Uncertainty Quantification course

Lecture 8: Multidimensional SC and NISP

University of Amsterdam, fall 2023

1 November 2023

Outline

1. Homework assignment from last week
2. Stochastic collocation with dim>1
3. Discrete projection & NISP

Homework assignment from last week

Presentation by team 1

Stochastic Collocation: recap

PDE: $\mathcal{L}(u, Z) = \mathcal{F}(Z), \quad u = u(x, Z)$, dim($Z$)=1

(i) Select set of nodes in Z-space: $\theta_M := \{Z^{(j)}\}_{j=1}^{M}$

(ii) Solve PDE on all nodes: $\mathcal{L}(u, Z^{(j)}) = \mathcal{F}(Z^{(j)})$

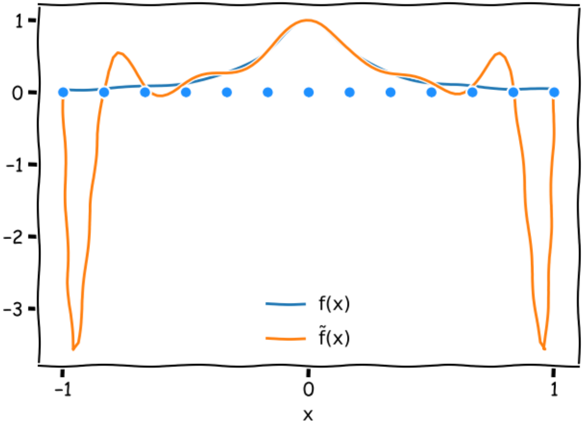
Solutions: $u^{(j)}(x)$ at node $Z^{(j)}$

(iii) Interpolation: $\tilde{u}_M(x, Z) = \sum_{j=1}^{M} u^{(j)}(x)\, L_j(Z)$

with Lagrange basis functions $L_j(Z) := \prod_{i \neq j} \frac{Z - Z^{(i)}}{Z^{(j)} - Z^{(i)}}$

standard SC in 1D

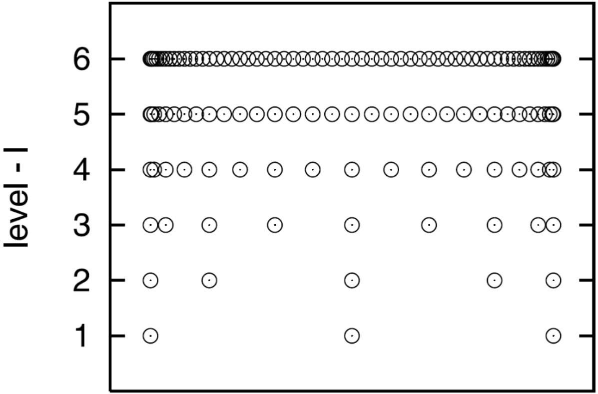
Uniform $\Theta_m := \{Z_i\}_{i=1}^m$: Runge phenomenon



standard SC in 1D

Better choice:

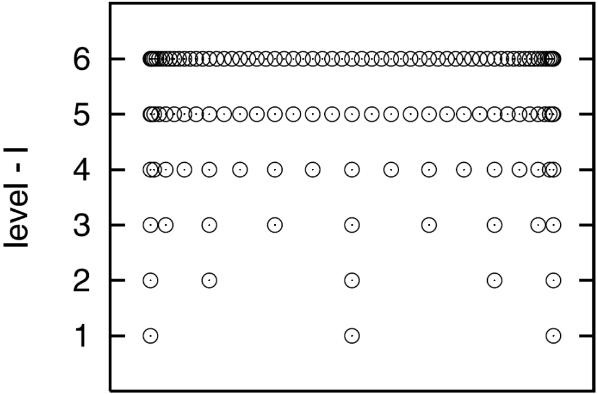
- non-uniform 1D quadrature points: building blocks SC method e.g. Clenshaw-Curtis (CC) nodes



standard SC in 1D

Better choice:

- non-uniform 1D quadrature points: building blocks SC method e.g. Clenshaw-Curtis (CC) nodes



Can be a nested rule over ‘levels’.

standard SC in 1D

Example over $[0, 1]$ (rescaled from standard CC domain of [-1, 1])

- Level $l = 1$: $x_i^{(1)} \in \{0.5\}$,
- Level $l = 2$: $x_i^{(2)} \in \{0.0,\ 0.5,\ 1.0\}$,
- Level $l = 3$: $x_i^{(3)} \in \{0.0,\ 0.146,\ 0.5,\ 0.854,\ 1.0\}$.

standard SC in 1D

Example over $[0, 1]$ (rescaled from standard CC domain of [-1, 1])

- Level $l = 1$: $x_i^{(1)} \in \{0.5\}$,
- Level $l = 2$: $x_i^{(2)} \in \{0.0,\ 0.5,\ 1.0\}$,
- Level $l = 3$: $x_i^{(3)} \in \{0.0,\ 0.146,\ 0.5,\ 0.854,\ 1.0\}$.

Exponential increase in number of points per level;

$$m_l = \begin{cases} 2^{l-1} + 1 & l > 1 \\ 1 & l = 1 \end{cases}$$

but useful for refinement.

standard SC in 1D

Example over $[0, 1]$ (rescaled from standard CC domain of [-1, 1])

- Level $l = 1$: $x_i^{(1)} \in \{0.5\}$,
- Level $l = 2$: $x_i^{(2)} \in \{0.0,\ 0.5,\ 1.0\}$,
- Level $l = 3$: $x_i^{(3)} \in \{0.0,\ 0.146,\ 0.5,\ 0.854,\ 1.0\}$.

Exponential increase in number of points per level;

$$m_l = \begin{cases} 2^{l-1} + 1 & l > 1 \\ 1 & l = 1 \end{cases}$$

but useful for refinement.

Weights are such that $\sum_i f(x_i) w_i \approx \int f(x) p(x) dx = \mathbb{E}[f]$.

Extension to higher dimensions

- Remember gPC with $d = 2$ uncertain inputs:

$$f(Z_1, Z_2) \approx \tilde{f}(Z_1, Z_2) = \sum_{\mathbf{i}} \hat{f}_{\mathbf{i}} \Phi_{i_1}(Z_1)\Phi_{i_2}(Z_2)$$

- Here, $\mathbf{i}$ was a **multi index** (i_1, i_2), determining the order of the basis functions.

Extension to higher dimensions

- Remember gPC with $d = 2$ uncertain inputs:

$$f(Z_1, Z_2) \approx \tilde{f}(Z_1, Z_2) = \sum_{\mathbf{i}} \hat{f}_{\mathbf{i}} \Phi_{i_1}(Z_1) \Phi_{i_2}(Z_2)$$

- Here, $\mathbf{i}$ was a **multi index** (i_1, i_2), determining the order of the basis functions.
- In SC, a multi index $\mathbf{l} = (l_1, l_2)$ determines the order of 1D nodes used for each input.
- Example: level 1 nodes = $\{0\}$, level 2 nodes = $\{-1, 0, 1\}$

$$\mathbf{l} = (1, 2) \Rightarrow \{0\} \otimes \{-1, 0, 1\} = \{(0, -1), (0, 0), (0, 1)\}$$

Code is sampled at these 3 (Z_1, Z_2) via this multi index.

Interpolation in multi-d

d=1: nodal set θ_M, $\tilde{u}(Z) = \sum_{j=1}^{M} u(Z^{(j)})\, L_j(Z;\theta_M)$, $\;Z^{(j)} \in \theta_M$

d>1: Tensor product, nodal set $\;\Theta_M = \theta_{M_1} \otimes \cdots \otimes \theta_{M_d}$

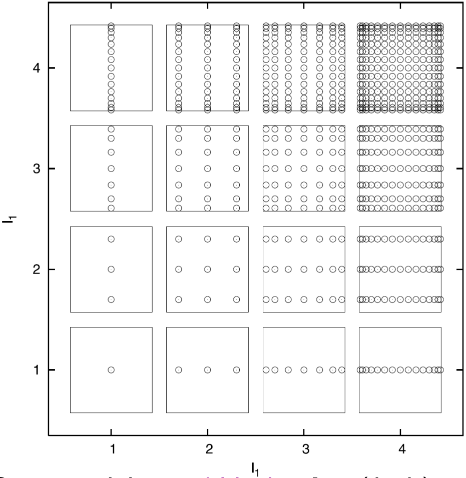
$$\tilde{u}(Z) = \sum_{j_1=1}^{M_1} \cdots \sum_{j_d=1}^{M_d} u(Z^{(j_1,\dots,j_d)}) L_{j_1,\dots,j_d}(Z;\Theta_M)$$

nodes $\;Z^{(j_1,\dots,j_d)} = (Z_1^{(j_1)}, \dots, Z_d^{(j_d)}) \in \Theta_M$

and basis functions $\;L_{j_1,\dots,j_d}(Z;\Theta_M) = \prod_{n=1}^{d} L_{j_n}(Z_n;\theta_{M_n})$

standard SC in multiple dimension

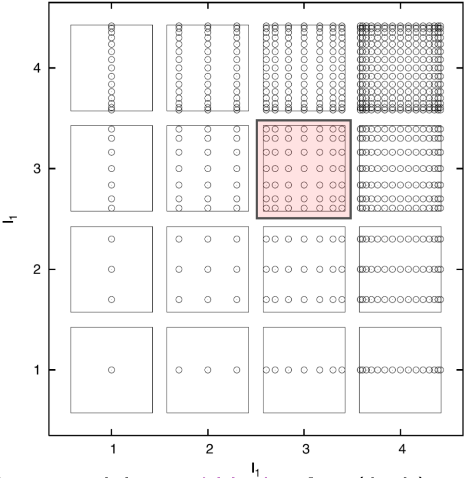
- More than 1 input: tensor products of 1D quad rules
- In 2D:



- Standard SC: user picks multi index $\mathbf{l} = (l_1, l_2)$

standard SC in multiple dimension

- More than 1 input: tensor products of 1D quad rules
- In 2D:



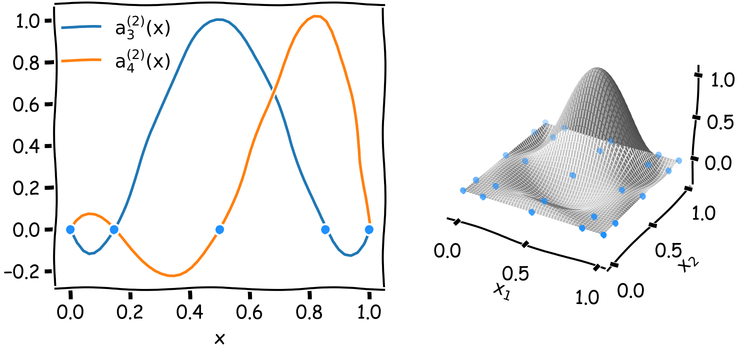
- Standard SC: user picks multi index $\mathbf{l} = (l_1, l_2)$

standard SC in multiple dimension

- Surrogate in 2D:

$$\tilde{f}(\mathbf{Z}) = \sum_{j_1=1}^{m_{l_1}} \sum_{j_2=1}^{m_{l_2}} f\left(Z_1^{(j_1)}, Z_2^{(j_2)}\right) L_{j_1}(Z_1) \otimes L_{j_2}(Z_2)$$

- 2D basis functions:



standard SC vs MC

- Example: wing-weight function

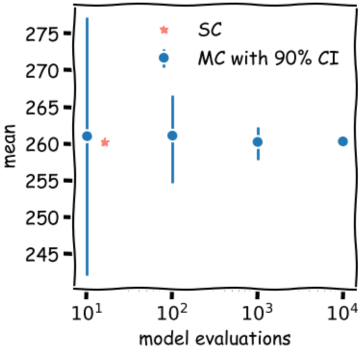
$$f(\mathbf{Z}) = 0.036 S_w^{0.758} W_{fw}^{0.0035} \left(\frac{A}{\cos^2(\Lambda)} \right)^{0.6} q^{0.006} \lambda^{0.04} \left(\frac{100 t_c}{\cos(\Lambda)} \right)^{-0.3} (N_z W_{dg})^{0.49} + S_w W_p$$

$S_w \in [150, 200]$	wing area (ft^2)
$W_{fw} \in [220, 300]$	weight of fuel in the wing (lb)
$A \in [6, 10]$	aspect ratio
$\Lambda \in [-10, 10]$	quarter-chord sweep (degrees)
$q \in [16, 45]$	dynamic pressure at cruise (lb/ft^2)
$\lambda \in [0.5, 1]$	taper ratio
$t_c \in [0.08, 0.18]$	aerofoil thickness to chord ratio
$N_z \in [2.5, 6]$	ultimate load factor
$W_{dg} \in [1700, 2500]$	flight design gross weight (lb)
$W_p \in [0.025, 0.08]$	paint weight (lb/ft^2)

- Example: wing-weight function

$f(\mathbf{Z}) =$

$0.036 S_w^{0.758} W_{fw}^{0.0035} \left(\frac{A}{cos^2(\Lambda)}\right)^{0.6} q^{0.006} \lambda^{0.04} \left(\frac{100 t_c}{cos(\Lambda)}\right)^{-0.3} (N_z W_{dg})^{0.49} +$
$S_w W_p$

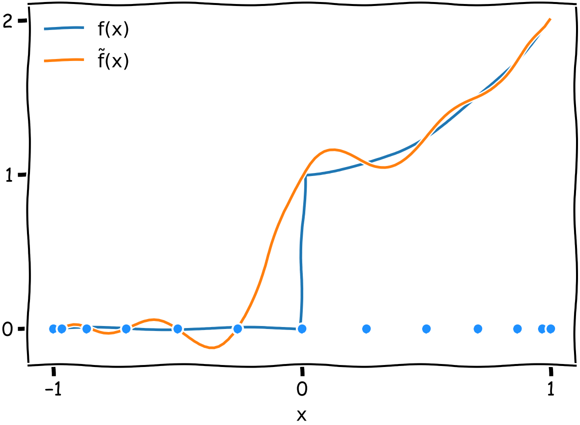


standard SC vs MC

- SC can outperform MC

standard SC vs MC

- SC can outperform MC
- Caveat 1: Accuracy poor if $f(\mathbf{x})$ is not smooth

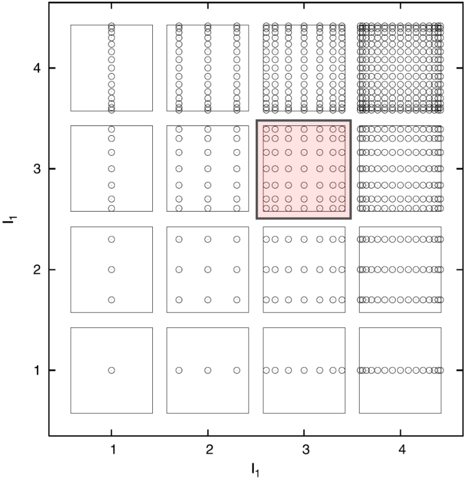


standard SC vs MC

- SC can outperform MC
- Caveat 2: $\mathbf{Z} = \{Z_1, \cdots, Z_d\}$, d must be small (e.g. ≤ 5)

standard SC vs MC

- SC can outperform MC
- Caveat 2: $\mathbf{Z} = \{Z_1, \cdots, Z_d\}$, d must be small (e.g. ≤ 5)
  2D: given $\mathbf{l} = (l_1, l_2)$, there are $M = M_1 \times M_2$ code evaluations.



Sparse grids

With tensor product $\Theta_M = \theta_{M_1} \otimes \cdots \otimes \theta_{M_d}$, total number of nodes is $M_1 \times M_2 \times ... \times M_d$.

Grows very rapidly with d: **curse of dimension**

Sparse grids

Curse of dimension example:

- You have a very fast code that takes 1 second to complete.
- You select 10 nodes per input parameter.

Sparse grids

Curse of dimension example:

- You have a very fast code that takes 1 second to complete.
- You select 10 nodes per input parameter.

1 parameter, simulation time = 10 seconds.

Sparse grids

Curse of dimension example:

- You have a very fast code that takes 1 second to complete.
- You select 10 nodes per input parameter.

1 parameter, simulation time = 10 seconds.

2 parameters, simulation time $\approx$ 1.5 minutes.

Sparse grids

Curse of dimension example:

- You have a very fast code that takes 1 second to complete.
- You select 10 nodes per input parameter.

1 parameter, simulation time = 10 seconds.

2 parameters, simulation time $\approx$ 1.5 minutes.

3 parameters, simulation time $\approx$ 15 minutes.

Sparse grids

Curse of dimension example:

- You have a very fast code that takes 1 second to complete.
- You select 10 nodes per input parameter.

1 parameter, simulation time = 10 seconds.

2 parameters, simulation time $\approx$ 1.5 minutes.

3 parameters, simulation time $\approx$ 15 minutes.

4 parameters, simulation time $\approx$ 3 hours.

Sparse grids

Curse of dimension example:

- You have a very fast code that takes 1 second to complete.
- You select 10 nodes per input parameter.

1 parameter, simulation time = 10 seconds.

2 parameters, simulation time $\approx$ 1.5 minutes.

3 parameters, simulation time $\approx$ 15 minutes.

4 parameters, simulation time $\approx$ 3 hours.

5 parameters, simulation time $\approx$ 1.1 days.

Sparse grids

Curse of dimension example:

- You have a very fast code that takes 1 second to complete.
- You select 10 nodes per input parameter.

1 parameter, simulation time = 10 seconds.

2 parameters, simulation time $\approx$ 1.5 minutes.

3 parameters, simulation time $\approx$ 15 minutes.

4 parameters, simulation time $\approx$ 3 hours.

5 parameters, simulation time $\approx$ 1.1 days.

6 parameters, simulation time $\approx$ 1.5 weeks.

Sparse grids

Curse of dimension example:

- You have a very fast code that takes 1 second to complete.
- You select 10 nodes per input parameter.

1 parameter, simulation time = 10 seconds.

2 parameters, simulation time $\approx$ 1.5 minutes.

3 parameters, simulation time $\approx$ 15 minutes.

4 parameters, simulation time $\approx$ 3 hours.

5 parameters, simulation time $\approx$ 1.1 days.

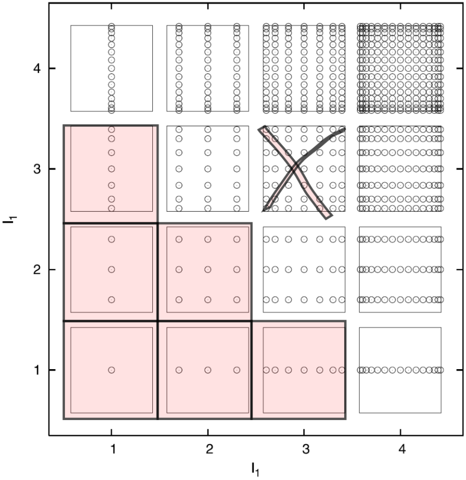
6 parameters, simulation time $\approx$ 1.5 weeks.

...

20 parameters, simulation time $\approx 3.17 \cdot 10^{12}$ years, 226 times the age of the universe.
No supercomputer can beat this, need smarter algorithms.

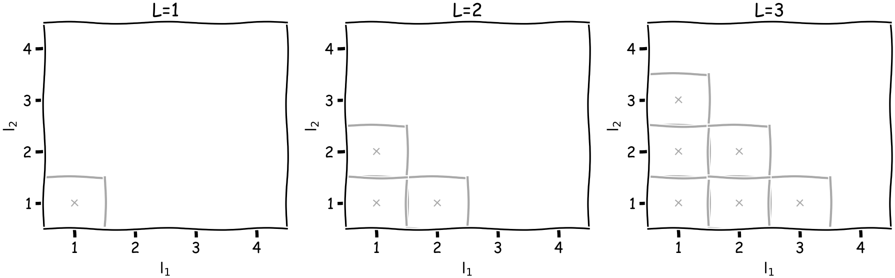
Sparse grids

- Postpone curse: sparse grids
  Select index set $\Lambda = \{\mathbf{l}_1, \mathbf{l}_2, \cdots\}$ instead of a single $\mathbf{l}$



Smolyak sparse grids

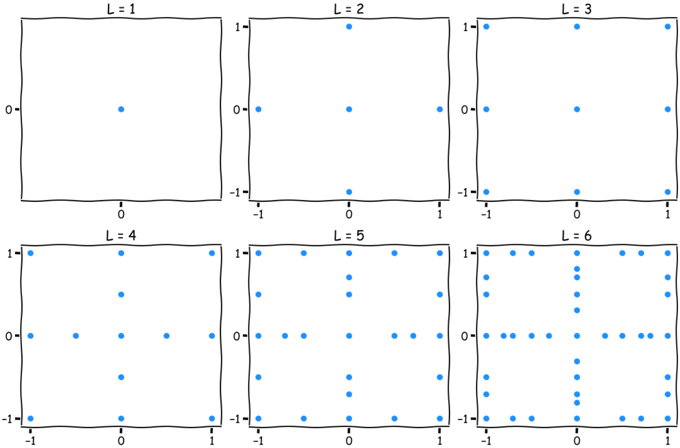
Visually, a standard sparse grid with $\{\mathbf{l} \mid |\mathbf{l}| \leq L + d - 1\}$ selects a 'triangle' of multi indices when $d = 2$ (2 uncertain inputs):



Remember: $|\mathbf{l}| = l_1 + \cdots + l_d$.

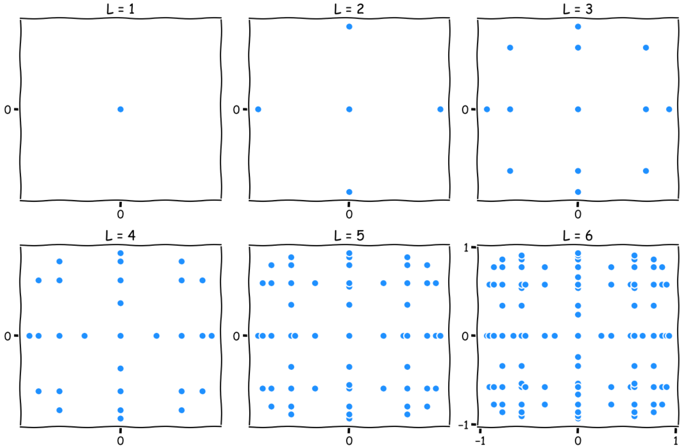
Isotropic sparse grids: sampling plan construction

- Given $\Lambda = \{(1,1),(1,2),\cdots\}$, create sampling plan by taking combination of tensor products for each $\mathbf{l} \in \Lambda$
- For nested Clenshaw Curtis quad rule:



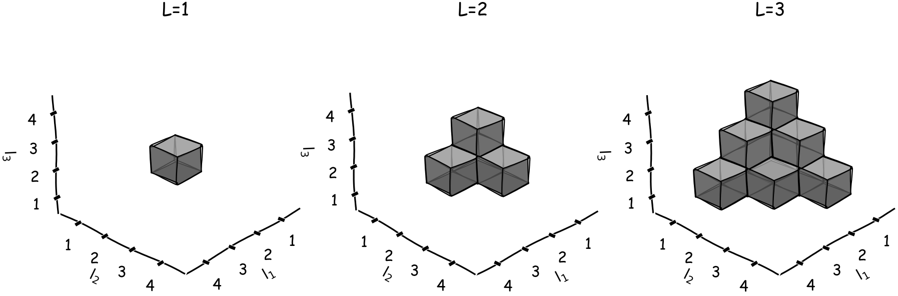
Isotropic sparse grids: sampling plan construction

- Given $\Lambda = \{(1,1),(1,2),\cdots\}$, create sampling plan by taking combination of tensor products for each $\mathbf{l} \in \Lambda$
- For non-nested Gaussian quad rule:



Smolyak sparse grids

For $d = 3$, $\{\mathbf{l} \mid |\mathbf{l}| \leq L + d - 1\}$ selects a 'simplex' of multi indices:



Smolyak sparse grids

- No explicit formula available to calculate M, total #nodes in sparse grid. Can be calculated via algorithm[1]. For Clenshaw-Curtis:

DIM:	1	2	3	4	5
CFN_E LEVEL					
0	1	1	1	1	1
1	3	5	7	9	11
2	5	13	25	41	61
3	9	29	69	137	241
4	17	65	177	401	801
5	33	145	441	1,105	2,433
6	65	321	1,073	2,929	6,993
7	129	705	2,561	7,537	19,313
8	257	1,537	6,017	18,945	51,713
9	513	3,329	13,953	46,721	135,073
10	1,025	7,169	32,001	113,409	345,665

DIM:	6	7	8	9	10
CFN_E LEVEL					
0	1	1	1	1	1
1	13	15	17	19	21
2	85	113	145	181	221
3	389	589	849	1,177	1,581
4	1,457	2,465	3,937	6,001	8,801
5	4,865	9,017	15,713	26,017	41,265
6	15,121	30,241	56,737	100,897	171,425
7	44,689	95,441	190,881	361,249	652,065
8	127,105	287,745	609,025	1,218,049	2,320,385
9	350,657	836,769	1,863,937	3,918,273	7,836,545
10	943,553	2,362,881	5,515,265	12,133,761	25,370,753

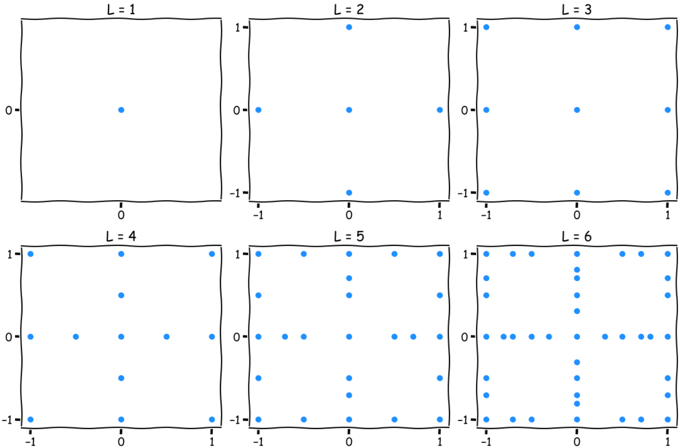
Full tensor grid equivalent would be 1025^{10}

[1] Burkardt, J. (2014). Counting Abscissas in Sparse Grids.

Anisotropic sparse grids: dimension adaptivity

- Isotopic sparse grids: less points but all inputs are equal.

$$\Lambda = \{\mathbf{l} \mid |\mathbf{l}|_1 - d + 1 \leq L\}$$



Anisotropic sparse grids: dimension adaptivity

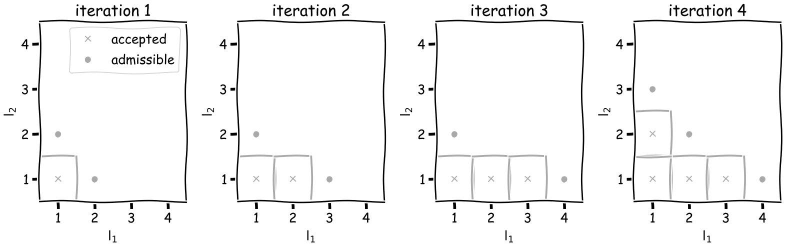
- In practice: some inputs are more important than others.

Anisotropic sparse grids: dimension adaptivity

- In practice: some inputs are more important than others.
- Dimension-adaptive sampling: find out 'on-the-fly' which ones

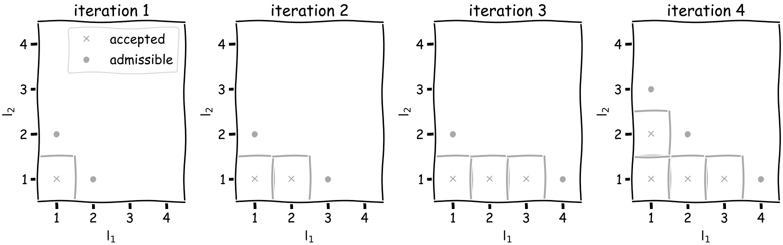
  $\rightarrow$ start with $\Lambda = \{1, 1, \cdots, 1\}$

  $\rightarrow$ adaptively refine Λ, add only 'important' $\mathbf{l} = (l_1, \cdots, l_d)$



Anisotropic sparse grids: dimension adaptivity

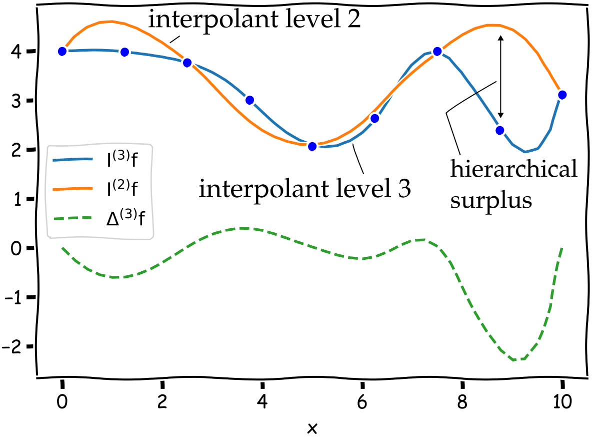
- In practice: some inputs are more important than others.
- Dimension-adaptive sampling: find out 'on-the-fly' which ones
  $\rightarrow$ start with $\Lambda = \{1, 1, \cdots, 1\}$
  $\rightarrow$ adaptively refine Λ, add only 'important' $\mathbf{l} = (l_1, \cdots, l_d)$



Which $\mathbf{l}$ are important?

Adaptive grids

One possibility: **hierarchical surplus**, error between **new** code value and **old** interpolant prediction.



From all candidate **l**, select the one with the highest surplus.

Adaptive grids

Example:
$f(\mathbf{x}) =$
$6x_1 + 4x_2 + 5.5x_3 + 3x_1x_2 + 2.2x_1x_3 + 1.4x_2x_3 + x_4 + 0.5x_5 + 0.2x_6 + 0.1x_7$

Polynomial function, adaptive algorithm should be able to **exactly** find it in 10 iterations

Adaptive grids

Example:

$f(\mathbf{x}) =$

$6x_1 + 4x_2 + 5.5x_3 + 3x_1x_2 + 2.2x_1x_3 + 1.4x_2x_3 + x_4 + 0.5x_5 + 0.2x_6 + 0.1x_7$

Polynomial function, adaptive algorithm should be able to **exactly** find it in 10 iterations

$\mathbf{l} = (1, 0, 0, 0, 0, 0, 0)$ refines x_1 line in 7-dimensional hypercube, enough to exactly describe $6x_1$ term.

Adaptive grids

Example:
$f(\mathbf{x}) =$
$6x_1 + 4x_2 + 5.5x_3 + 3x_1x_2 + 2.2x_1x_3 + 1.4x_2x_3 + x_4 + 0.5x_5 + 0.2x_6 + 0.1x_7$

Polynomial function, adaptive algorithm should be able to **exactly** find it in 10 iterations

$\mathbf{l} = (1, 0, 0, 0, 0, 0, 0)$ refines x_1 line in 7-dimensional hypercube, enough to exactly describe $6x_1$ term.

$\mathbf{l} = (1, 1, 0, 0, 0, 0, 0)$ refines (x_1, x_2) plane in 7-dimensional hypercube, enough to exactly describe $3x_1x_2$ term.

Discrete projection

aka Non-Intrusive Spectral Projection (NISP) or pseudospectral method

Recall (lecture 2) approximation of function $f(x)$ by projection:

$f(x) \approx f_N(x) = \sum_{n=0}^{N} \hat{f}_n \, \phi_n(x)$ with orthogonal basis functions $\{\phi_n(x)\}$,

and $\hat{f}_n = \frac{\langle f, \phi_n \rangle}{\langle \phi_n, \phi_n \rangle}$ with appropriate (weighted) inner product $\langle ., . \rangle$

Discrete projection: approximate inner products with **quadrature**

Quadrature

Quadrature: numerical integration, $\int f(x)w(x)\,dx \approx \sum_{j=1}^{q} f(x^{(j)})\alpha^{(j)}$ with nodes $\{x^{(j)}\}$ and weights $\{\alpha^{(j)}\}$ (density $w(x)$ absorbed into weights)

Quadrature in dim>1 is called **cubature**

Inner products for projection with q-point quadrature/cubature rule:

$$\langle f, \phi_n \rangle = \int f(x)\,\phi_n(x)\,w(x)\,dx \approx \sum_{j=1}^{q} f(x^{(j)})\,\phi_n(x^{(j)})\,\alpha^{(j)}$$

NISP

Stochastic system (e.g. PDE) with exact solution $u(t,x,Z)$, $Z \in \mathbb{R}^d$

Spectral (gPC) approximation: $u_N(t,x,Z) = \sum_{n=0}^{N} \hat{u}_n(t,x)\Phi_n(Z)$

$$\hat{u}_n(t,x) = \gamma_n^{-1}\langle u, \Phi_n\rangle = \gamma_n^{-1}\mathbb{E}[u(t,x,Z),\Phi_n(Z)]$$

$$\approx \tilde{u}_n(t,x) = \gamma_n^{-1}\sum_{j=1}^{q} u(t,x,z^{(j)})\,\Phi_n(z^{(j)})\,\alpha^{(j)}$$

Needed: q solutions of original (non-stochastic) system, $u(t,x,z^{(j)})$ with $j = 1, ..., q$

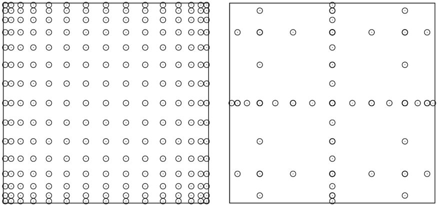
$\rightarrow$ NISP is a **non-intrusive method**

dim(Z)>1: Cubature

From quadrature if dim(Z)=1 to cubature if dim(Z)>1:

Tensor product construction, based on 1-d quadrature and nodal sets

Again, sparse grids to reduce no. of nodes



dim(Z)>1: Cubature

Example: your homework assignment from lecture 6!

Consider the following function with two random inputs:

$$f(Z_1, Z_2) = -(Z_1 - 2)(Z_2 - 1)^3 + \exp\left[-\frac{1}{2}(Z_1 - 2)^2 - \frac{1}{10}(Z_2 - 1)^2\right]$$

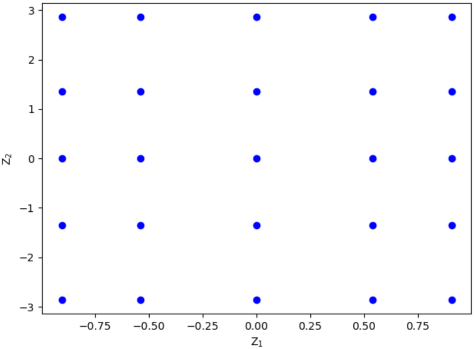
One input has a uniform distribution on $[1, 3]$, the other has a normal distribution with mean 1 and variance 1. Thus, $Z_1 \sim \mathcal{U}[1, 3]$ and $Z_2 \sim \mathcal{N}(1, 1)$.

- If $f(Z_1, Z_2)$ is considered as your 'code'.
- This involved computing $\mathbb{E}\left[f(Z_1, Z_1)\Phi^{(1)}_{i_1}(Z_1)\Phi^{(2)}_{i_2}(Z_2)\right]/\gamma_{\mathbf{i}}$.
- This can be done with in-built (SciPy) integration routines.

- But also with cubature:

$$\mathbb{E}\left[f(Z_1, Z_1)\Phi^{(1)}_{i_1}(Z_1)\Phi^{(2)}_{i_2}(Z_2)\right] \approx$$

$$\sum_{p=1}^{m_p}\sum_{q=1}^{m_q} f(Z_{1,p}Z_{2,q})\Phi^{(1)}_{i_1}(Z_1)\Phi^{(2)}_{i_2}(Z_2)\alpha_{1,p}\alpha_{2,q}$$



Galerkin, Collocation & NISP

Solution $u(t,x,Z)$ of PDE system, approximated by
$\tilde{u}(t,x,Z) = \sum_{n=0}^{N} \hat{u}_n(t,x)\Psi_n(Z)$

- Galerkin method: projection-based, leading to coupled equations for coefficients $\{\hat{u}_n(t,x)\}_{n=0}^{N}$. Intrusive.
- Collocation: interpolation in Z-space of deterministic solutions $\{u(t,x,z^{(j)})\}_{j=1}^{q}$. Non-intrusive.
- NISP: projection coefficients $\{\hat{u}_n(t,x)\}_{n=0}^{N}$ approximated with cubature using deterministic solutions $\{u(t,x,z^{(j)})\}_{j=1}^{q}$. Non-intrusive.

Next week

- Inverse UQ: Bayesian calibration and MCMC.
  Xiu section 8.2, Smith section 8.1 - 8.3 (on Canvas)
- Presentation about homework by TEAM 2?
- Homework: Multi-dimensional SC *(more details on Canvas)*