_+^d$ be a multi-index and $g : \mathbb{N}_+^d \to \mathbb{N} \to$ a mapping between a multi-index $\boldsymbol{l}$ and the level p used to construct the sparse grid.

The L-th level generalized sparse- grid approximation of $v \in C^0(\mathcal{U})$ is given by

$$u_L^{\mathsf{SG}} = \mathcal{I}_L^{m,g}[v] = \sum_{g(\boldsymbol{l}) \le L} \bigotimes_{n=1}^{d} \Delta_n^{m(l_n)}[v]$$

$$= \mathcal{I}_{L-1}^{m,g}[v] + \sum_{g(\boldsymbol{l}) = L} \bigotimes_{n=1}^{d} \Delta_n^{m(l_n)}[v]$$

Generalized sparse grid interpolation

How to choose the index set $g(\boldsymbol{l}) \leq L$?

- Can build sparse grids corresponding to any polynomial space $\mathbb{P}_{\Lambda_L}(\mathcal{U})$
  - Tensor product (TP): $m(l) = l$, $g(\boldsymbol{l}) = \max_n \alpha_n (l_n - 1) \leq L$
  - Total Degree (TD): $m(l) = l$, $g(\boldsymbol{l}) = \sum_n \alpha_n (l_n - 1) \leq L$
  - Hyperbolic Cross (HC): $m(l) = l$, $g(\boldsymbol{l}) = \prod_n (l_n)^{\alpha_n} \leq L + 1$
  - Smolyak (SM): $m(l) = 2^{l-1} + 1$, $l > 1$, $g(\boldsymbol{l}) = \sum_n \alpha_n (l_n - 1) \leq L$
- The corresponding anisotropic versions are straightforward
- SM is the most widely used approach and corresponds to the original Smolyak construction [Smolyak '63]

$$\left\| u - u_L^{\text{TP}} \right\|_{L^\infty} \lesssim C e^{-f(\boldsymbol{\alpha}) m^{1/d}} \qquad \text{[Babuška, Nobile, Tempone '06]}$$

$$\left\| u - u_L^{\text{SG}} \right\|_{L^\infty} \lesssim C e^{-f(\boldsymbol{\alpha}) m^{1/\log(d)}} \qquad \text{[Nobile, Tempone, W. '08]}$$

Generalized sparse grid interpolation

How to choose the index set $g(\boldsymbol{l}) \leq L$?

- Can build sparse grids corresponding to any polynomial space $\mathbb{P}_{\Lambda_L}(\mathcal{U})$
  - Tensor product (TP): $m(l) = l,\ g(\boldsymbol{l}) = \max_n \alpha_n(l_n - 1) \leq L$
  - Total Degree (TD): $m(l) = l,\ g(\boldsymbol{l}) = \sum_n \alpha_n(l_n - 1) \leq L$
  - Hyperbolic Cross (HC): $m(l) = l,\ g(\boldsymbol{l}) = \prod_n (l_n)^{\alpha_n} \leq L + 1$
  - Smolyak (SM): $m(l) = 2^{l-1} + 1,\ l > 1,\ g(\boldsymbol{l}) = \sum_n \alpha_n(l_n - 1) \leq L$
- The corresponding anisotropic versions are straightforward
- SM is the most widely used approach and corresponds to the original Smolyak construction [Smolyak '63]

$$\left\| u - u_L^{\text{TP}} \right\|_{L^\infty} \lesssim C e^{-f(\boldsymbol{\alpha}) m^{1/d}} \qquad \text{[Babuška, Nobile, Tempone '06]}$$

$$\left\| u - u_L^{\text{SG}} \right\|_{L^\infty} \lesssim C e^{-f(\boldsymbol{\alpha}) m^{1/\log(d)}} \qquad \text{[Nobile, Tempone, W. '08]}$$

Generalized sparse grid interpolation

How to choose the index set $g(\boldsymbol{l}) \leq L$?

- Can build sparse grids corresponding to any polynomial space $\mathbb{P}_{\Lambda_L}(\mathcal{U})$
  - Tensor product (TP): $m(l) = l,\ g(\boldsymbol{l}) = \max_n \alpha_n(l_n - 1) \leq L$
  - Total Degree (TD): $m(l) = l,\ g(\boldsymbol{l}) = \sum_n \alpha_n(l_n - 1) \leq L$
  - Hyperbolic Cross (HC): $m(l) = l,\ g(\boldsymbol{l}) = \prod_n (l_n)^{\alpha_n} \leq L + 1$
  - Smolyak (SM): $m(l) = 2^{l-1} + 1,\ l > 1,\ g(\boldsymbol{l}) = \sum_n \alpha_n(l_n - 1) \leq L$
- The corresponding anisotropic versions are straightforward
- SM is the most widely used approach and corresponds to the original Smolyak construction [Smolyak '63]

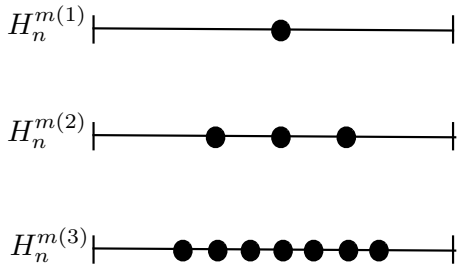
$$\left\| u - u_L^{\text{TP}} \right\|_{L^\infty} \lesssim C e^{-f(\boldsymbol{\alpha}) m^{1/d}} \qquad \text{[Babuška, Nobile, Tempone '06]}$$

$$\left\| u - u_L^{\text{SG}} \right\|_{L^\infty} \lesssim C e^{-f(\boldsymbol{\alpha}) m^{1/\log(d)}} \qquad \text{[Nobile, Tempone, W. '08]}$$

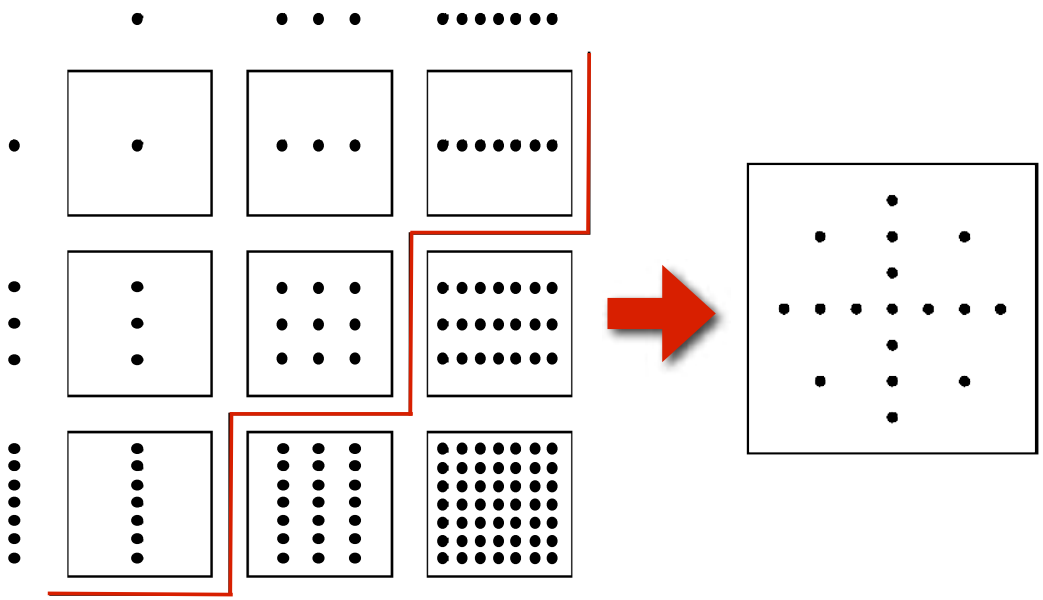
Example: $d = 2$ isotropic sparse grid

Nested rules minimize the amount of work

Nested equidistant grids with $m(l) = 1, 3$ and 7:



Grids $H_1^{m(l_1)} \otimes H_2^{m(l_1)}$ for $l_1, l_2 \leq p = 3$ and the *isotropic* sparse grid $4 = p + 1 \leq l_1 + l_2 \leq p + 2 = 5$, e.g. $(3,1) + (1,3) + (2,2) + (2,3) + (3,2)$

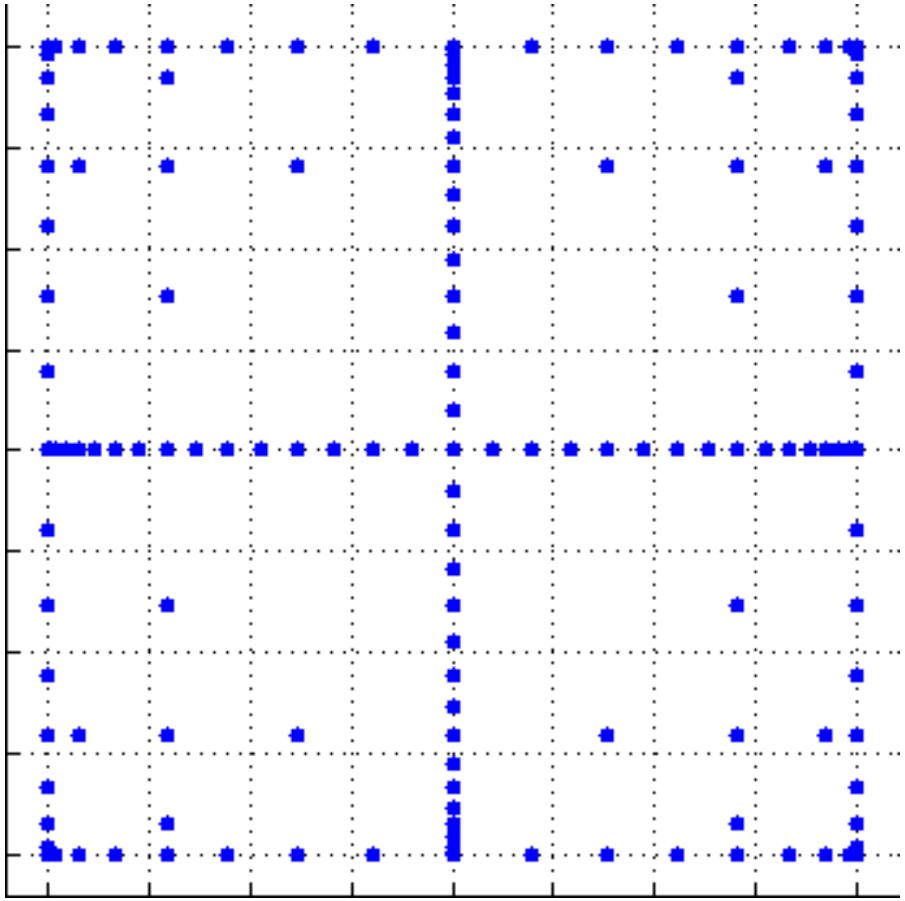


Generating Smolyak sparse grids: $d = 2$

Using Clenshaw-Curtis abscissas

$d = 2$ isotropic sparse grid: $L = 5$

$\Rightarrow |\boldsymbol{l}|_1 \leq L + d = 7$

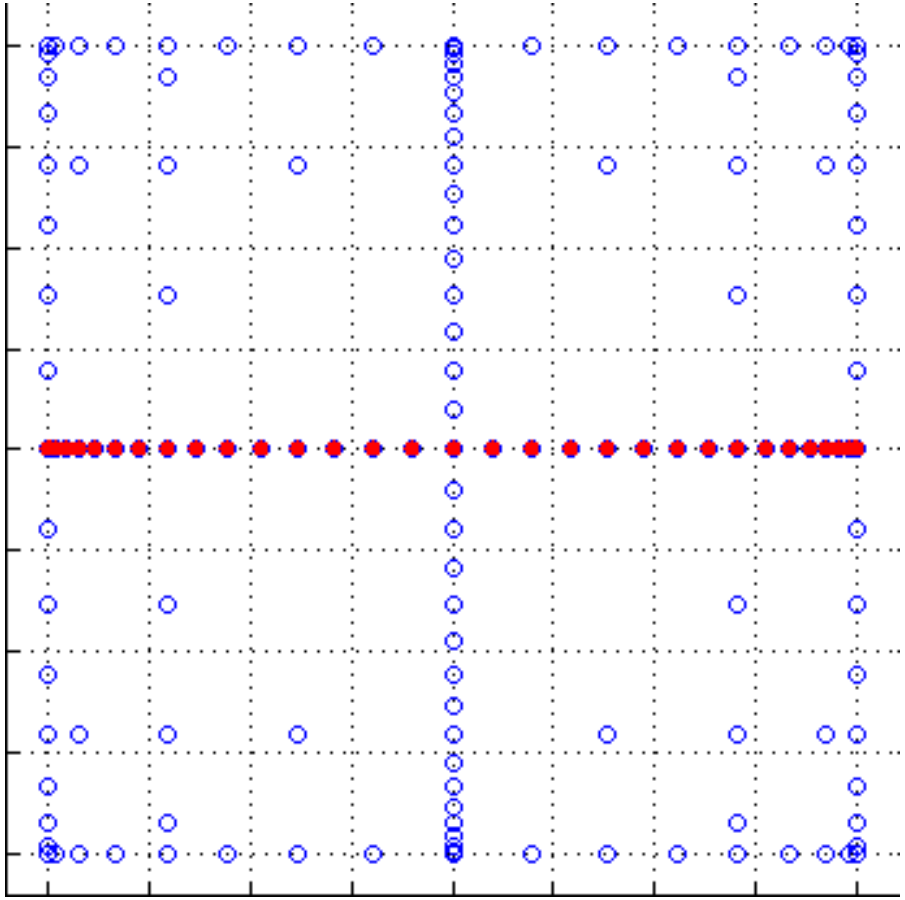


Generating Smolyak sparse grids: $d = 2$

Using Clenshaw-Curtis abscissas

$d = 2$ isotropic sparse grid: $L = 5$

$\Rightarrow |\boldsymbol{l}|_1 \leq L + d = 7$



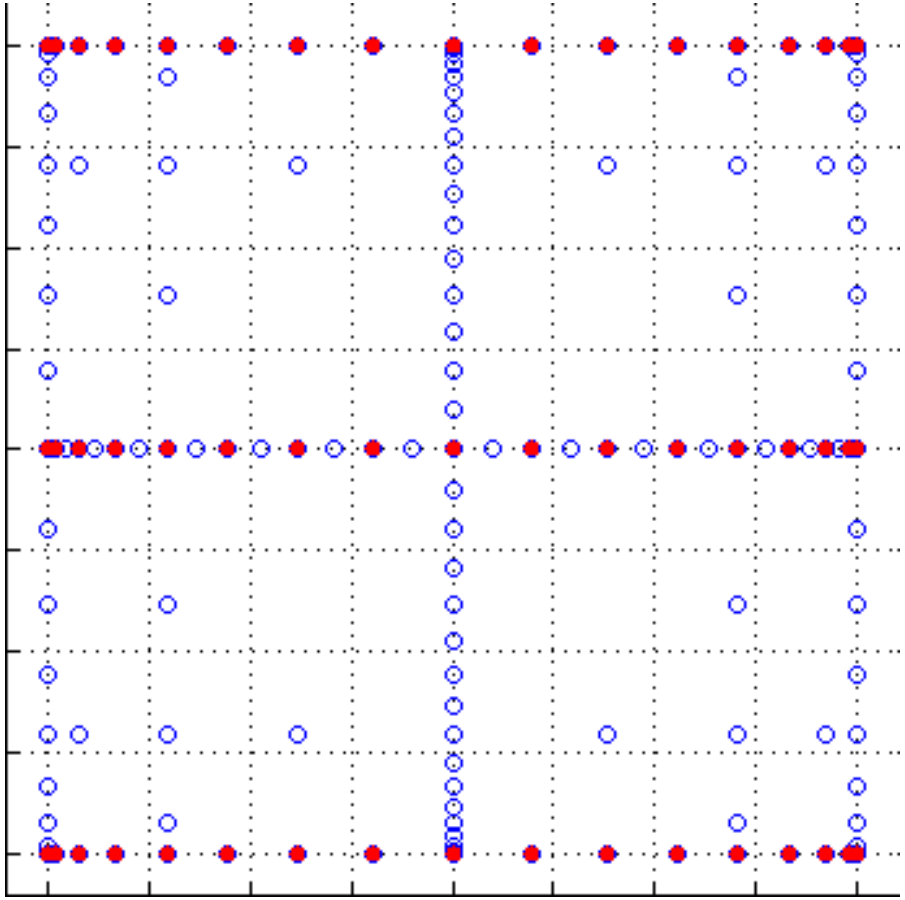
$\boldsymbol{l} = (6, 1) \Rightarrow (33 \times 1)$

Generating Smolyak sparse grids: $d = 2$

Using Clenshaw-Curtis abscissas

$d = 2$ isotropic sparse grid: $L = 5$

$\Rightarrow |\boldsymbol{l}|_1 \leq L + d = 7$



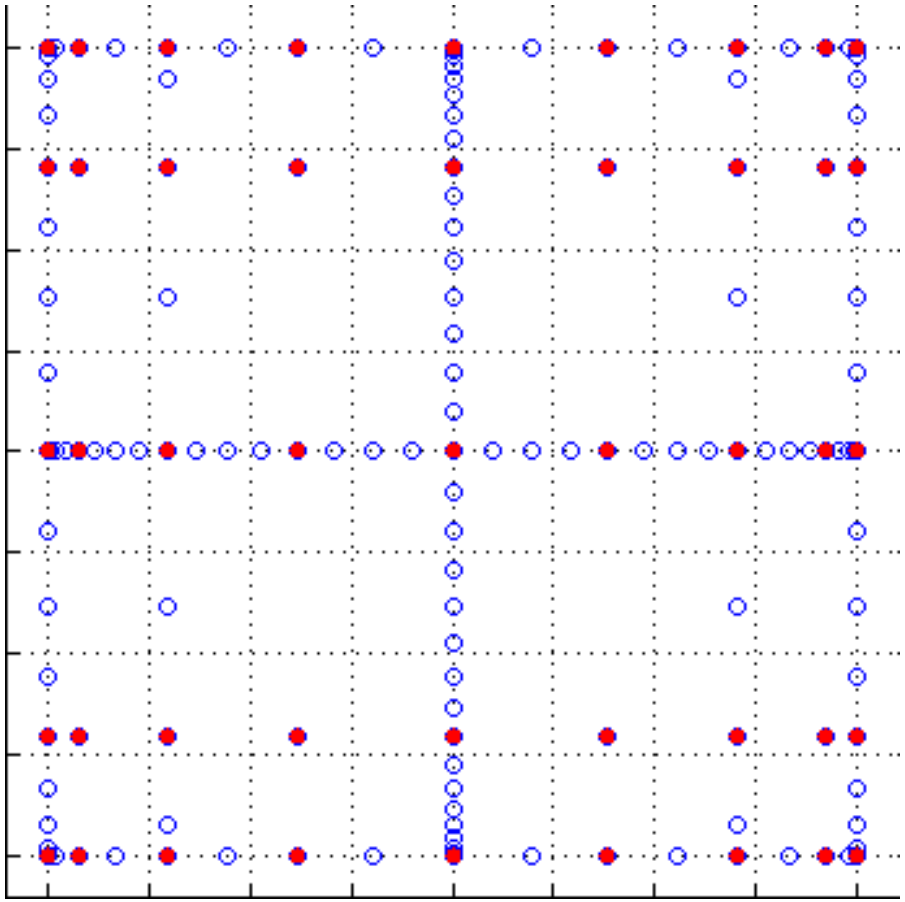
$\boldsymbol{l} = (5, 2) \Rightarrow (17 \times 3)$

Generating Smolyak sparse grids: $d = 2$

Using Clenshaw-Curtis abscissas

$d = 2$ isotropic sparse grid: $L = 5$

$\Rightarrow |\boldsymbol{l}|_1 \leq L + d = 7$



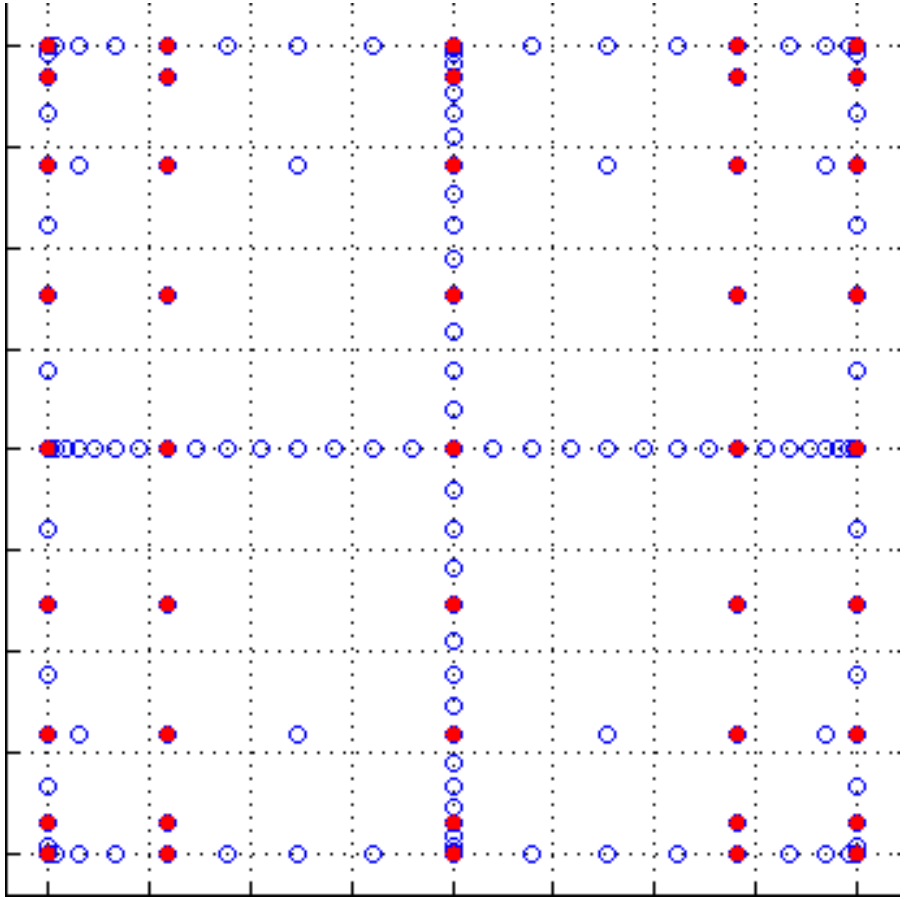
$\boldsymbol{l} = (4, 3) \Rightarrow (9 \times 5)$

Generating Smolyak sparse grids: $d = 2$

Using Clenshaw-Curtis abscissas

$d = 2$ isotropic sparse grid: $L = 5$

$\Rightarrow |\boldsymbol{l}|_1 \leq L + d = 7$



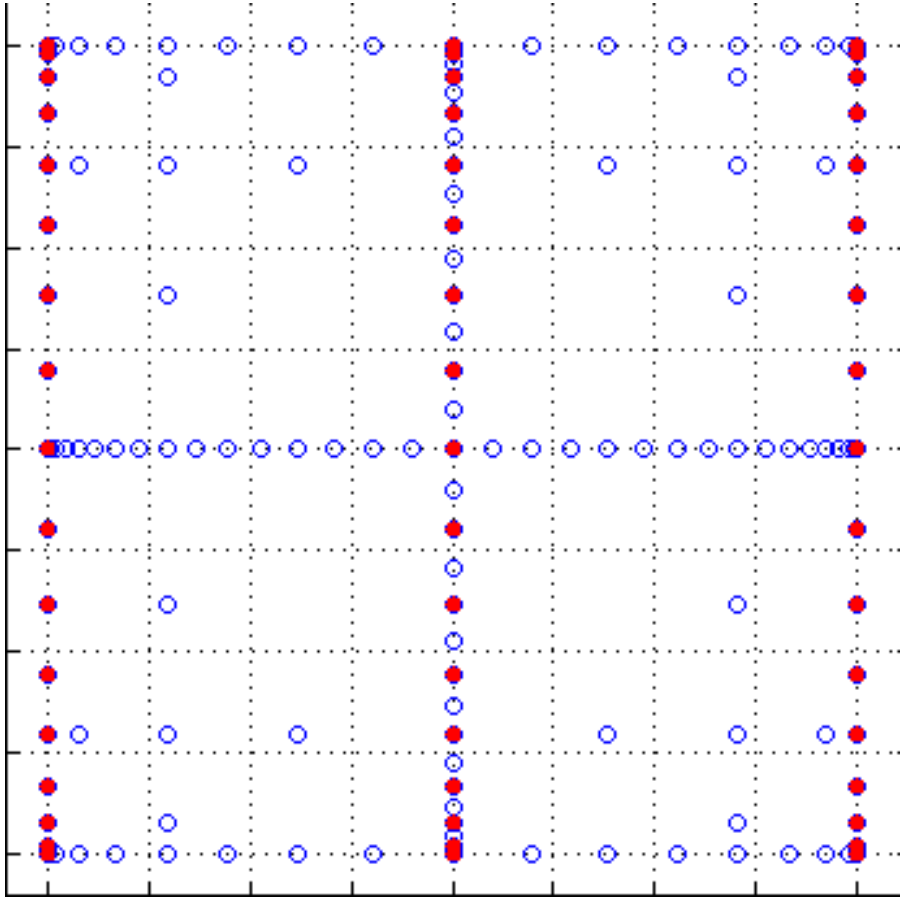
$\boldsymbol{l} = (3, 4) \Rightarrow (5 \times 9)$

Generating Smolyak sparse grids: $d = 2$

Using Clenshaw-Curtis abscissas

$d = 2$ isotropic sparse grid: $L = 5$

$\Rightarrow |\boldsymbol{l}|_1 \leq L + d = 7$



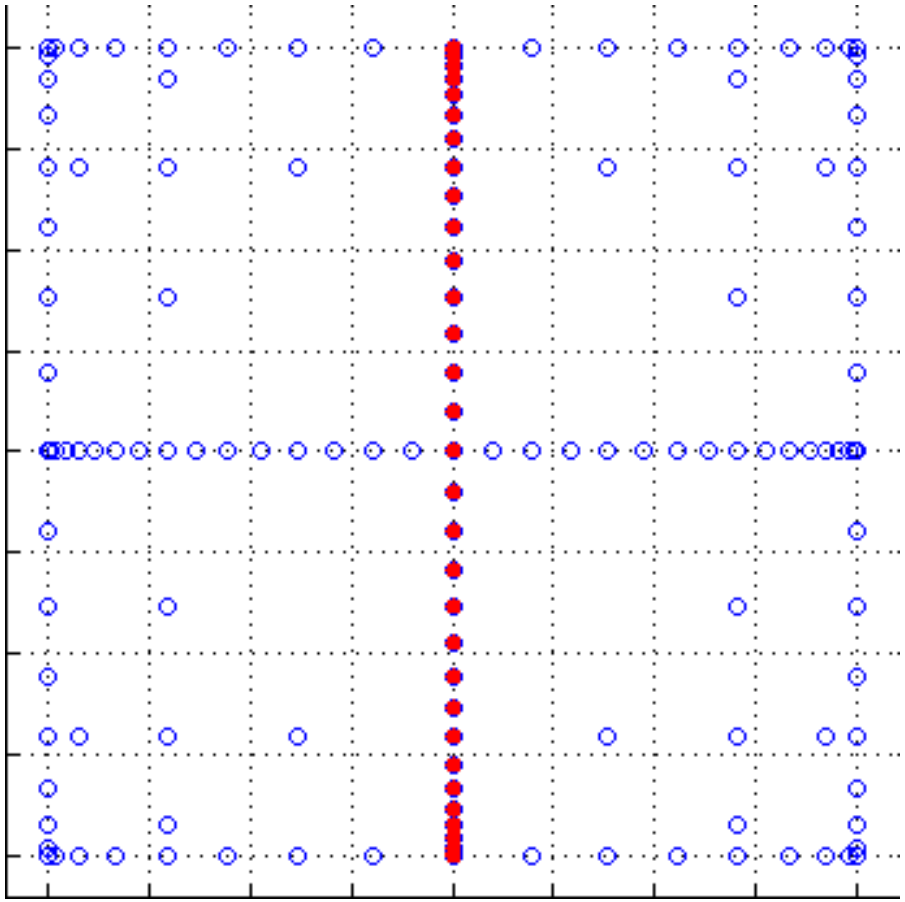
$\boldsymbol{l} = (2, 5) \Rightarrow (3 \times 17)$

Generating Smolyak sparse grids: $d = 2$

Using Clenshaw-Curtis abscissas

$d = 2$ isotropic sparse grid: $L = 5$

$\Rightarrow |\boldsymbol{l}|_1 \leq L + d = 7$



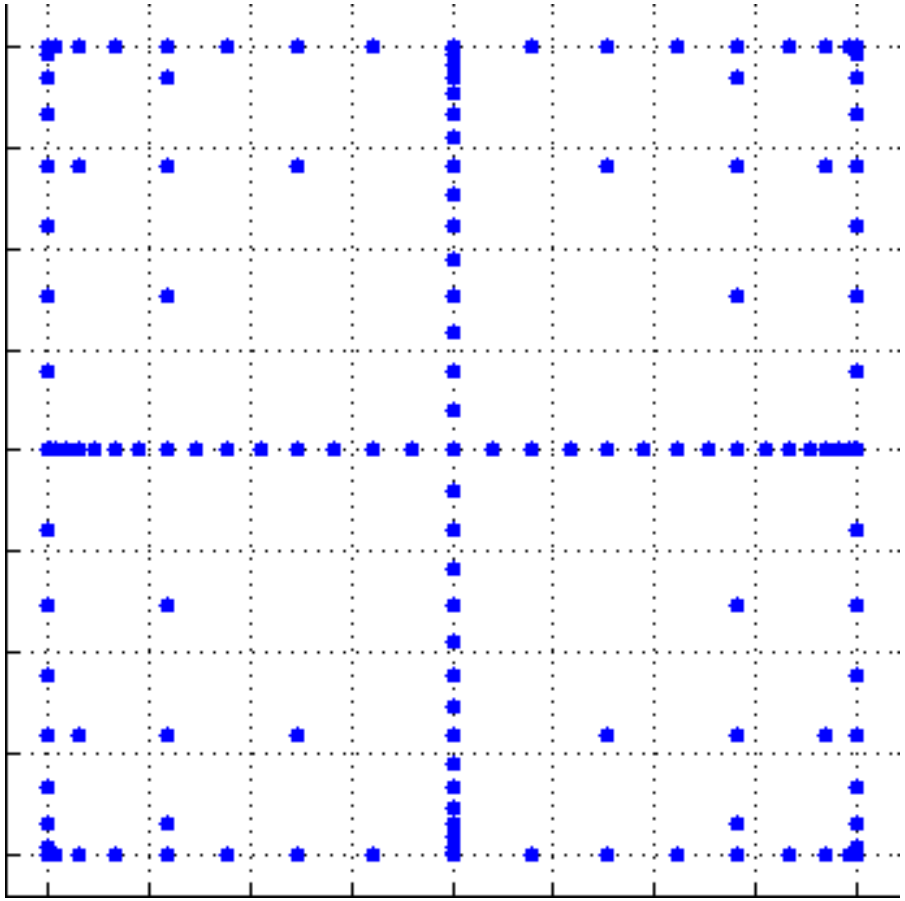
$\boldsymbol{l} = (1, 6) \Rightarrow (1 \times 33)$

Generating Smolyak sparse grids: $d = 2$

Using Clenshaw-Curtis abscissas

$d = 2$ isotropic sparse grid: $L = 5$

$\Rightarrow |\boldsymbol{l}|_1 \leq L + d = 7$

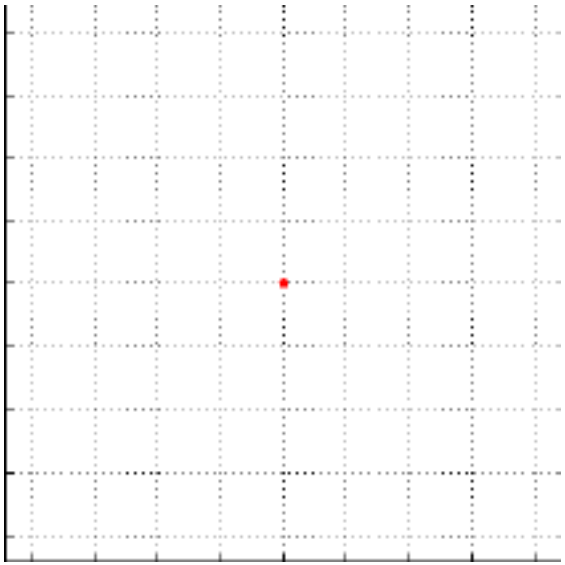


Dimension-adaptive sparse grids

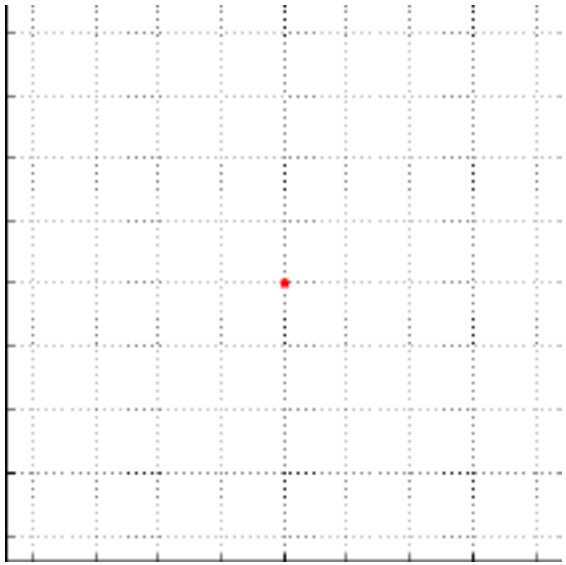
Anisotropic methods [Nobile, Tempone, W. 2008]

Let $g(\boldsymbol{l}) = \sum_{n=1}^{d} \alpha_n (l_n - 1)$, and $m(l) = \begin{cases} 1, & l = 1 \\ 2^{l+1} - 1, & l > 1 \end{cases}$

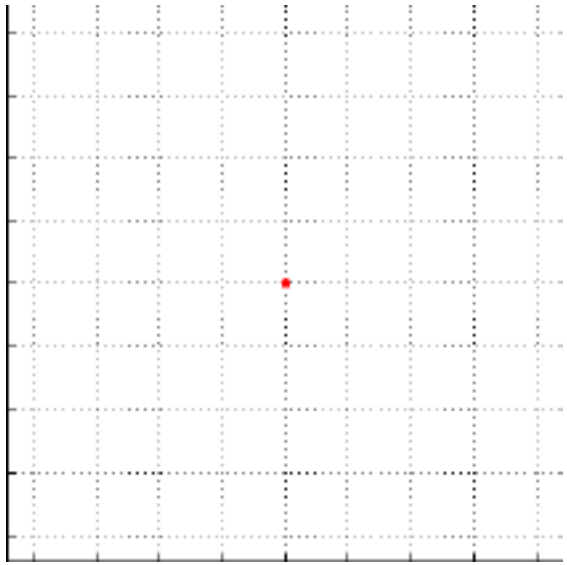
$d = 2$ anisotropic sparse grid: $g(\mathbf{i}) \leq L = 0$



$\alpha_2/\alpha_1 = 1$



$\alpha_2/\alpha_1 = 1.5$



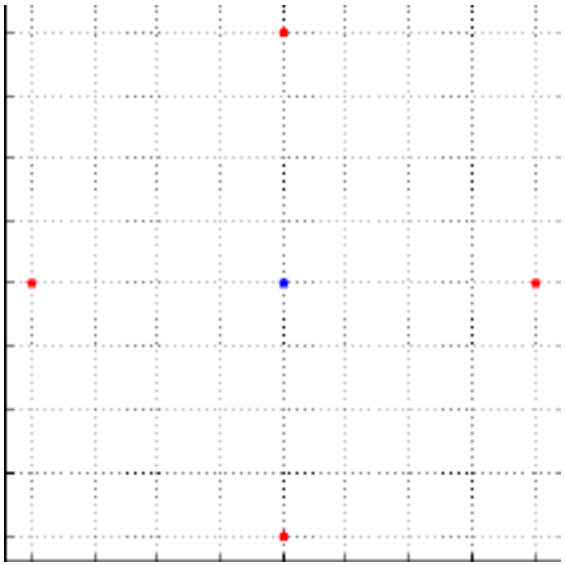
$\alpha_2/\alpha_1 = 2$

Dimension-adaptive sparse grids

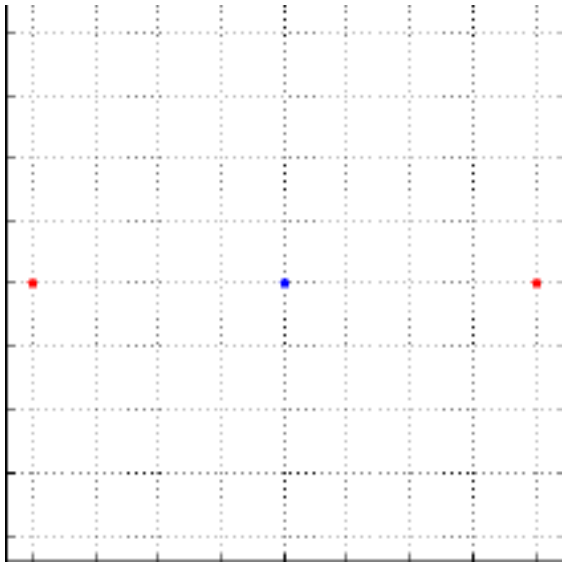
Anisotropic methods [Nobile, Tempone, W. 2008]

Let $g(\boldsymbol{l}) = \sum_{n=1}^{d} \alpha_n (l_n - 1)$, and $m(l) = \begin{cases} 1, & l = 1 \\ 2^{l+1} - 1, & l > 1 \end{cases}$

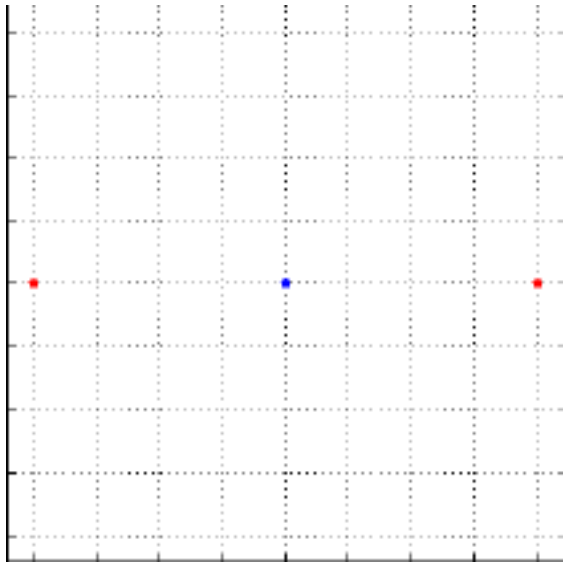
$d = 2$ anisotropic sparse grid: $g(\mathbf{i}) \leq L = 1$



$\alpha_2/\alpha_1 = 1$



$\alpha_2/\alpha_1 = 1.5$



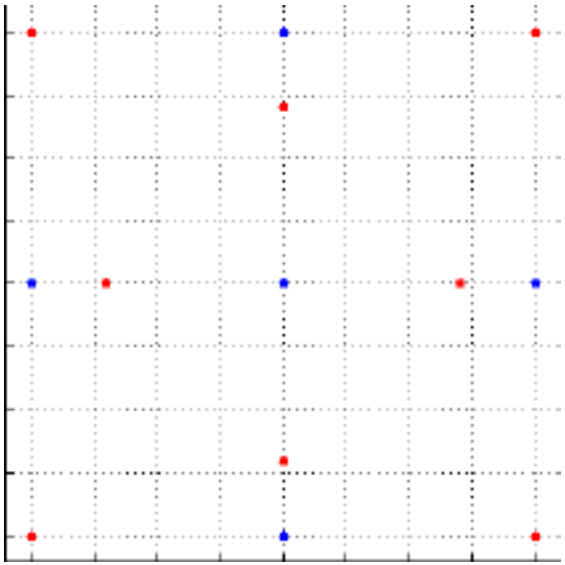
$\alpha_2/\alpha_1 = 2$

Dimension-adaptive sparse grids

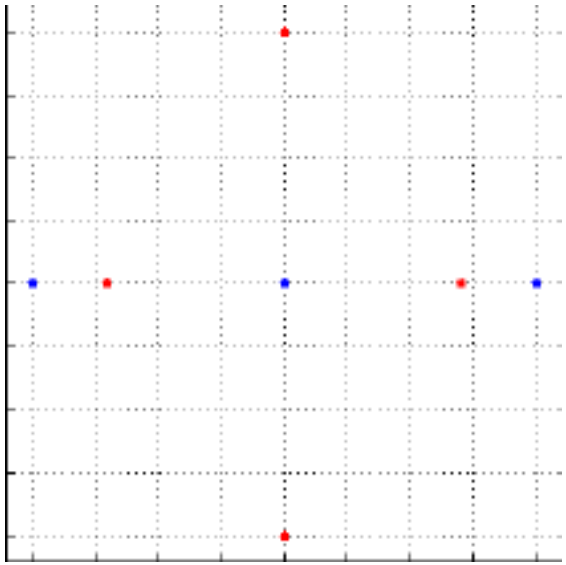
Anisotropic methods [Nobile, Tempone, W. 2008]

Let $g(\boldsymbol{l}) = \sum_{n=1}^{d} \alpha_n (l_n - 1)$, and $m(l) = \begin{cases} 1, & l = 1 \\ 2^{l+1} - 1, & l > 1 \end{cases}$

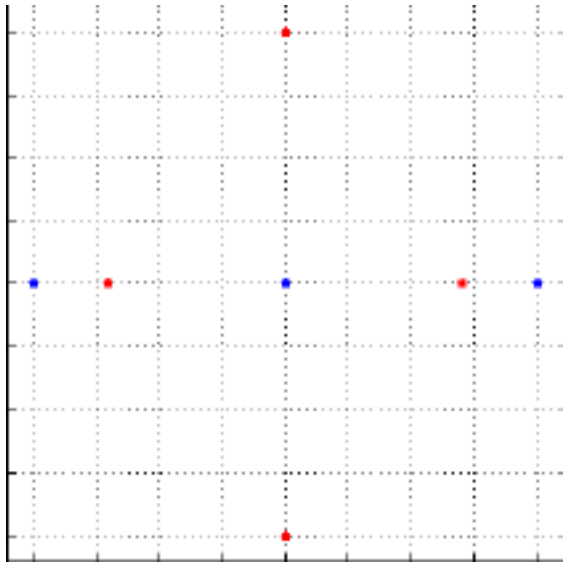
$d = 2$ anisotropic sparse grid: $g(\mathbf{i}) \leq L = 2$



$\alpha_2/\alpha_1 = 1$



$\alpha_2/\alpha_1 = 1.5$



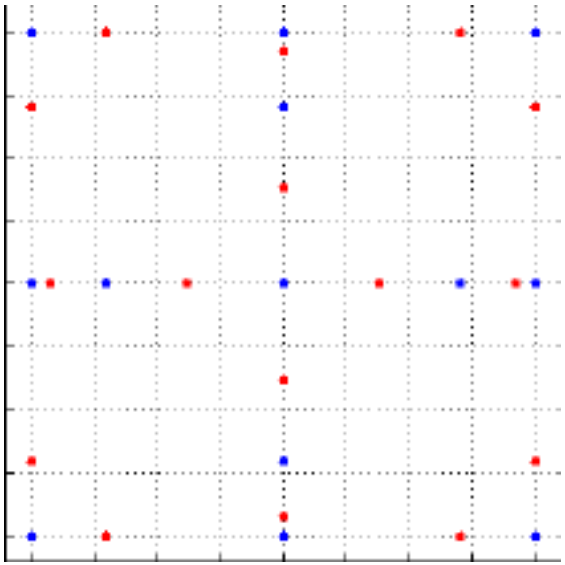
$\alpha_2/\alpha_1 = 2$

Dimension-adaptive sparse grids

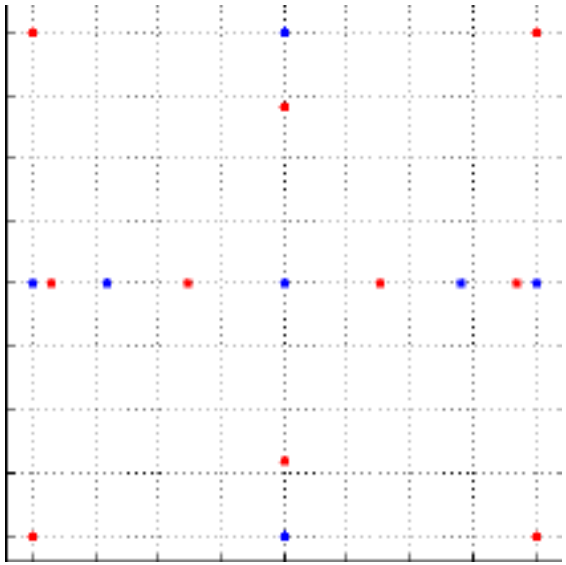
Anisotropic methods [Nobile, Tempone, W. 2008]

Let $g(\boldsymbol{l}) = \sum_{n=1}^{d} \alpha_n (l_n - 1)$, and $m(l) = \begin{cases} 1, & l = 1 \\ 2^{l+1} - 1, & l > 1 \end{cases}$

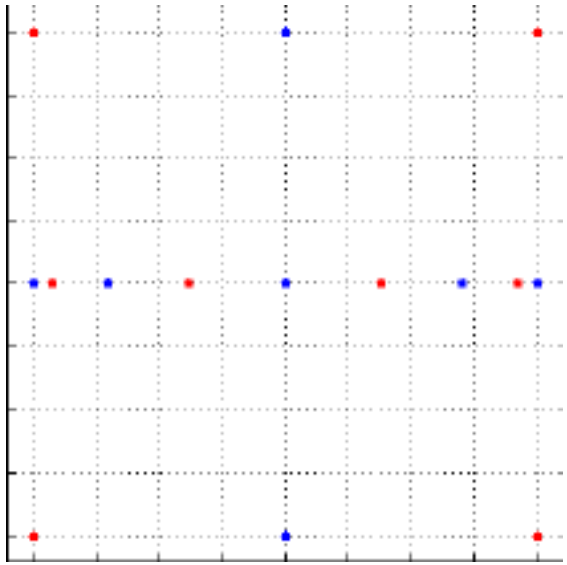
$d = 2$ anisotropic sparse grid: $g(\mathbf{i}) \leq L = 3$



$\alpha_2/\alpha_1 = 1$



$\alpha_2/\alpha_1 = 1.5$



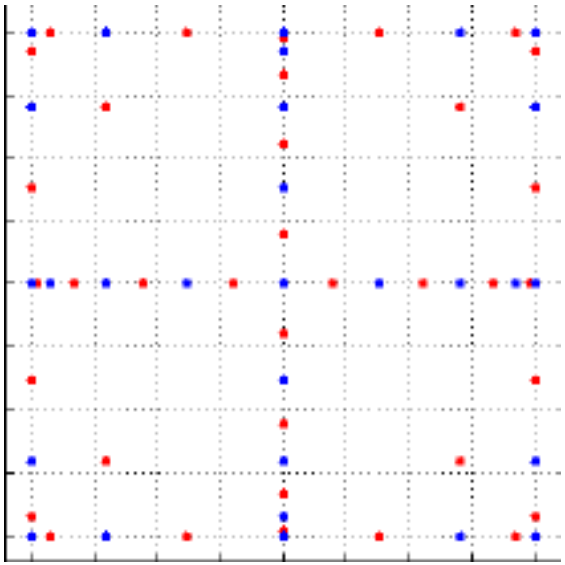
$\alpha_2/\alpha_1 = 2$

Dimension-adaptive sparse grids

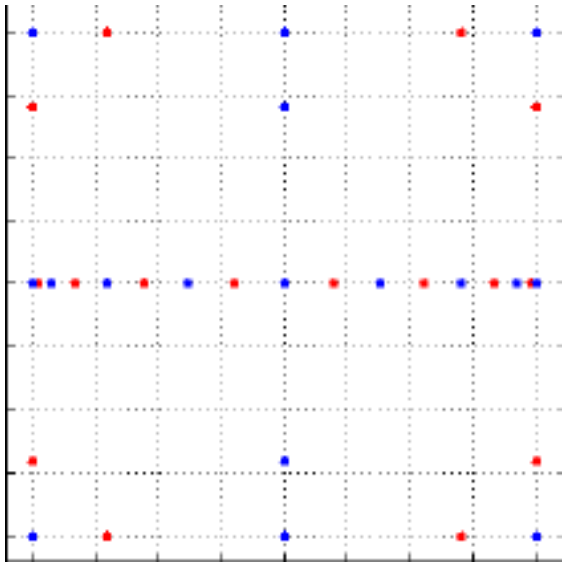
Anisotropic methods [Nobile, Tempone, W. 2008]

Let $g(\boldsymbol{l}) = \sum_{n=1}^{d} \alpha_n (l_n - 1)$, and $m(l) = \begin{cases} 1, & l = 1 \\ 2^{l+1} - 1, & l > 1 \end{cases}$

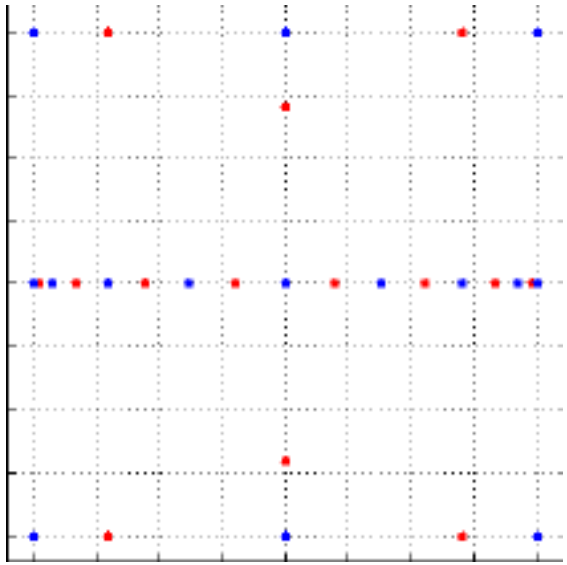
$d = 2$ anisotropic sparse grid: $g(\mathbf{i}) \leq L = 4$



$\alpha_2/\alpha_1 = 1$



$\alpha_2/\alpha_1 = 1.5$



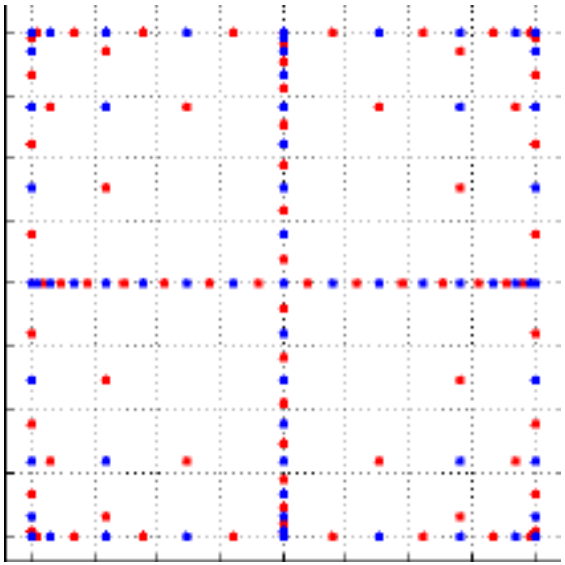
$\alpha_2/\alpha_1 = 2$

Dimension-adaptive sparse grids

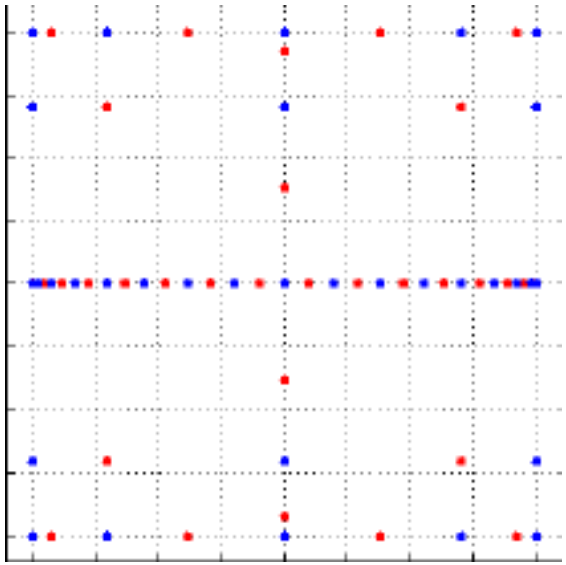
Anisotropic methods [Nobile, Tempone, W. 2008]

Let $g(\boldsymbol{l}) = \sum_{n=1}^{d} \alpha_n (l_n - 1)$, and $m(l) = \begin{cases} 1, & l = 1 \\ 2^{l+1} - 1, & l > 1 \end{cases}$

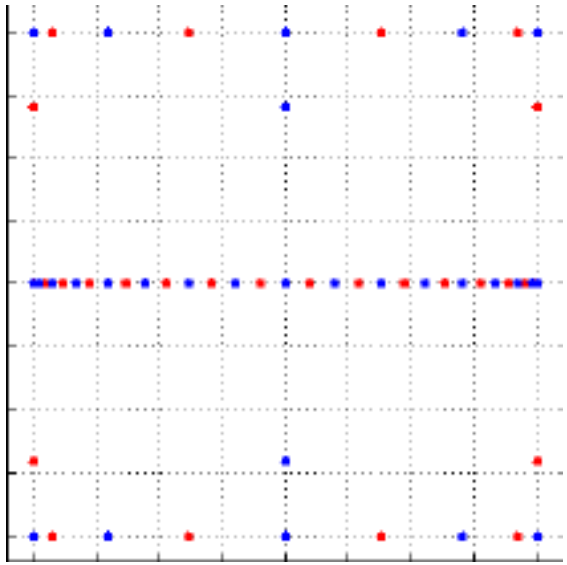
$d = 2$ anisotropic sparse grid: $g(\mathbf{i}) \leq L = 5$



$\alpha_2/\alpha_1 = 1$



$\alpha_2/\alpha_1 = 1.5$



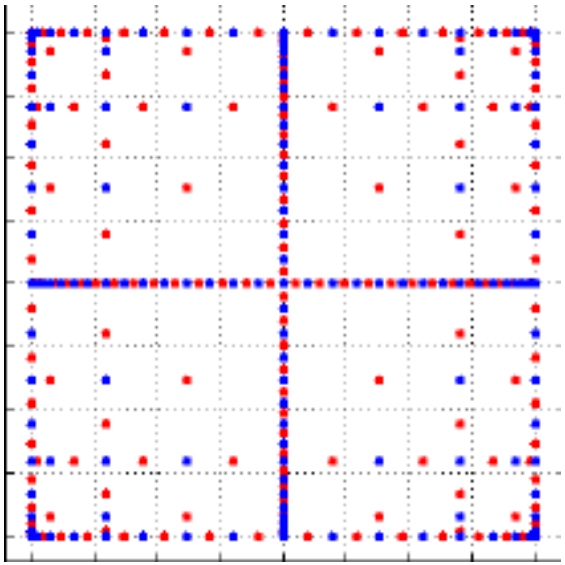
$\alpha_2/\alpha_1 = 2$

Dimension-adaptive sparse grids

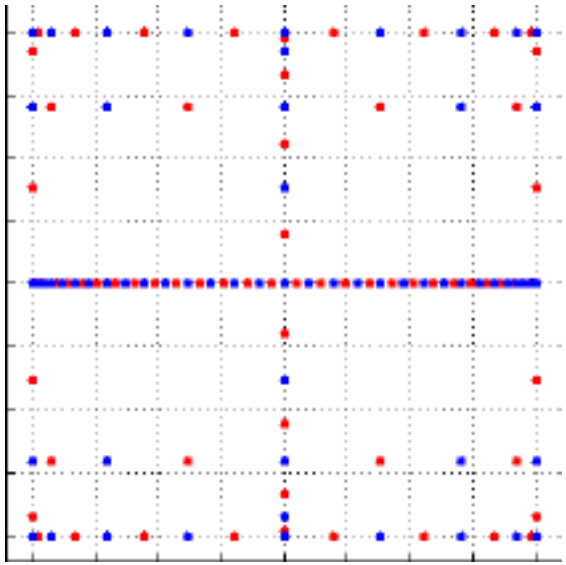
Anisotropic methods [Nobile, Tempone, W. 2008]

Let $g(\boldsymbol{l}) = \sum_{n=1}^{d} \alpha_n (l_n - 1)$, and $m(l) = \begin{cases} 1, & l = 1 \\ 2^{l+1} - 1, & l > 1 \end{cases}$

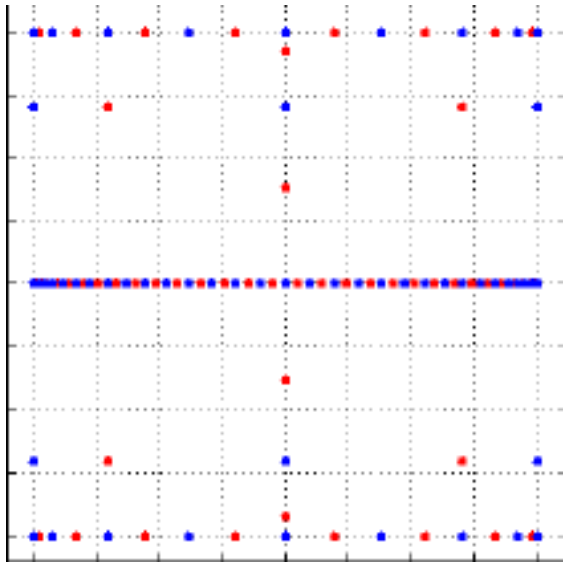
$d = 2$ anisotropic sparse grid: $g(\mathbf{i}) \leq L = 6$



$\alpha_2/\alpha_1 = 1$



$\alpha_2/\alpha_1 = 1.5$



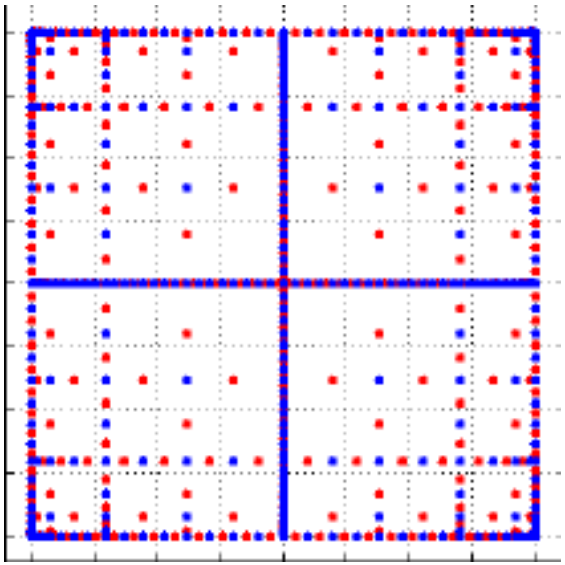
$\alpha_2/\alpha_1 = 2$

Dimension-adaptive sparse grids

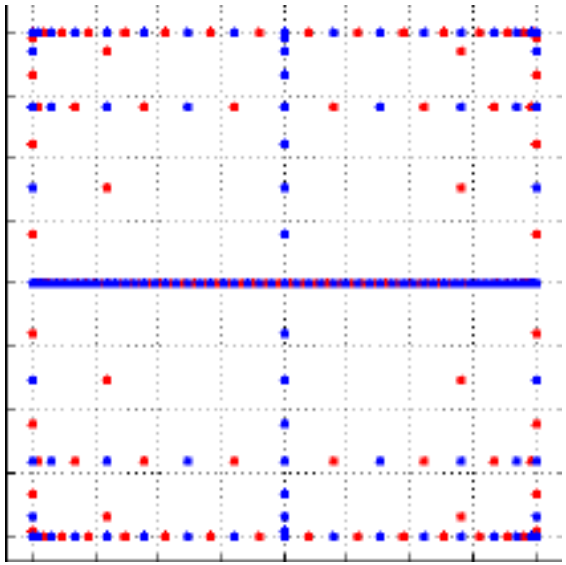
Anisotropic methods [Nobile, Tempone, W. 2008]

Let $g(\boldsymbol{l}) = \sum_{n=1}^{d} \alpha_n (l_n - 1)$, and $m(l) = \begin{cases} 1, & l = 1 \\ 2^{l+1} - 1, & l > 1 \end{cases}$

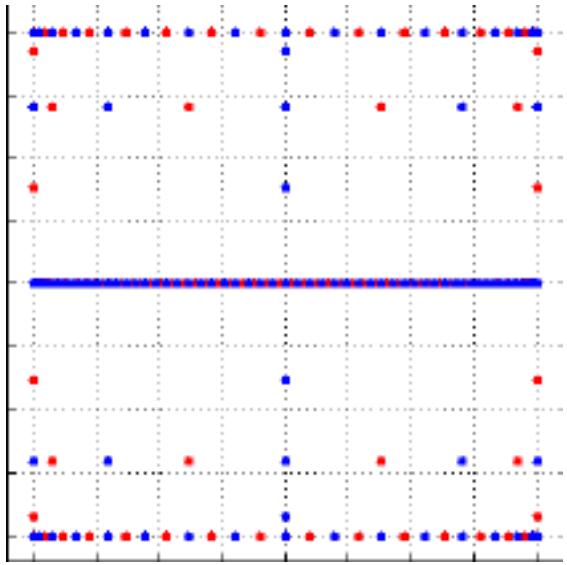
$d = 2$ anisotropic sparse grid: $g(\mathbf{i}) \leq L = 7$



$\alpha_2/\alpha_1 = 1$



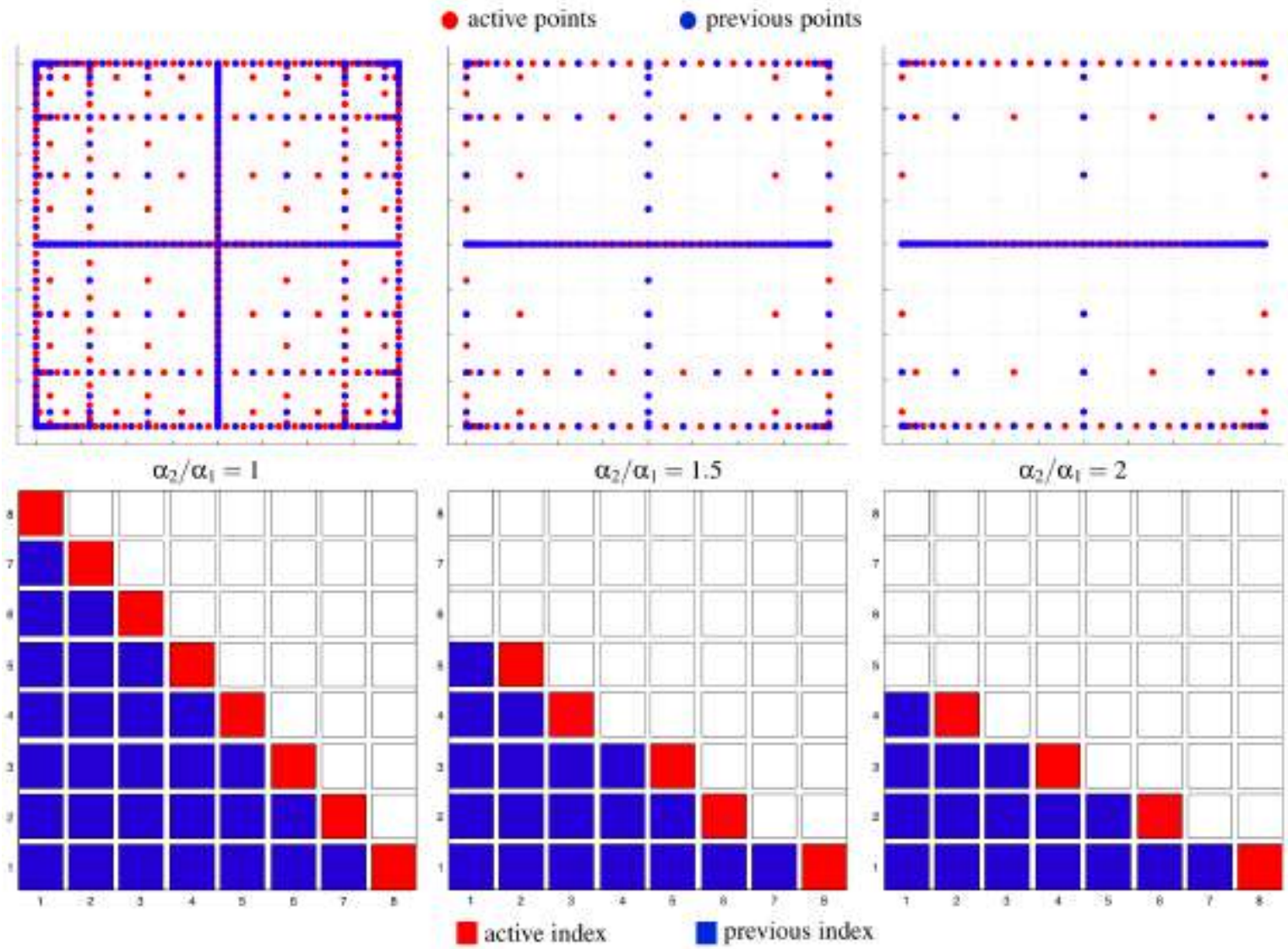
$\alpha_2/\alpha_1 = 1.5$



$\alpha_2/\alpha_1 = 2$

Anisotropic Clenshaw-Curtis sparse grids

Correspondng indices (l_1, l_2) s.t. $\alpha_1 l_1 + \alpha_2 l_2 \leq 7$



Level L=5 sparse grids

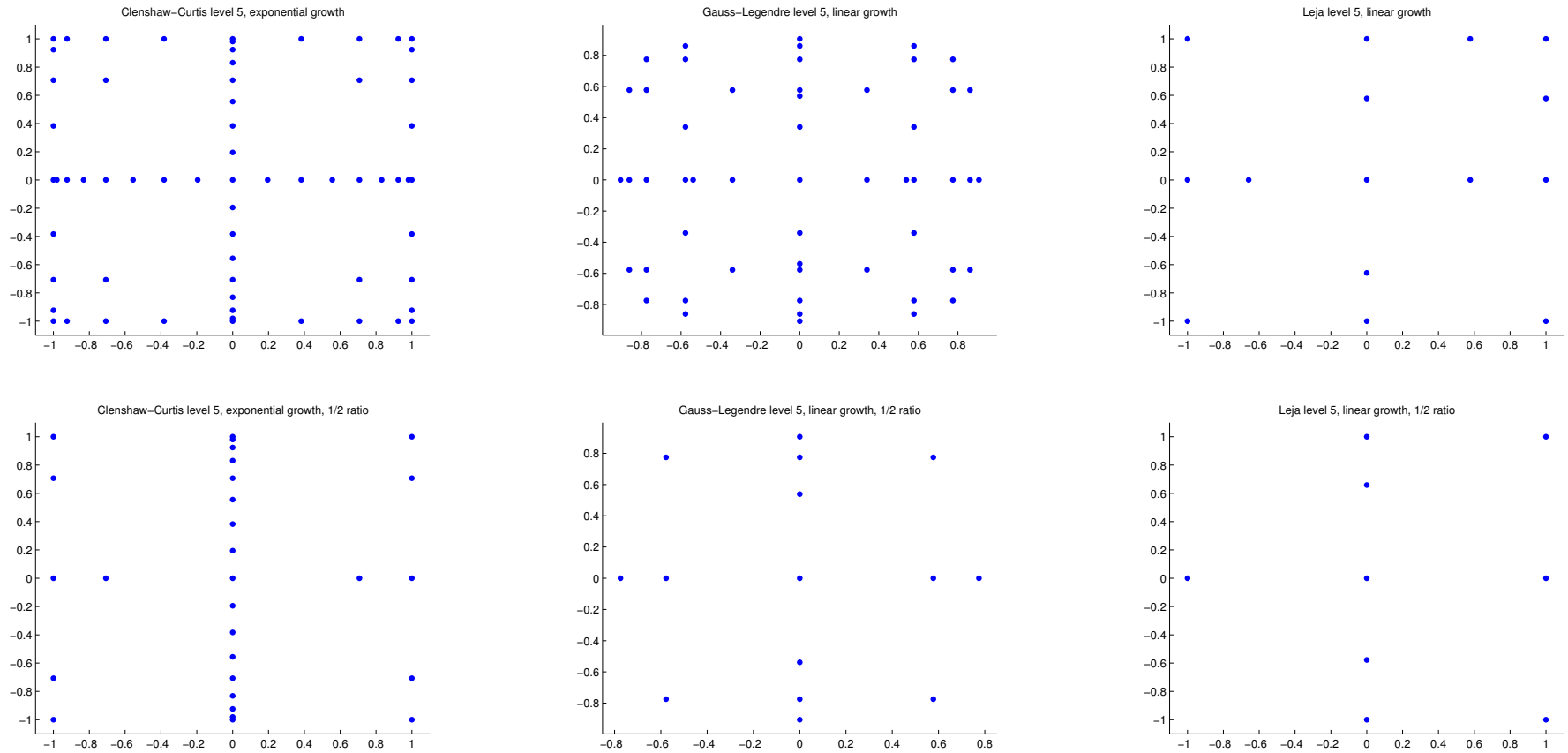


Figure $d = 2$ and level $L = 5$: **(top)**: the isotropic sparse grid with: Clenshaw-Curtis abscissas with exponential growth, Gauss-Legendre abscissas with linear growth, and the Leja abscissas with linear growth. **(bottom)**: the corresponding anisotropic versions with $\alpha_2/\alpha_1 = 2$.

Numerical example

Sparse grid stochastic collocation FEM

We let $\mathbf{x} = (x_1, x_2)$ and consider the following parameterized stochastic elliptic PDE:

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

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

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

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

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

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

Numerical example

Sparse grid stochastic collocation FEM

We let $\mathbf{x} = (x_1, x_2)$ and consider the following parameterized stochastic elliptic PDE:

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

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

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

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

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

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

Calculating the weighting parameters

A priori selection: $d = 11$

A priori selection of the dimension weights α_n:

$$\alpha_n = \log\left(\frac{2\varrho_n}{|\mathcal{U}_n|} + \sqrt{1 + \frac{4\varrho_n{}^2}{|\mathcal{U}_n|^2}}\right) \quad \text{and} \quad \varrho_n = \frac{1}{12\sqrt{\lambda_n}\|b_n\|_{L^\infty(D)}}$$

For this problem we have

$$\alpha_n = \begin{cases} \log\left(1 + c/\sqrt{R}\right), & \text{for } n << R^{-2} \\ n^2 R^2, & \text{for } n > R^{-2} \end{cases}$$

	α_1	$\alpha_2,\ \alpha_3$	$\alpha_4,\ \alpha_5$	$\alpha_6,\ \alpha_7$	$\alpha_8,\ \alpha_9$	$\alpha_{10},\ \alpha_{11}$
$R = 1/2$	0.20	0.19	0.42	1.24	3.1	5.8
$R = 1/64$	0.79	0.62	0.62	0.62	0.62	0.62

Goal: $\|\mathbb{E}[\epsilon]\|_{L^2(D)} \approx \|\mathbb{E}\left[u_L^{\mathsf{SG}}(x,\boldsymbol{y}) - u_{L_{\max}+1}^{\mathsf{SG}}(x,\boldsymbol{y})\right]\|_{L^2(D)}$

- $L = 0, 1, \ldots, L_{\max}$ and $u_{L_{\max}+1}$ is an "overkilled solution."

Calculating the weighting parameters

A priori selection: $d = 11$

A priori selection of the dimension weights α_n:

$$\alpha_n = \log\left(\frac{2\varrho_n}{|\mathcal{U}_n|} + \sqrt{1 + \frac{4\varrho_n{}^2}{|\mathcal{U}_n|^2}}\right) \quad \text{and} \quad \varrho_n = \frac{1}{12\sqrt{\lambda_n}\|b_n\|_{L^\infty(D)}}$$

For this problem we have

$$\alpha_n = \begin{cases} \log\left(1 + c/\sqrt{R}\right), & \text{for } n << R^{-2} \\ n^2 R^2, & \text{for } n > R^{-2} \end{cases}$$

	α_1	$\alpha_2,\ \alpha_3$	$\alpha_4,\ \alpha_5$	$\alpha_6,\ \alpha_7$	$\alpha_8,\ \alpha_9$	$\alpha_{10},\ \alpha_{11}$
$R = 1/2$	0.20	0.19	0.42	1.24	3.1	5.8
$R = 1/64$	0.79	0.62	0.62	0.62	0.62	0.62

Goal: $\|\mathbb{E}[\epsilon]\|_{L^2(D)} \approx \|\mathbb{E}\left[u_L^{\mathsf{SG}}(x,\boldsymbol{y}) - u_{L_{\max}+1}^{\mathsf{SG}}(x,\boldsymbol{y})\right]\|_{L^2(D)}$

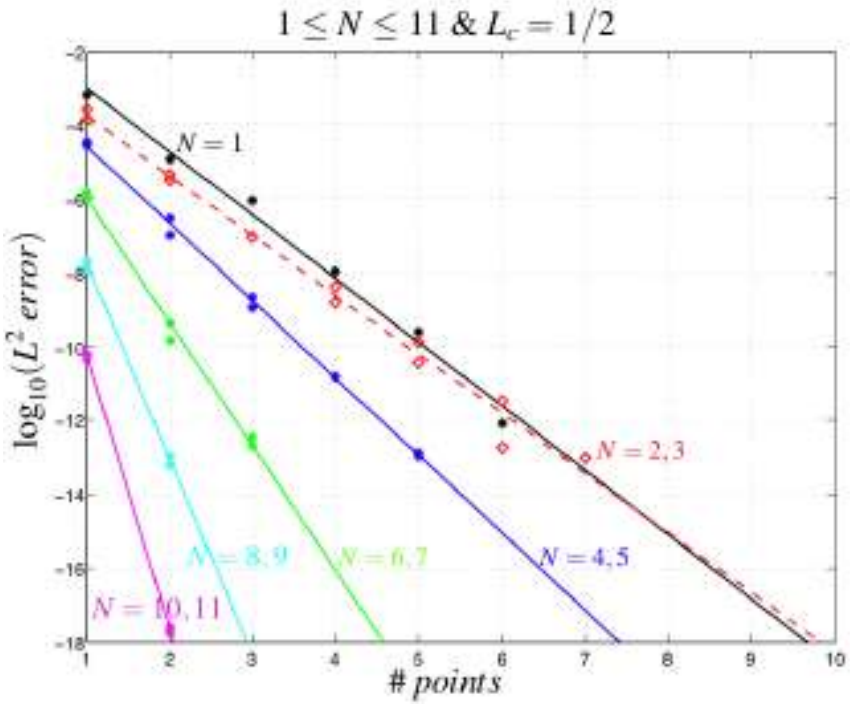
- $L = 0, 1, \ldots, L_{\max}$ and $u_{L_{\max}+1}$ is an "overkilled solution."

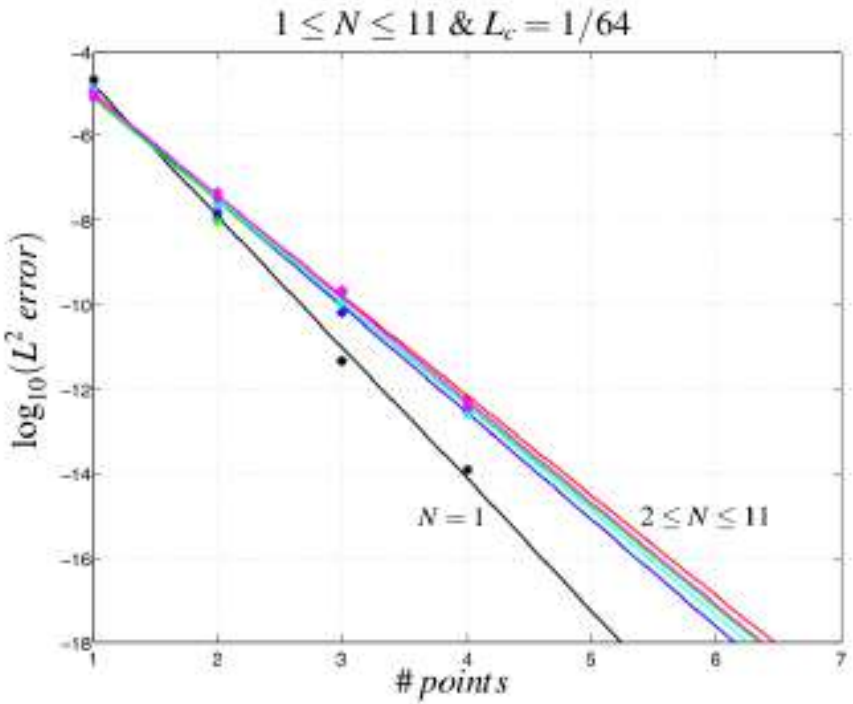
Calculating the weighting parameters

A posteriori selection: $N = 11$

Goal: $\|\mathbb{E}[\epsilon]\|_{L^2(D)} \approx \|\mathbb{E}\left[u_L^{\mathsf{SG}}(x,\boldsymbol{y}) - u_{L_{\mathsf{max}}+1}^{\mathsf{SG}}(x,\boldsymbol{y})\right]\|_{L^2(D)}$

- $L = 0, 1, \ldots, L_{\mathsf{max}}$ and $u_{L_{\mathsf{max}}+1}$ is an "overkilled solution."

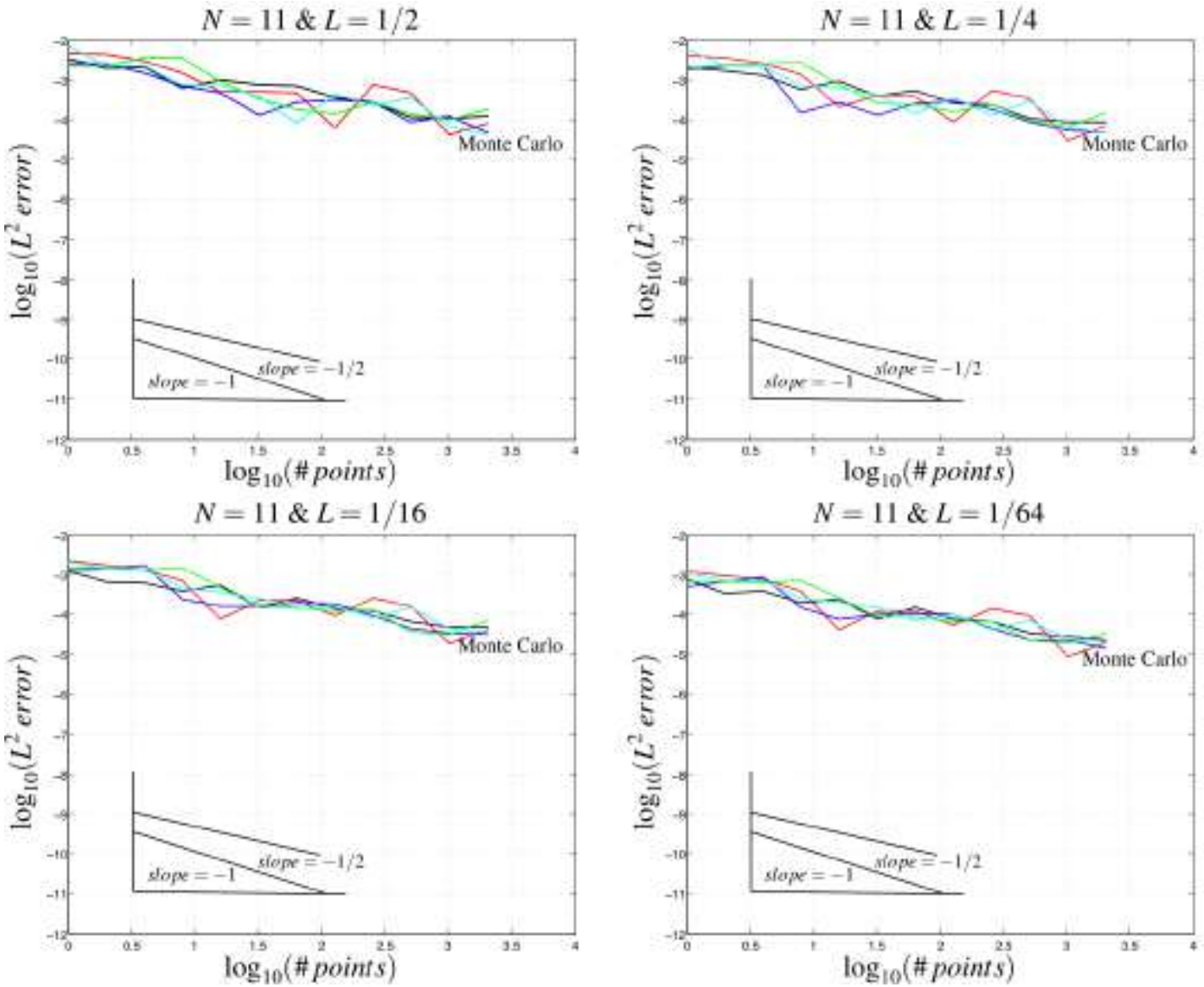




A linear least square approximation to fit $\log_{10}(\|E[\varepsilon_n]\|_{L^2(D)})$ versus l_n. For $n = 1, 2, \ldots, d = 11$ we plot: on the left, the highly anisotropic case $R = 1/2$ and on the right, the isotropic case $R = 1/64$

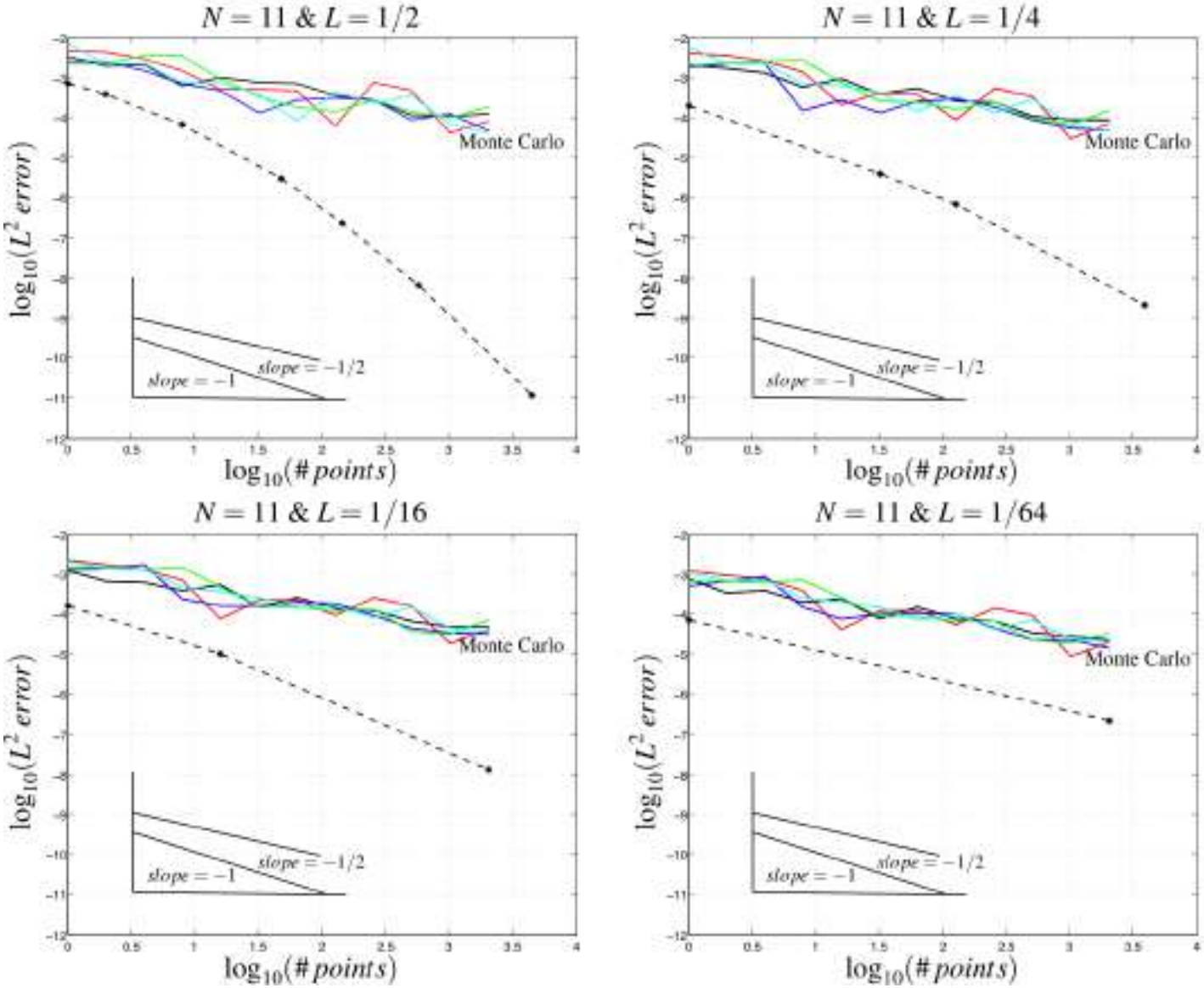
Convergence Comparisons

$d = 11$ random variables



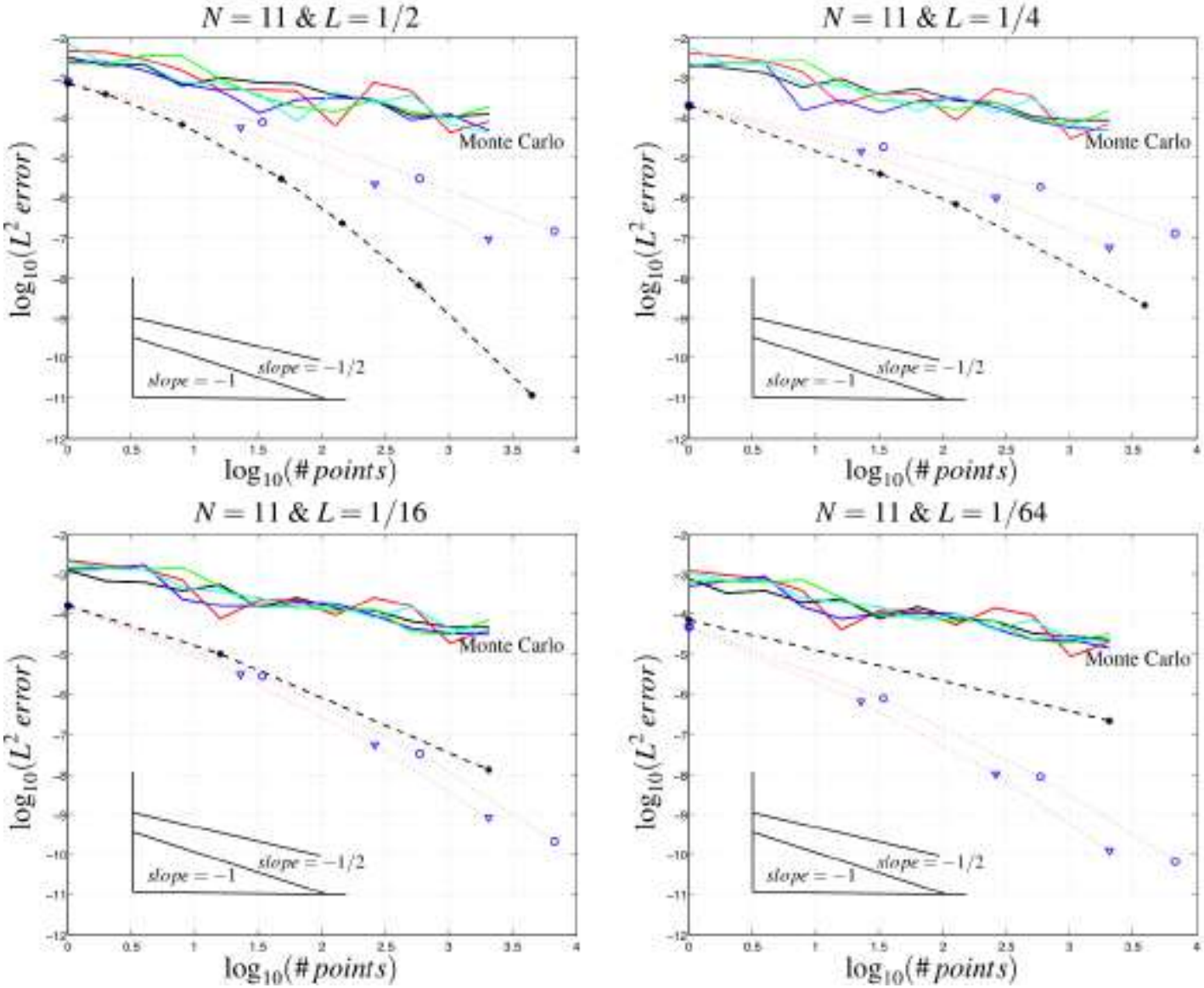
Convergence Comparisons

$d = 11$ random variables



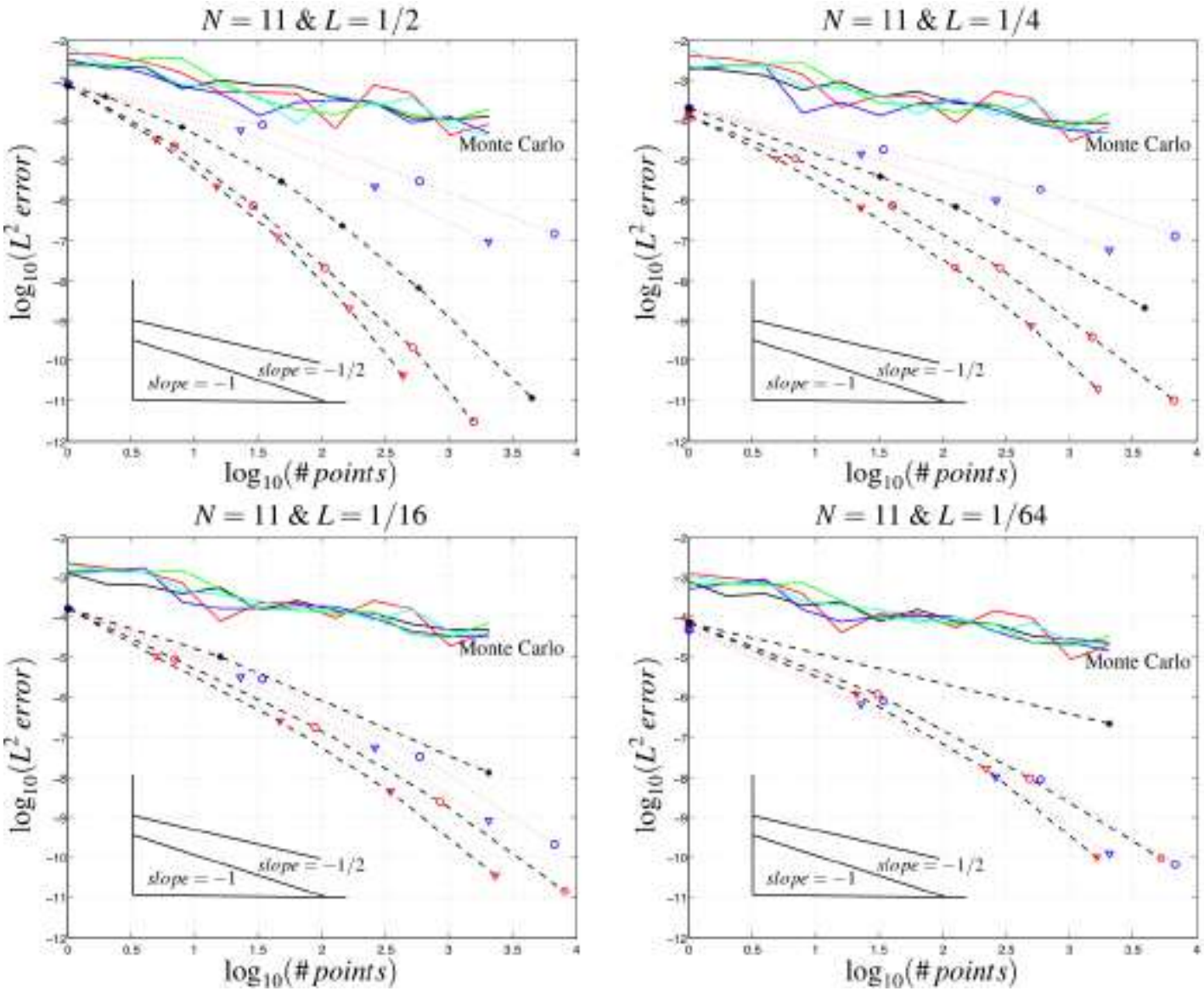
Convergence Comparisons

$d = 11$ random variables



Convergence Comparisons

$d = 11$ random variables



Convergence Comparisons II

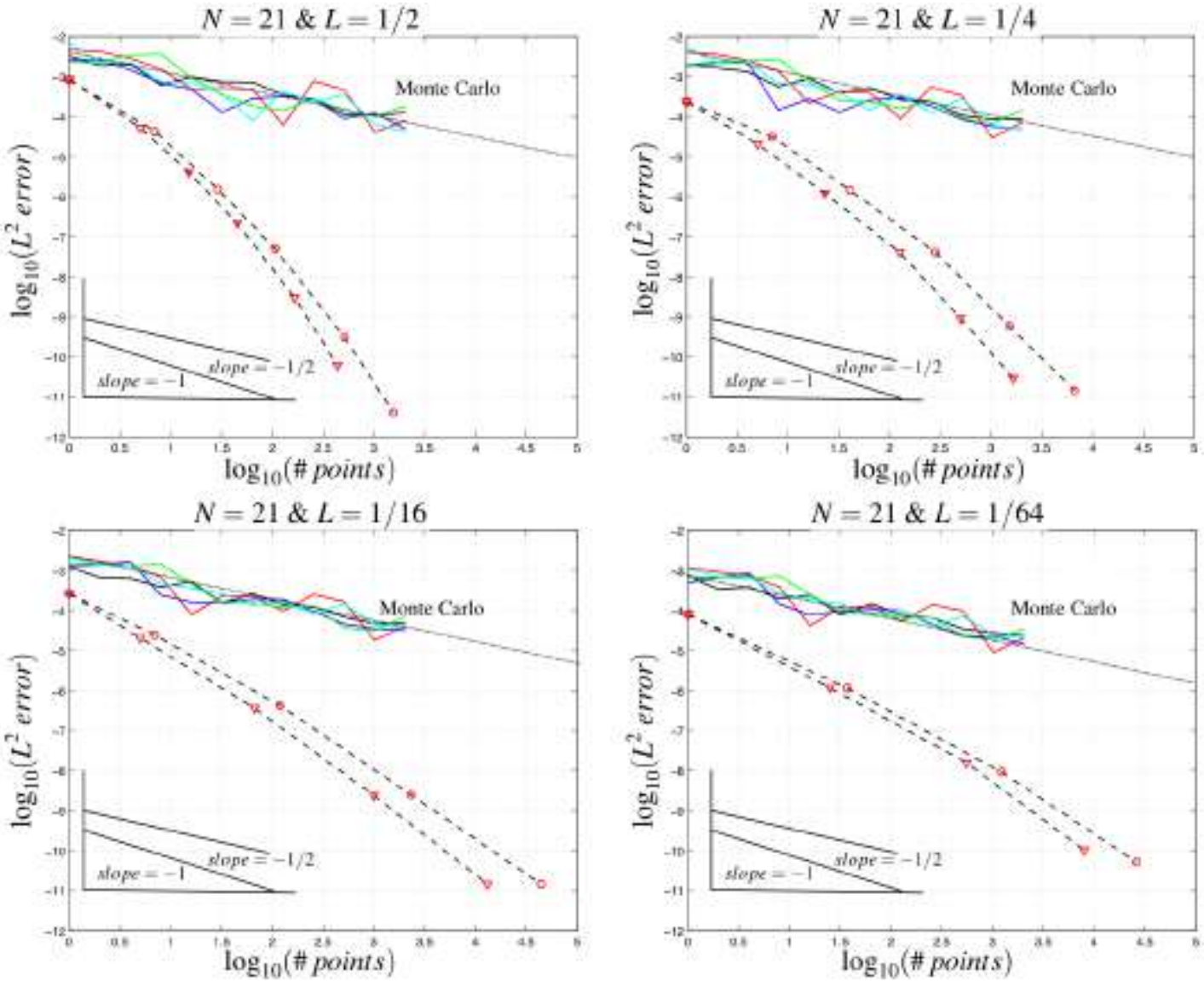
$d = 11$ random variables

R	AS	AF	IS	MC
$1/2$	50	252	2512	$5.0e+09$
$1/4$	158	1259	3981	$2.0e+09$
$1/16$	199	1958	501	$1.6e+09$
$1/64$	316	199530	360	$1.3e+09$

TableFor Γ^d, with $N = 11$, we compare the number of deterministic solutions required by the Anisotropic Smolyak (AS) using Clenshaw-Curtis abscissas, Anisotropic Full Tensor product method (AF) using Gaussian abscissas, Isotropic Smolyak (IS) using Clenshaw-Curtis abscissas and the Monte Carlo (MC) method using random abscissas, to reduce the original error by a factor of 10^4.

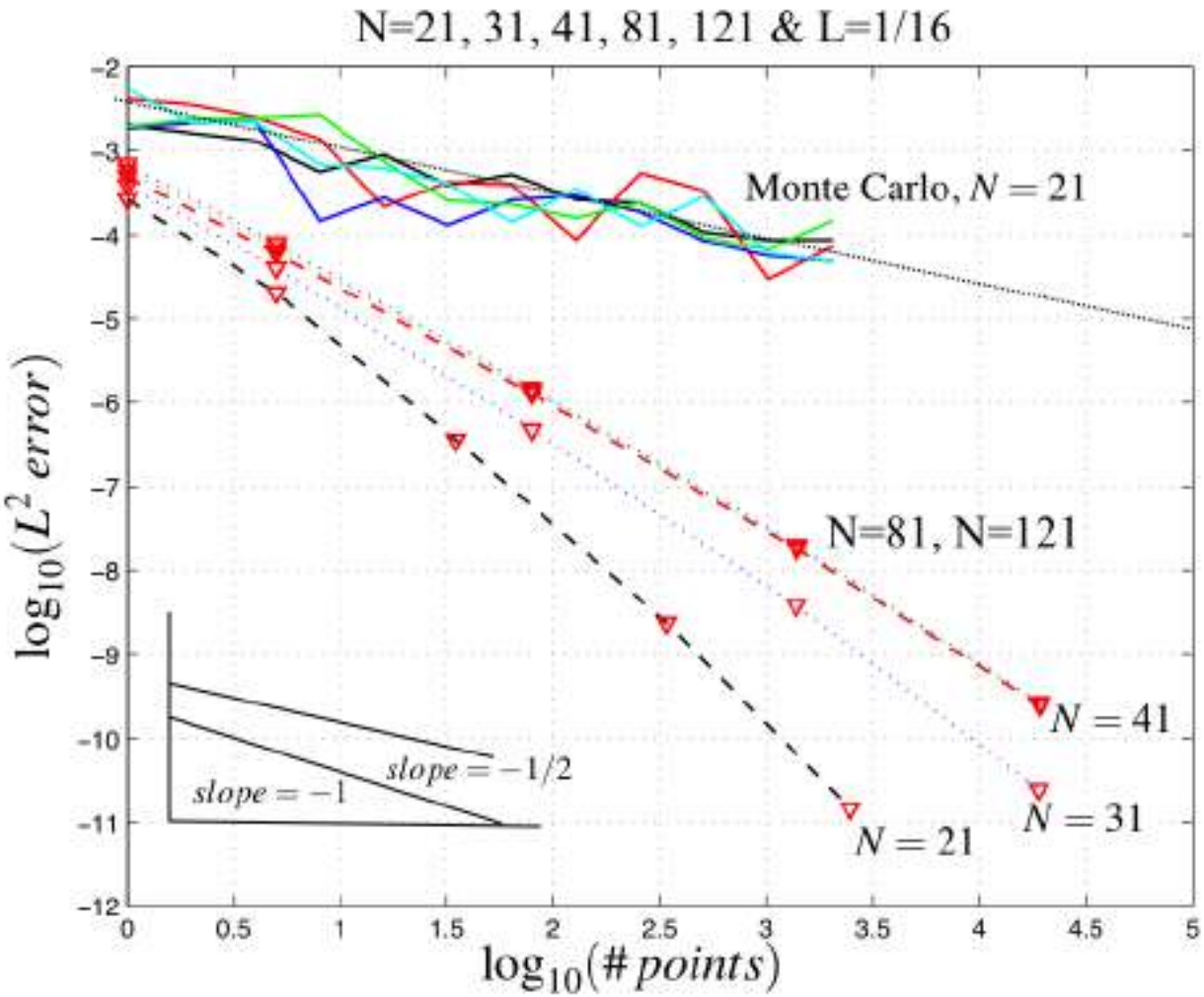
Convergence Comparisons III

$d = 21$ random variables

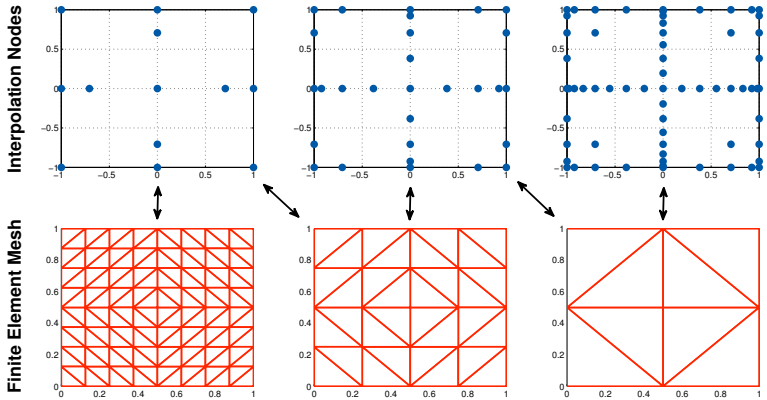


Convergence Comparisons IV:

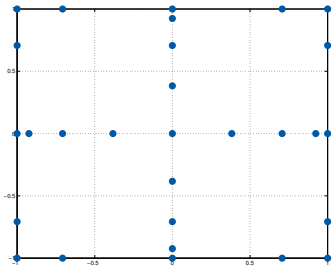
$N = 21, \ldots, 121, \ldots \infty$ random variables (*A posteriori approach*)



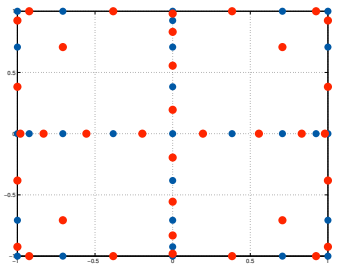
Reducing the computational cost of multivariate interpolation



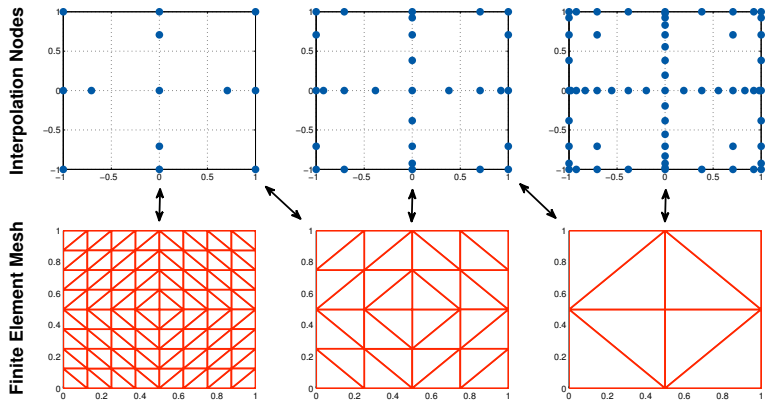
- **Exploit the hierarchy in deterministic approximation:** For a given accuracy, multilevel methods seek to reduce complexity by spreading computational cost across several resolutions of the spatial discretization
- **Exploit the hierarchy in stochastic approximation:** Sparse grids with nested grid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



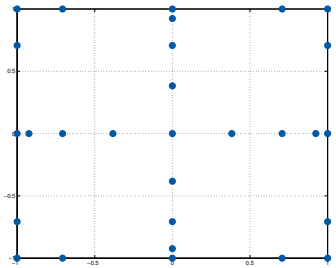
Solve $A_j \boldsymbol{c}_j = \boldsymbol{f}_j$
at all blue points
$\longrightarrow$
Interpolate to
accelerate solution
$(j = 1, \ldots, m)$



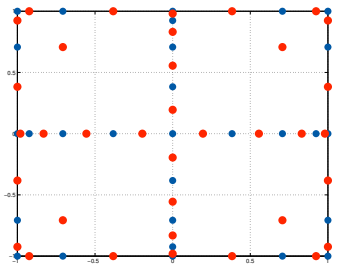
Reducing the computational cost of multivariate interpolation



- **Exploit the hierarchy in deterministic approximation:** For a given accuracy, multilevel methods seek to reduce complexity by spreading computational cost 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



Solve $A_j \boldsymbol{c}_j = \boldsymbol{f}_j$
at all blue points
$\longrightarrow$
Interpolate to
accelerate solution
$(j = 1, \ldots, m)$



Reducing the computational cost of multivariate interpolation

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]:

$$\mathcal{A}_L^{m,g}[v](\boldsymbol{y}) = \sum_{j=1}^{m_L} v(\boldsymbol{y}_j) \underbrace{\sum_{\substack{\boldsymbol{j}\in\{0,1\}^d \\ g(\boldsymbol{l}+\boldsymbol{j})\leq L}} (-1)^{|\boldsymbol{j}|_1} \prod_{n=1}^{N} \ell_{k_n}^{l_n-i_n}(y_n)}_{\Psi_{L,j}(\boldsymbol{y})} .$$

- $\{\boldsymbol{y}_j\}_{j=1}^{m_L}$ is the reordered set of the m_L interpolation points involved in $\mathcal{A}_L^{p,g}$.
- $\{\Psi_{L,j}\}_{j=1}^{m_L}$ is the simplified basis—a linear combination of tensorized Lagrange polynomials.

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

Reducing the computational cost of multivariate interpolation

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]:

$$\mathcal{A}_L^{m,g}[v](\boldsymbol{y}) = \sum_{j=1}^{m_L} v(\boldsymbol{y}_j) \underbrace{\sum_{\substack{\boldsymbol{j}\in\{0,1\}^d \\ g(\boldsymbol{l}+\boldsymbol{j})\leq L}} (-1)^{|\boldsymbol{j}|_1} \prod_{n=1}^{N} \ell_{k_n}^{l_n-i_n}(y_n)}_{\Psi_{L,j}(\boldsymbol{y})} .$$

- $\{\boldsymbol{y}_j\}_{j=1}^{m_L}$ is the reordered set of the m_L interpolation points involved in $\mathcal{A}_L^{p,g}$.
- $\{\Psi_{L,j}\}_{j=1}^{m_L}$ is the simplified basis—a linear combination of tensorized Lagrange polynomials.

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

Construction of the fully discrete solution

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

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

where

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

is the output of the solver s.t. $\|\boldsymbol{c}_j - \widetilde{\boldsymbol{c}}_j\|_{A_j} < \tau$.

- We can rewrite (after a re-ordering) the fully discrete generalized sparse grid SC approximation (at level L) as:

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

Construction of the fully discrete solution

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

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

  where

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

  is the output of the solver s.t. $\|\boldsymbol{c}_j - \widetilde{\boldsymbol{c}}_j\|_{A_j} < \tau$.

- We can rewrite (after a re-ordering) the fully discrete generalized sparse grid SC approximation (at level L) as:

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

Improved initial vectors and pre-conditioners

Example: Application to conjugate gradient (CG) methods

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

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

Accelerate the performance of the CG solver by reducing the condition number κ_j or improving the initial guess $\boldsymbol{c}_j^{(0)}$.

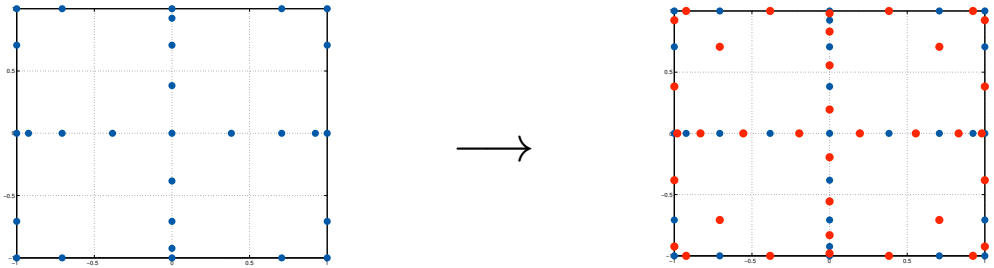
Improved initial vectors and pre-conditioners

Example: Application to conjugate gradient (CG) methods

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

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

Accelerate the performance of the CG solver by reducing the condition number κ_j or improving the initial guess $\boldsymbol{c}_j^{(0)}$.



Specifically, assume we have solved for each the vectors $\widetilde{\boldsymbol{c}}_m, m = 1, \ldots, m_{L-1}$

- for any new point $\boldsymbol{y}_j \in \Delta\mathcal{H}_L$, a good approximation to $\boldsymbol{c}_j$ is given by

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

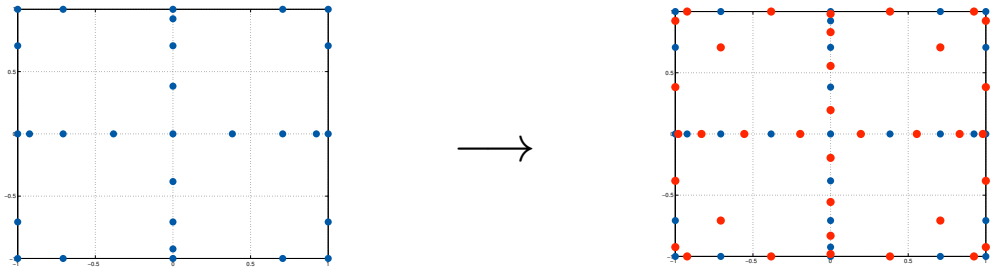
Improved initial vectors and pre-conditioners

Example: Application to conjugate gradient (CG) methods

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

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

Accelerate the performance of the CG solver by reducing the condition number κ_j or improving the initial guess $\boldsymbol{c}_j^{(0)}$.



Alternatively, suppose we have constructed pre-conditioners $\boldsymbol{P}_m, m = 1, \ldots, m_{L-1}$

- for any new point $\boldsymbol{y}_j \in \Delta\mathcal{H}_L$, improved pre-conditioners are give by

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

Computational cost analysis for ε-complexity

The goal is to estimate the computational cost within a prescribed accuracy ε, i.e., split the total error $e = u(x, \boldsymbol{y}) - \tilde{u}_{h,m_L}(x, \boldsymbol{y})$ into:

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

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

$$\|e_1\| \leq C_1 \, h^s \leq \frac{\varepsilon}{3}$$

$$\|e_2\| \leq C_2(d) \, e^{-r(d)L} \leq \frac{\varepsilon}{3}$$

$$\|e_3\| \leq C_3 \, \Lambda_L \, e_{\text{cg}} \leq \frac{\varepsilon}{3}$$

- Here s, $r(d)$ are the convergence rates of the FEM and collocation scheme, Λ_L is the Lebesgue constant, and

$$e_{\text{cg}} = \max_{\boldsymbol{y}_j \in \mathcal{H}_L} \|\boldsymbol{c}_j - \tilde{\boldsymbol{c}}_j\|_{A_j}$$

Computational cost analysis for ε-complexity

The goal is to estimate the computational cost within a prescribed accuracy ε, i.e., split the total error $e = u(x, \boldsymbol{y}) - \tilde{u}_{h,m_L}(x, \boldsymbol{y})$ into:

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

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

$$\|e_1\| \leq C_1 \, h^s \leq \frac{\varepsilon}{3}$$
$$\|e_2\| \leq C_2(d) \, e^{-r(d)L} \leq \frac{\varepsilon}{3}$$
$$\|e_3\| \leq C_3 \, \Lambda_L \, e_{\text{cg}} \leq \frac{\varepsilon}{3}$$

- Here s, $r(d)$ are the convergence rates of the FEM and collocation scheme, Λ_L is the Lebesgue constant, and

$$e_{\text{cg}} = \max_{\boldsymbol{y}_j \in \mathcal{H}_L} \|\boldsymbol{c}_j - \tilde{\boldsymbol{c}}_j\|_{A_j}$$

CG iteration estimate: minimum computational cost

Restart with "zero" vector vs. acceleration

Theorem: [Galindo, Jantsch, W.,Zhang, 2015]

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

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

where $\bar{\kappa} = \max_{\boldsymbol{y} \in \mathcal{H}_L} \bar{\kappa}(\boldsymbol{y})$.

The first line comes from the number of collocation nodes which remains the same:

$$m_L \leq e^{d-1} 2^{L+1} \left(1 + \frac{L}{d-1}\right)^{d-1}.$$

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).

CG iteration estimate: minimum computational cost

Restart with "zero" vector vs. acceleration

Theorem: [Galindo, Jantsch, W.,Zhang, 2015]

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

$$K_{\text{accel}} \leq \alpha_1(d)\, \varepsilon^{\frac{-\log(2)}{r}} \left\{ \alpha_2(d) + \alpha_3 \log\left(\frac{1}{\varepsilon}\right) \right\}^{d-1} \times \sqrt{\bar{\kappa}} \left\{ \alpha_5(d) + \log(\Lambda_L) \right\},$$

where $\bar{\kappa} = \max_{\boldsymbol{y} \in \mathcal{H}_L} \bar{\kappa}(\boldsymbol{y})$.

The first line comes from the number of collocation nodes which remains the same:

$$m_L \leq e^{d-1} 2^{L+1} \left(1 + \frac{L}{d-1}\right)^{d-1}.$$

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).

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

Some 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 $(2^{1/d} - 1) \log \varepsilon^{-1}$
- 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. (Examples 2,3)
- The acceleration scheme should always be used in adaptive interpolation settings, or with sparse grids based on hierarchical Lagrange interpolants.

Some 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 $(2^{1/d} - 1) \log \varepsilon^{-1}$
- 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. (Examples 2,3)
- The acceleration scheme should always be used in adaptive interpolation settings, or with sparse grids based on hierarchical Lagrange interpolants.

Some 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 $(2^{1/d} - 1) \log \varepsilon^{-1}$
- 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. (Examples 2,3)
- The acceleration scheme should always be used in adaptive interpolation settings, or with sparse grids based on hierarchical Lagrange interpolants.

Example 1: Global Basis w/ Error Balancing

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

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

with

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

Error	#SG Pts	CG iters	CG + acc	% Savings
1×10^{-2}	137	29,355	22,219	24.3
5×10^{-3}	401	180,087	90,300	49.9
1×10^{-3}	1105	2,072,625	696,935	66.4
5×10^{-4}	2929	11,253,264	2,217,615	80.3
1×10^{-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

Example 1: Global Basis w/ Error Balancing

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

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

with

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

Error	#SG Pts	CG iters	CG + acc	% Savings
1×10^{-2}	137	29,355	22,219	24.3
5×10^{-3}	401	180,087	90,300	49.9
1×10^{-3}	1105	2,072,625	696,935	66.4
5×10^{-4}	2929	11,253,264	2,217,615	80.3
1×10^{-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

Example 2: Global Basis w/ Interpolated Preconditioners

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

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

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

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

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

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

- $\mathbb{E}[y_n] = 0$ 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: Savings vs Level/Error

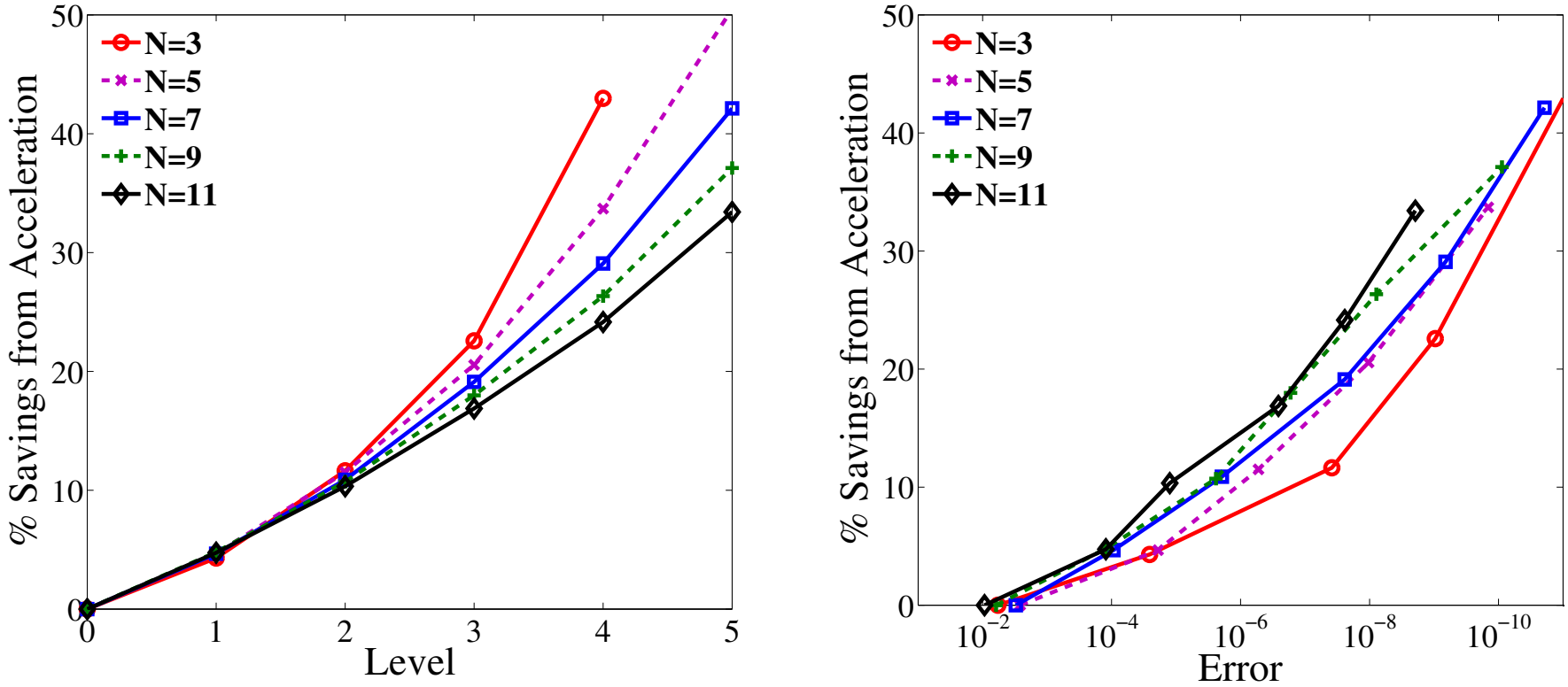


Figure Percentage reduction in CG iterations per level (left) and vs error (right) with $d = 3, 5, 7, 9, 11$ and 13 and for correlation length $R = 1/64$

Example 3: Nonlinear problem

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

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

with a as in Example 2:

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

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 3: Nonlinear problem

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

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

with a as in Example 2:

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

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 3: $-\nabla(a \cdot \nabla u) + F(u) = f$

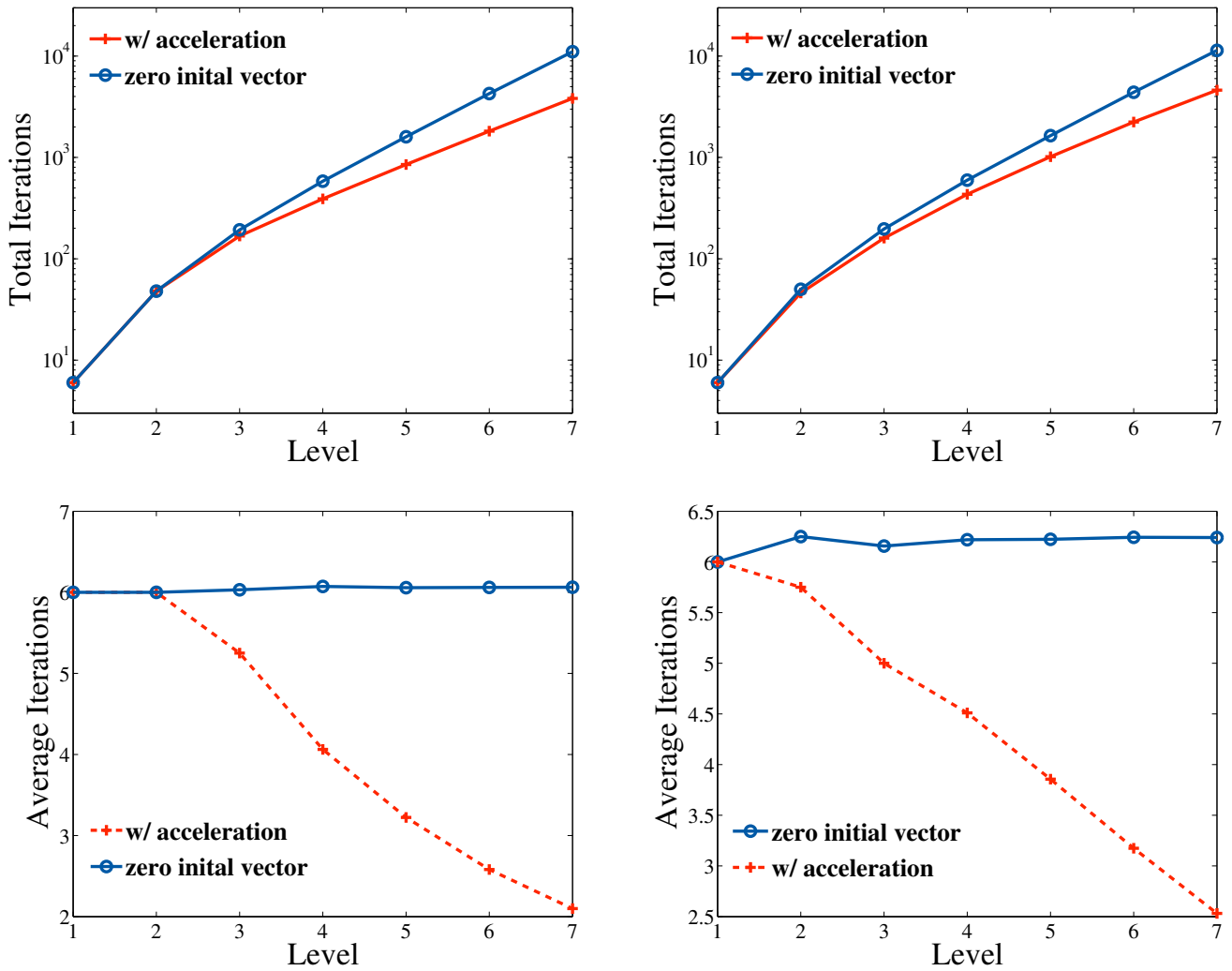


Figure Cumulative total (top) and average per-level (bottom) number of Newton iterations with $F(u) = u * u'$ (left) and $F(u) = u^5$ (right)