UNIL | Université de Lausanne
HEC Lausanne

Banishing the Curse of Dimensionality in Solving Economic Models

Simon Scheidegger

RANEPA, Moscow, October 24th, 2018

More info: https://sites.google.com/site/simonscheidegger/publications

Outline

1. Motivation – Heterogeneous & high-dimensional economic models

2. Adaptive Sparse Grids (see, e.g., Brumm & Scheidegger (2017); Brumm, Kubler & Scheidegger (2017))

I. From Full (Cartesian) to Sparse Grids

II. Adaptive Sparse Grids

3. High-dimensional model reduction (see, e.g., Eftekhari, Scheidegger & Schenk (2017), Eftekhari & S. (2018))

I. Standard HDMR

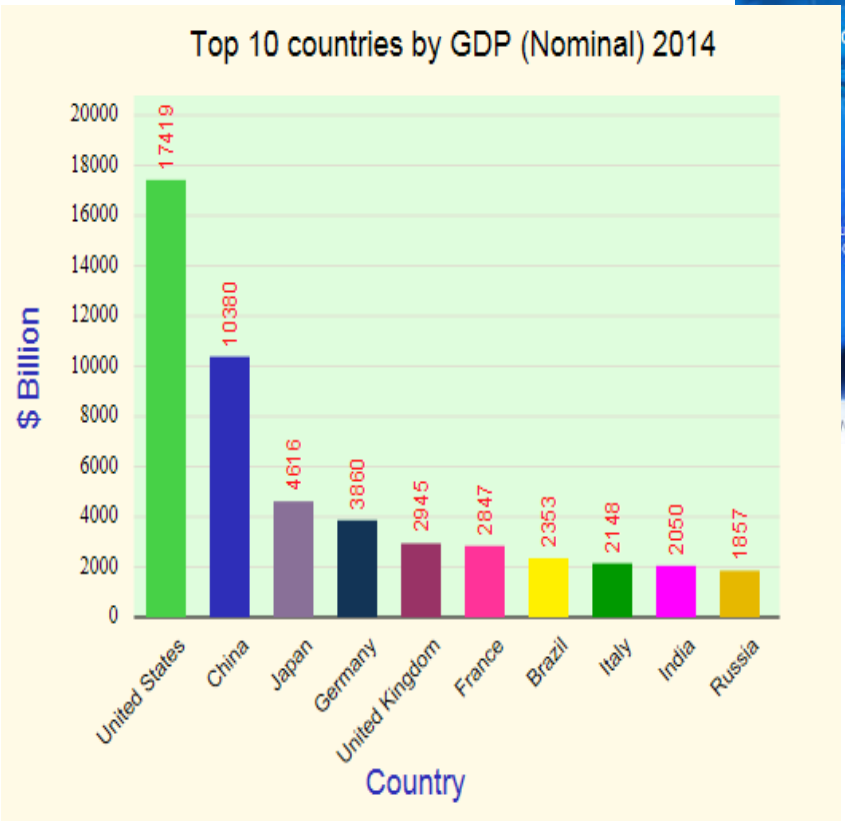
II. Adaptive HDMR

Example – Heterogeneity in IRBC models

- Model trade imbalance

- FX rates

-...



- How many regions does a minimal model have?
- Are policy functions smooth? (borrowing constraints)

→ **Model heterogeneous & high-dimensional**

Example: pay-as-you-go social security

(See, e.g., Krüger & Kübler (2006))

- Income distribution across age.

- How many age groups.

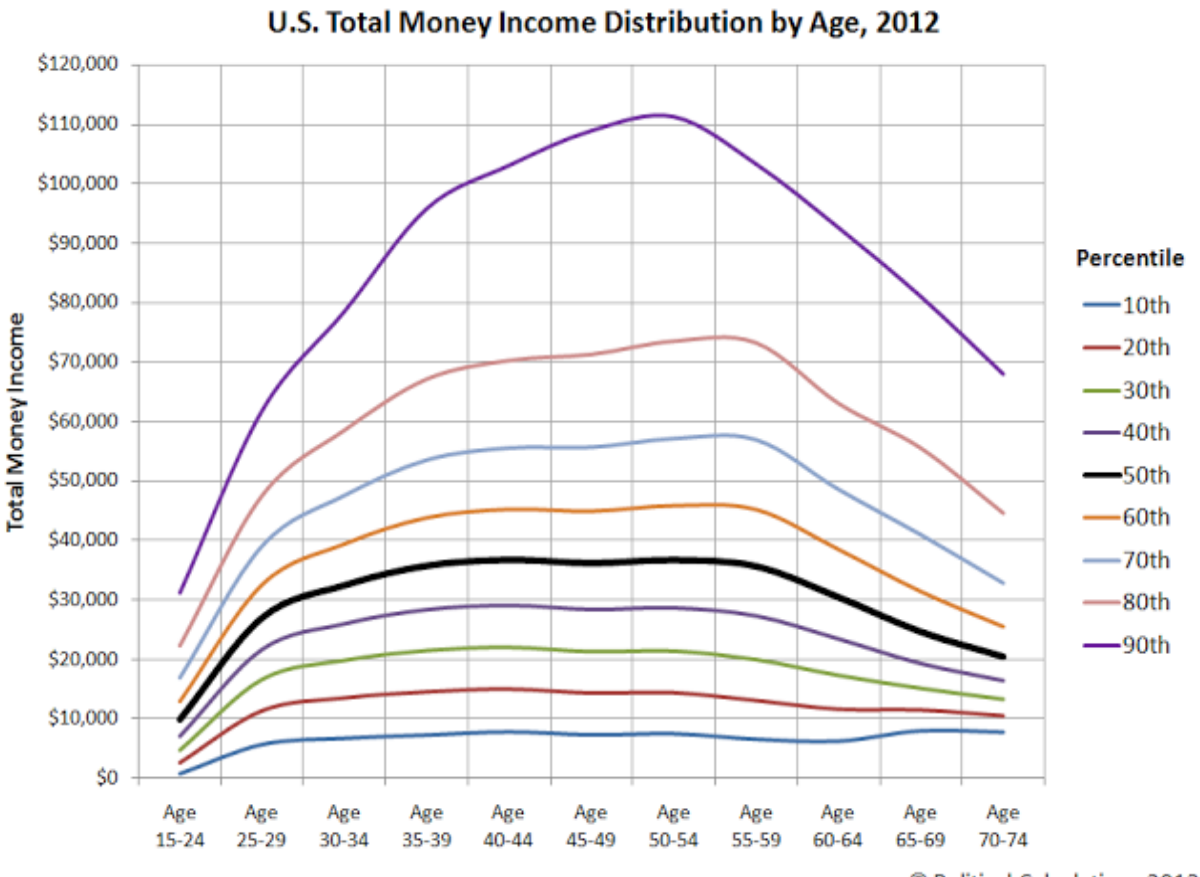
- borrowing constraints.

- taxes on capital and labour.

- uncertainty in productivity and tax rates.

→ already a simple problem is 60d, with stochastic components.

→ Model: heterogeneous & very high-dimensional



Value function iteration (on irregularly-shaped domains)

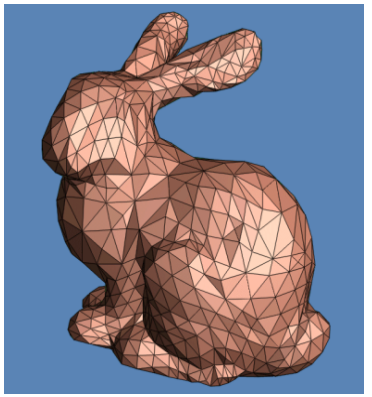
e.g. Bellman (1961), Stokey, Lucas & Prescott (1989), Judd (1998), ...

The solution is approached in the limit as $j \to \infty$ by iterations on:

$$V_{j+1}(x) = \max_u \{r(x,u) + \beta V_j(\tilde{x})\}$$

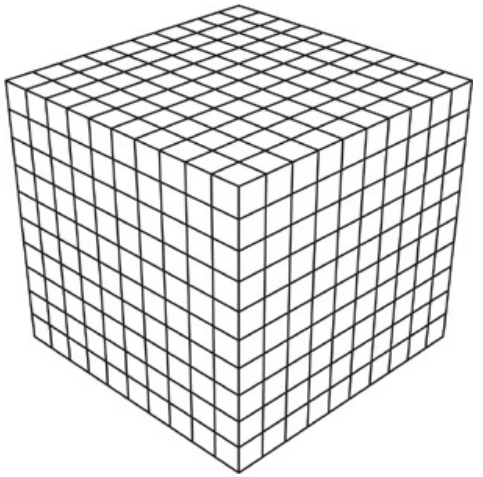
s.t.

$$\tilde{x} = g(x,u)$$



x: point in state space; describes your system.
State-space potentially **irregularly-shaped** and **high-dimensional.**

`old solution':
high-dimensional function on which **we interpolate**.



→ **N^d** points in ordinary discretization schemes.

→ ***"Curse of dimensionality"***

→ **Use-case e.g. for Adaptive Sparse Grids**

How many is dimensions is high dimensions?

?

How many is dimensions is high dimensions?

Number of parameters (the dimension)	**Number of model runs** (at 10 points per dimension)	**Time for parameter study** (at 1 second per run)
1	10	10 sec
2	100	~ 1.6 min
3	1,000	~ 16 min
4	10,000	~ 2.7 hours
5	100,000	~ 1.1 days
6	1,000,000	~ 1.6 weeks
…	…	…
20	1e20	3 trillion years (240x age of the universe)

How many is dimensions is high dimensions?

Dimension reduction
Gaussian processes &
active subspaces (“machine learning”)

Deal with #Points
e.g. Adaptive Sparse Grids
(but you need some shape)

High-performance computing
Reduces time to solution, but not the problem size

How many is dimensions is high dimensions?

Number of parameters (the dimension)	Number of model runs (at 10 points per dimension)	Time for parameter study (at 1 second per run)
1	10	10 sec
2	100	~ 1.6 min
3	1,000	~ 16 min
4	10,000	~ 2.7 hours
5	100,000	~ 1.1 days
6	1,000,000	~ 1.6 weeks
…	…	…
20	1e20	3 trillion years (240x age of the universe)

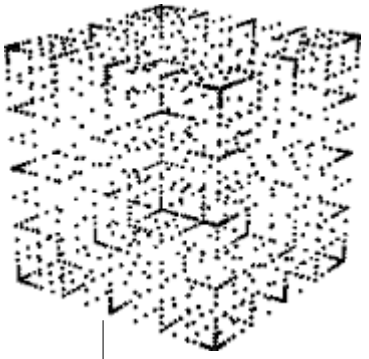
Dimension reduction
Gaussian processes &
active subspaces ("machine learning")

Deal with #Points
e.g. Adaptive Sparse Grids
(but you need some shape)

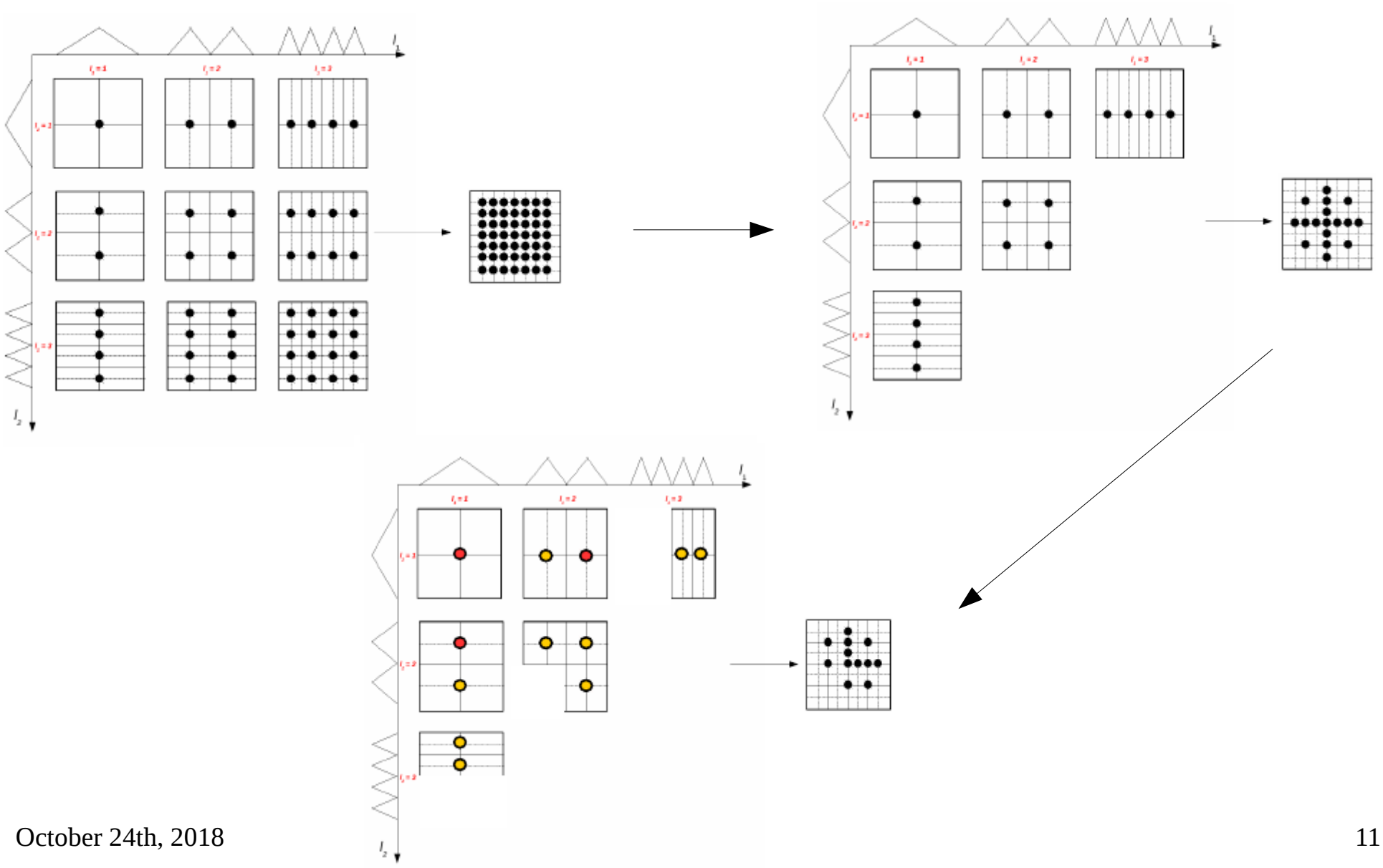
High-performance computing
Reduces time to solution, but not the problem size

Computational modelling

$$\lambda_t \cdot \left[1 + \phi \cdot g^j_{t+1}\right] - \mu^j_t$$
$$- \beta \mathbb{E}_t \left\{ \lambda_{t+1} \left[a^j_{t+1} A \zeta (k^j_{t+1})^{\zeta - 1} + 1 - \delta + \frac{\phi}{2} g^j_{t+2} \left(g^j_{t+2} + 2 \right) \right] - (1 - \delta) \mu^j_{t+1} \right\} = 0,$$
$$0 \leq \mu^j_t \perp \left(k^j_{t+1} - k^j_t (1 - \delta) \right) \geq 0.$$



II. Adaptive Sparse Grids



Interpolation on a Full Grid

-Consider a **1-dimensional function** $f : \Omega \to \mathbb{R}$ **on [0,1]**

-In numerical simulations:
f might be expensive to evaluate! (solve PDEs/system of non-linear Eqs.)
But: need to be able to evaluate *f* at arbitrary points using a numerical code

-Construct an interpolant ***u*** of ***f*** $f(\vec{x}) \approx u(\vec{x}) := \sum_i \alpha_i \varphi_i(\vec{x})$

-With suitable basis functions: $\varphi_i(\vec{x})$
and coefficients: α_i

- For simplicity: focus on case where $f|_{\partial\Omega} = 0$

Basis Functions

-Hierarchical basis based on **hat functions**

$$\phi(x) = \begin{cases} 1 - |x| & \text{if } x \in [-1, 1] \\ 0 & \text{else} \end{cases}$$

-Used to generate a **family of basis functions** $\phi_{l,i}$ having support $[x_{l,i} - h_l, x_{l,i} + h_l]$ by **dilation** and **translation**

$$\phi_{l,i}(x) := \phi\left(\frac{x - i \cdot h_l}{h_l}\right)$$

Hierarchical Increment Spaces

Hierarchical increment spaces:

$$W_l := \operatorname{span}\{\phi_{l,i} : i \in I_l\}$$

with the **index set**

$$I_l = \{i \in \mathbb{N}, 1 \leq i \leq 2^l - 1, i \text{ odd}\}$$

The corresponding function space:

$$V_l = \bigoplus_{k \leq l} W_k$$

The **1d-interpolant:**

$$f(x) \approx u(x) = \sum_{k=1}^{l} \sum_{i \in I_k} \alpha_{k,i} \phi_{k,i}(x)$$

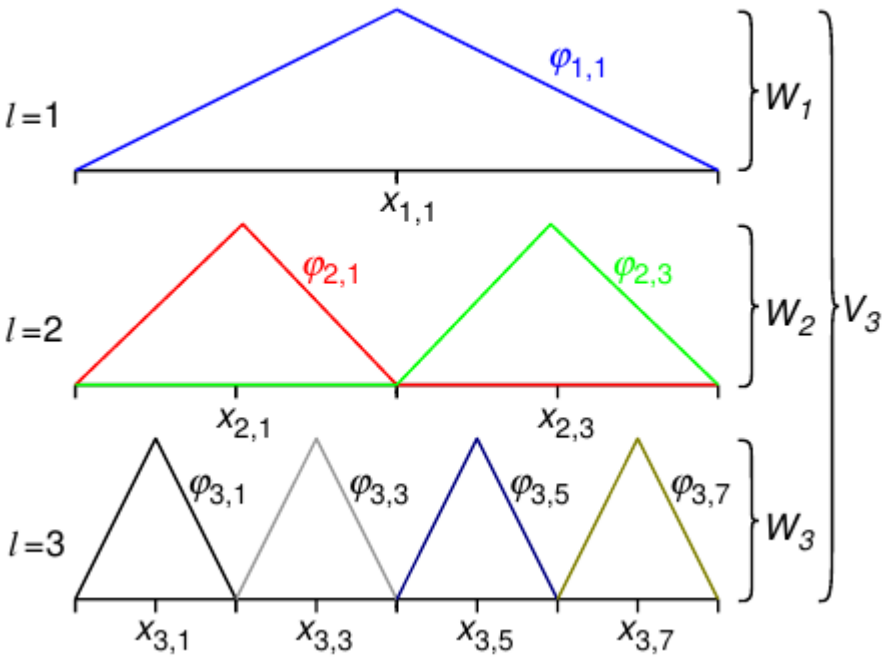


Fig.: **1-d basis functions** $\phi_{l,i}$ and the corresponding **grid points** up level ***l* = 3** in the hierarchical basis.

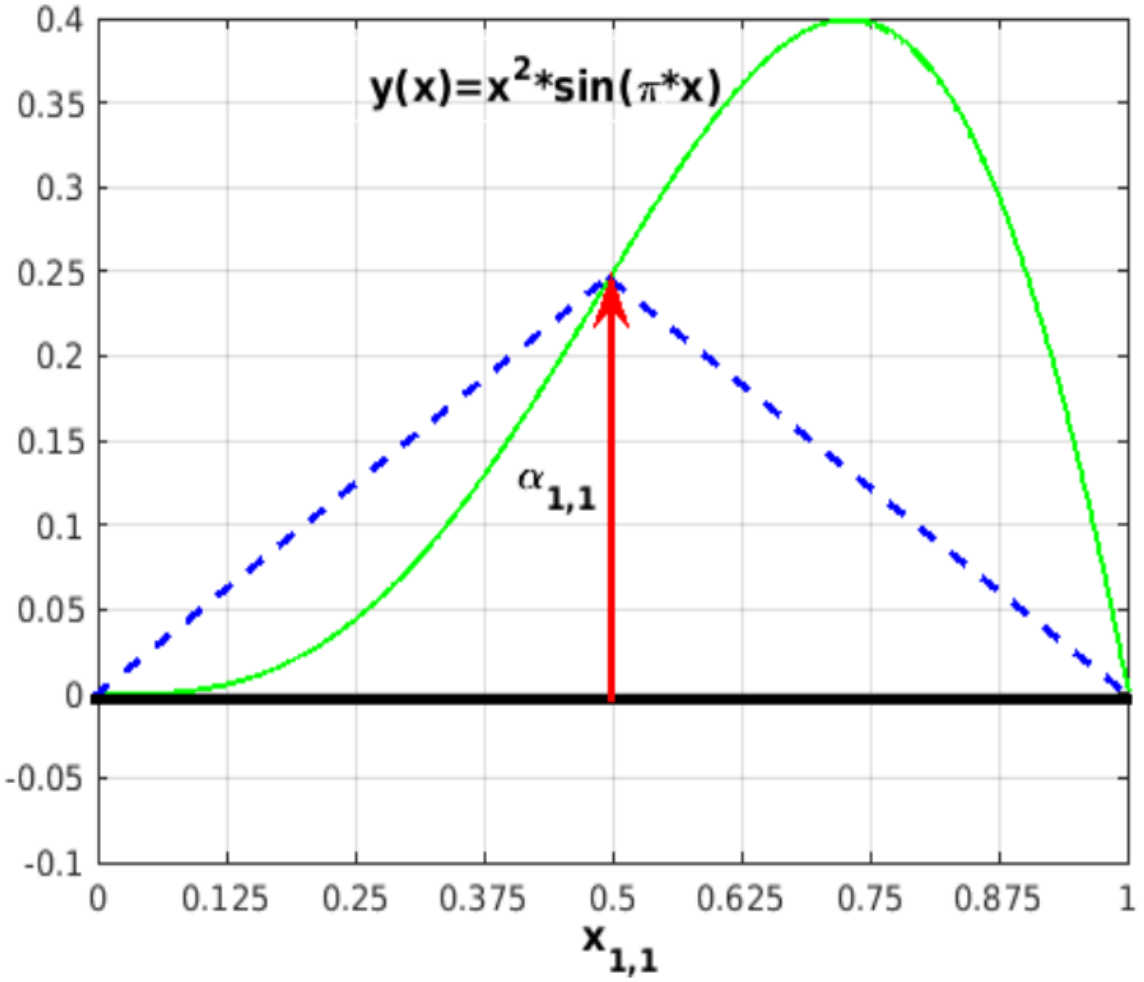
Note: supports of all basis functions of W_k mutually disjoint!

Piecewise Linear Interpolation: Level I

Coefficients:
hierarchical surpluses

They correct the interpolant of level *l*-1 at $\vec{x}_{l,i}$ to the actual value of $f(\vec{x}_{l,i})$

Nested structure:
Evaluate function only at points that are unique to the new level.

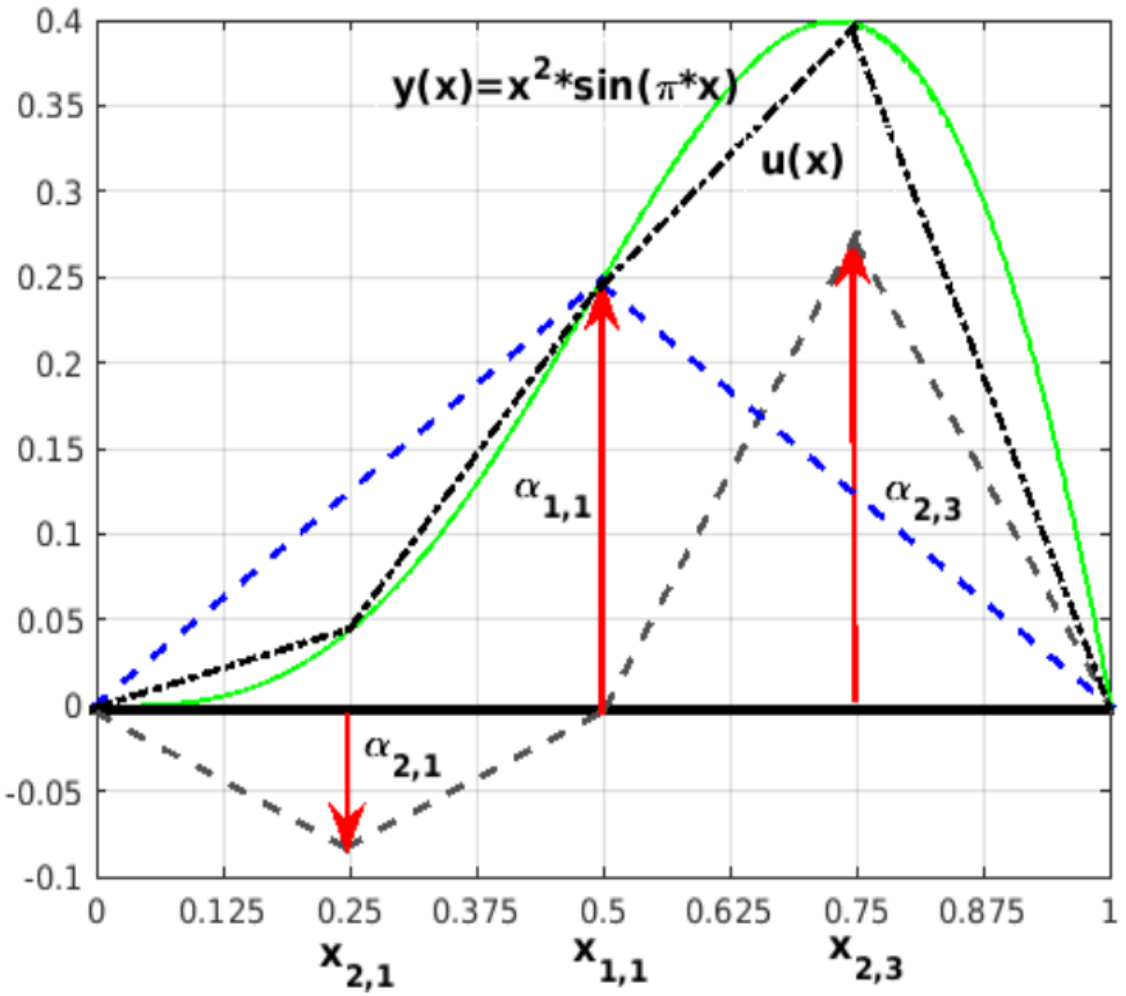


Piecewise Linear Interpolation: Level II

Coefficients:
hierarchical surpluses

They correct the interpolant of level *l*-1 at $\vec{x}_{l,i}$ to the actual value of $f(\vec{x}_{l,i})$

Nested structure:
Evaluate function only at points that are unique to the new level.

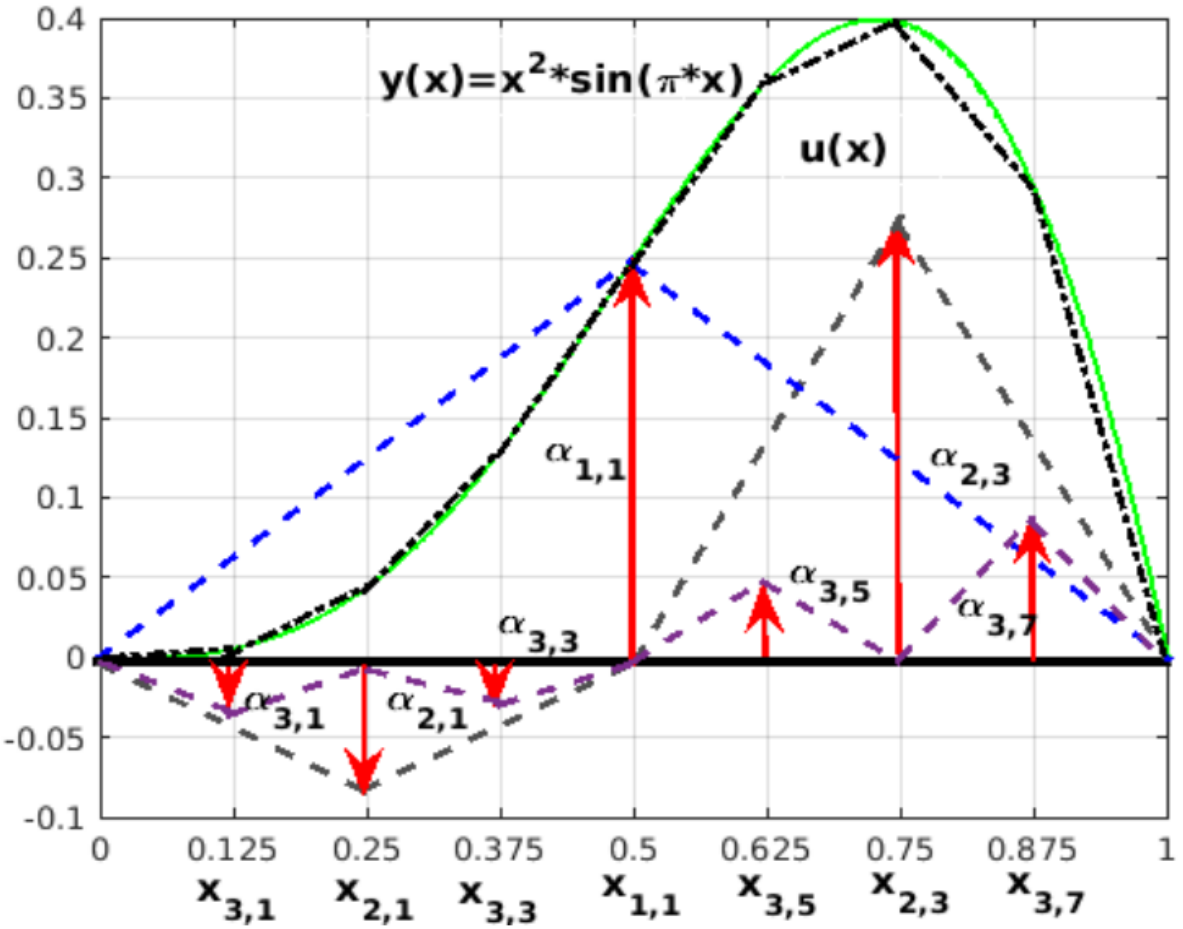


Piecewise Linear Interpolation: Level III

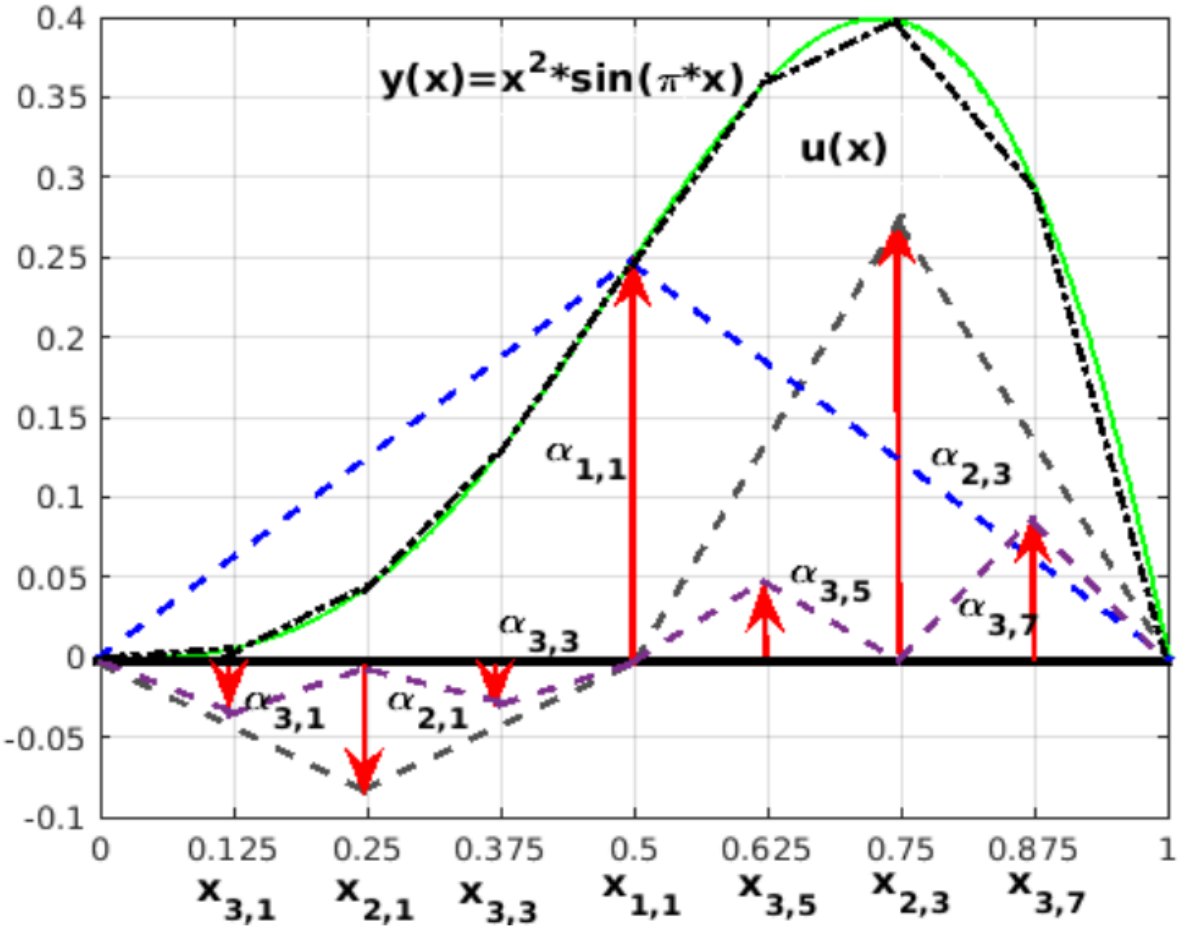
Coefficients:
hierarchical surpluses

They correct the interpolant of level *l*-1 at $\vec{x}_{l,i}$ to the actual value of $f(\vec{x}_{l,i})$

Nested structure:
Evaluate function only at points that are unique to the new level.



MOVIE


y(x)=x²*sin(π*x)
u(x)
α1,1
α2,3
α3,5
α3,7
α3,3
α3,1
α2,1
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
0.125
0.25
0.375
0.5
0.625
0.75
0.875
1
x3,1
x2,1
x3,3
x1,1
x3,5
x2,3
x3,7

Some definitions & notation

(see, e.g. Zenger (1991), Bungartz & Griebel (2004), Garcke (2012), Pflüger (2010),...)

- We will focus on the domain $\Omega = [0,1]^d$

  d: dimensionality; **other domains: rescale**

- introduce **multi-indices**:

  grid refinement level: $\vec{l} = (l_1, ..., l_d) \in \mathbb{N}^d$

  spatial position: $\vec{i} = (i_1, ..., i_d) \in \mathbb{N}^d$

- Discrete, (Cartesian) full grid $\Omega_{\vec{l}}$ on Ω

- Grid $\Omega_{\vec{l}}$ consists of points: $\vec{x}_{\vec{l},\vec{i}} := (x_{l_1,i_1}, ..., x_{l_d,i_d})$

Where $x_{l_t,i_t} := i_t \cdot h_{l_t} = i_t \cdot 2^{-l_t}$ and $i_t \in \{0, 1, ..., 2^{l_t}\}$

Multi-Dimensional Interpolant

Extension to multi-d by a **tensor-product construction**:

Multi-d basis: $$\phi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{t=1}^{d} \phi_{l_t,i_t}(x_t)$$

Index set: $$I_{\vec{l}} := \{\vec{i} : 1 \le i_t \le 2^{l_t} - 1, i_t \text{ odd}, 1 \le t \le d\}$$

Hierarchical increments: $$W_{\vec{l}} := \text{span}\{\phi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\}$$

Multi-d interpolant:

$$\longrightarrow \quad f(\vec{x}) \approx u(\vec{x}) = \sum_{|l|_\infty \le n} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x})$$

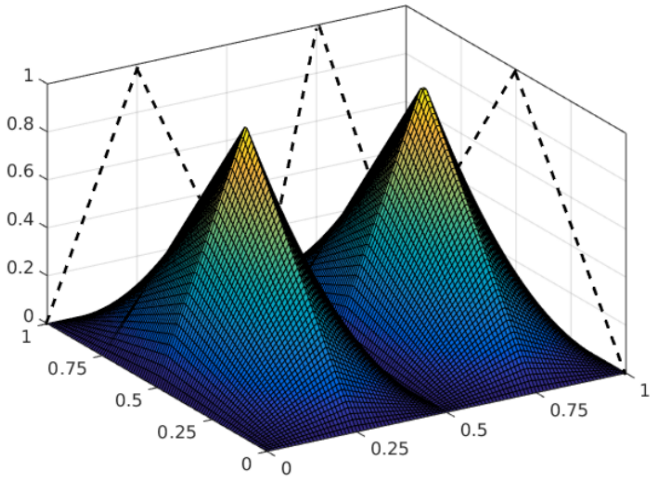


Fig.: Basis functions of the **subspace $W_{2,1}$**

Why reality bites...

Interpolant consists of ***O(2^{nd})*** grid points

For **sufficiently smooth *f*** and its interpolant ***u***, we obtain an asymptotic error decay of $\| f(\vec{x}) - u(\vec{x}) \|_{L_2} \in \mathcal{O}\left(h_n^2\right)$

But at the cost of $\mathcal{O}\left(h_n^{-d}\right) = \mathcal{O}\left(2^{nd}\right)$

function evaluations → **“curse of dimensionality”**

Hard to handle more than 4 dimensions numerically

⟶ e.g. d=10, n = 4, 15 points/d, **5.8 x 10^{11}** grid points

`Breaking' the curse of dimensionality I

Question: **“can we construct discrete approximation spaces that are better in the sense that the same number of invested grid points leads to a higher order of accuracy?”** **YES √**

(see, e.g. Bungartz & Griebel (2004))

→ If **second mixed derivatives are bounded**, then the hierarchical **surpluses decay rapidly** with increasing approximation level.

$$|\alpha_{\vec{l},\vec{i}}| = \mathcal{O}\left(2^{-2|\vec{l}|_1}\right)$$

`Breaking' the curse of dimensionality II

(see, e.g. Bungartz & Griebel (2004))

Strategy of constructing sparse grid: leave out those subspaces from full grid that only contribute little to the overall interpolant.

Optimization w.r.t. number of degrees of freedom (grid points) and the approximation accuracy leads to the sparse grid space of level ***n***.

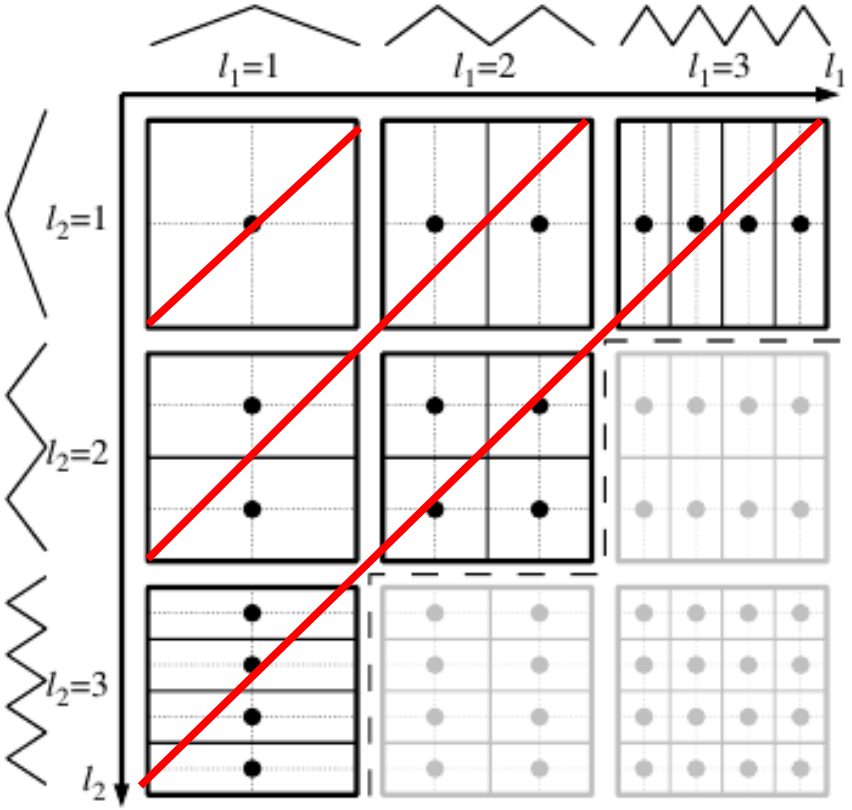
$$V_{0,n}^S := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}$$

Interpolant: $$f_{0,n}^S(\vec{x}) \approx u(\vec{x}) = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x})$$

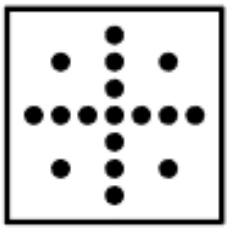
grid points: $\mathcal{O}\left(h_n^{-1} \cdot \left(\log(h_n^{-1})\right)^{d-1}\right) = \mathcal{O}\left(2^n \cdot n^{d-1}\right)$ << $\mathcal{O}\left(h_n^{-d}\right) = \mathcal{O}\left(2^{nd}\right)$

Accuracy of the interpolant: $\mathcal{O}\left(h_n^2 \cdot \log(h_n^{-1})^{d-1}\right)$ vs. $\mathcal{O}\left(h_n^2\right)$

Sparse grid construction in 2D



$$V_{0,n}^S := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}$$



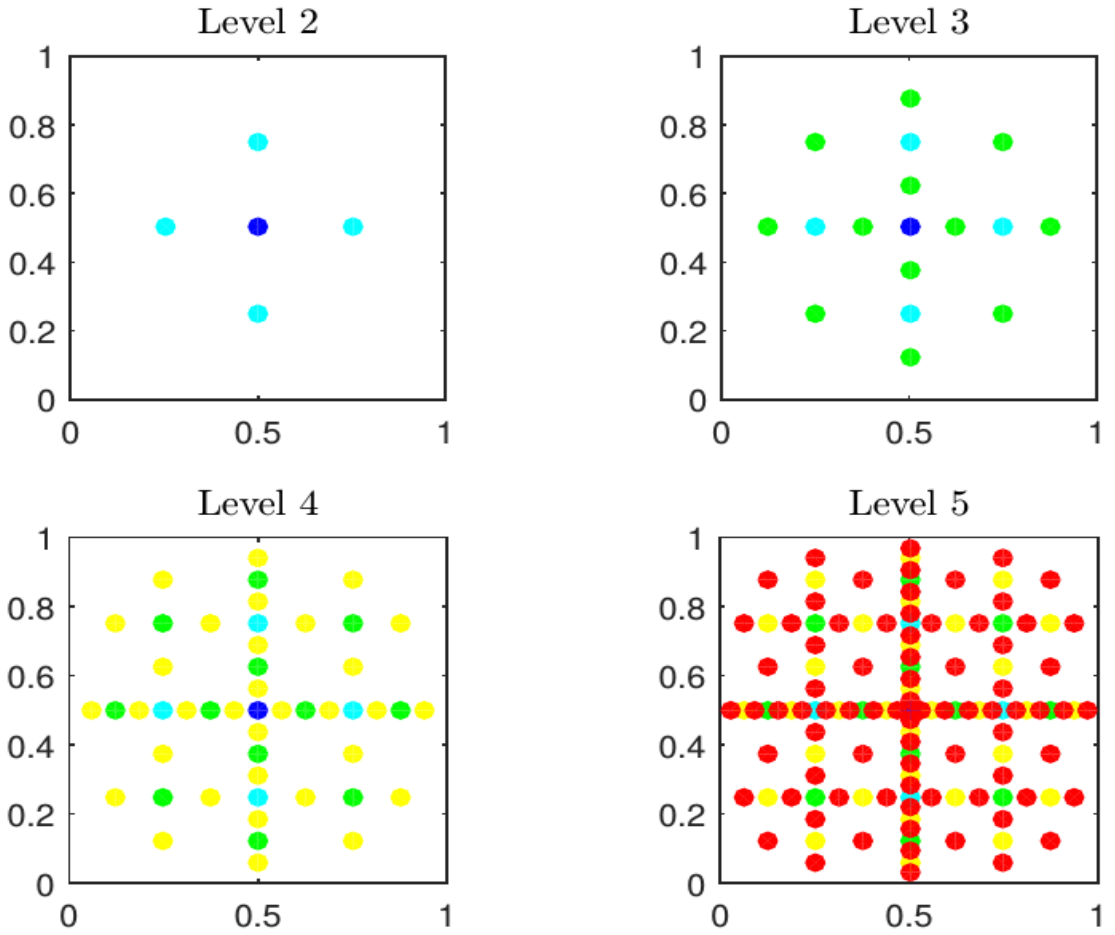
V_3

Sparse grid: 17 pt.
Full grid : 49 pt.

Fig.: Two-dimensional subspaces $\mathbf{W_l}$ up to ***l***=3 (h_3 = 1/8) in each dimension.

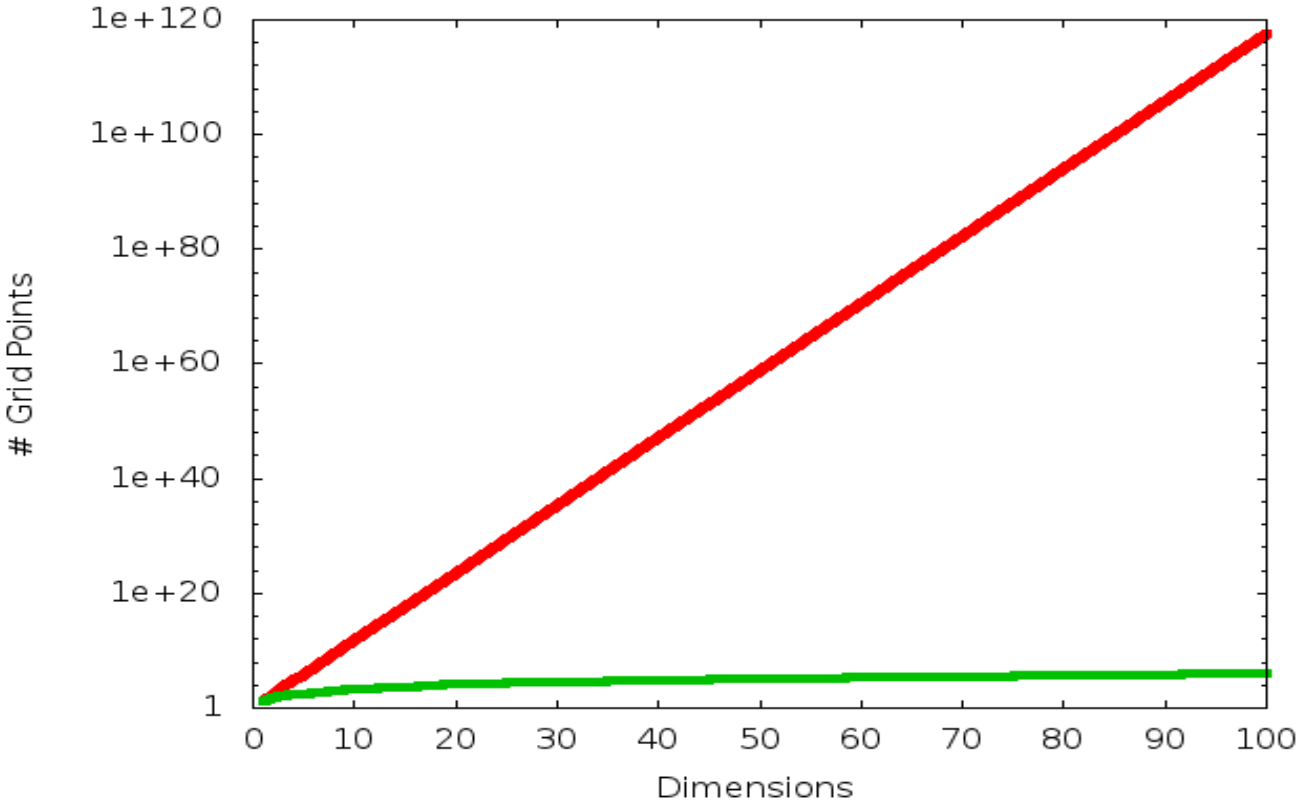
The optimal a priori selection of subspaces in shown in black **(left)** and the Corresponding sparse grid of level n = 3 **(right)**.
For the full grid, the gray subspaces have to be used as well.

Sparse Grid of Increasing level



Grid Points

d	$\lvert V_n \rvert$	$\lvert V_{0,n}^S \rvert$
1	15	15
2	225	49
3	3375	111
4	50'625	209
5	759'375	351
10	$5.77 \cdot 10^{11}$	2'001
15	$4.37 \cdot 10^{17}$	5'951
20	$3.33 \cdot 10^{23}$	13'201
30	$1.92 \cdot 10^{35}$	41'601
40	$1.11 \cdot 10^{47}$	95'201
50	$6.38 \cdot 10^{58}$	182'001
100	>Googol	1'394'001



Tab.: Number of grid points for several types of sparse grids of level $n = 4$.
Middle: Full grid; **right:** **classical sparse grid** with **no points at the boundaries.**
Fig.: Number of grid points growing with dimension (full grid vs. sparse grid).

Hierarchical Integration

High-dimensional integration easy with sparse grids, e.g. compute expectations
Let's assume uniform probability density:

$$\mathbb{E}\left[u(\vec{x})\right] = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \int_{\Omega} \phi_{\vec{l},\vec{i}}(\vec{x}) d\vec{x}$$

The one-dimensional integral can now be computed analytically (Ma & Zabras (2008))

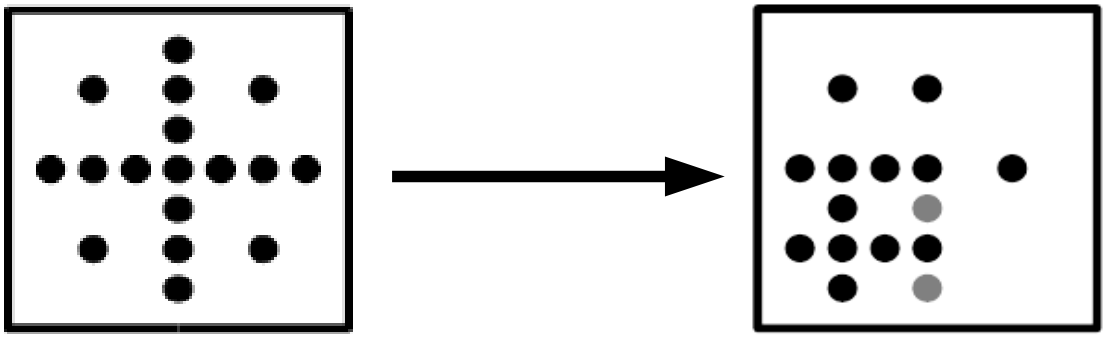
$$\int_0^1 \phi_{l,i}(x)\, dx = \begin{cases} 1, & \text{if } l = 1 \\ \frac{1}{4} & \text{if } l = 2 \\ 2^{1-l} & \text{else} \end{cases}$$

Note that this result is independent of the location of the interpolant to dilation
And translation properties of the hierarchical basis functions.
→ **Multi-d integrals are therefore again products of 1-d integrals.**

We denote $\int_{\Omega} \phi_{l,i}(\vec{x})\, d\vec{x} = J_{\vec{l},\vec{i}}$

$$\longrightarrow \mathbb{E}\left[u(\vec{x})\right] = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot J_{\vec{l},\vec{i}}$$

Adaptive Sparse Grids



Sketch of adaptive refinement

See, e.g. Ma & Zabaras (2008), Pflüger (2010), Bungartz (2003),..

-Surpluses should quickly decay to zero

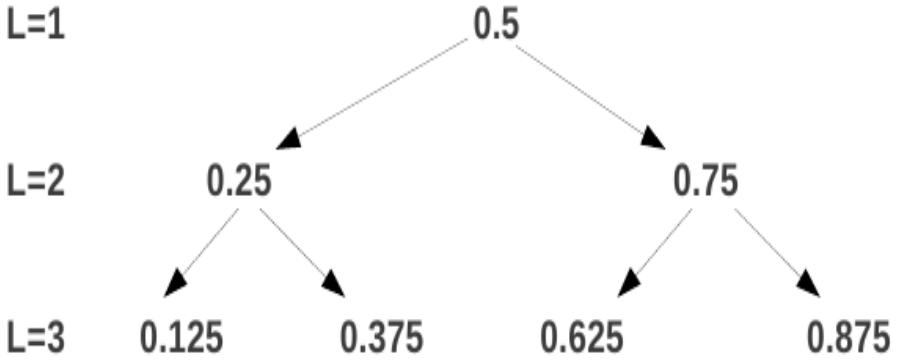
-Use hierarchical surplus as error indicator.

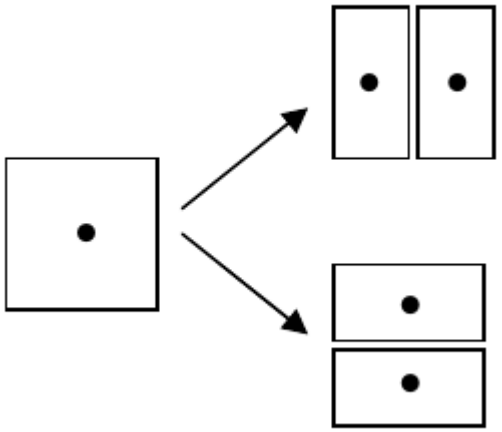
-**Automatically detect "discontinuity regions"** and adaptively refine the points in this region.

-Each grid point has ***2d*** neighbours

-Add neighbour points, i.e. locally refine interpolation level from *l* to *l+1*

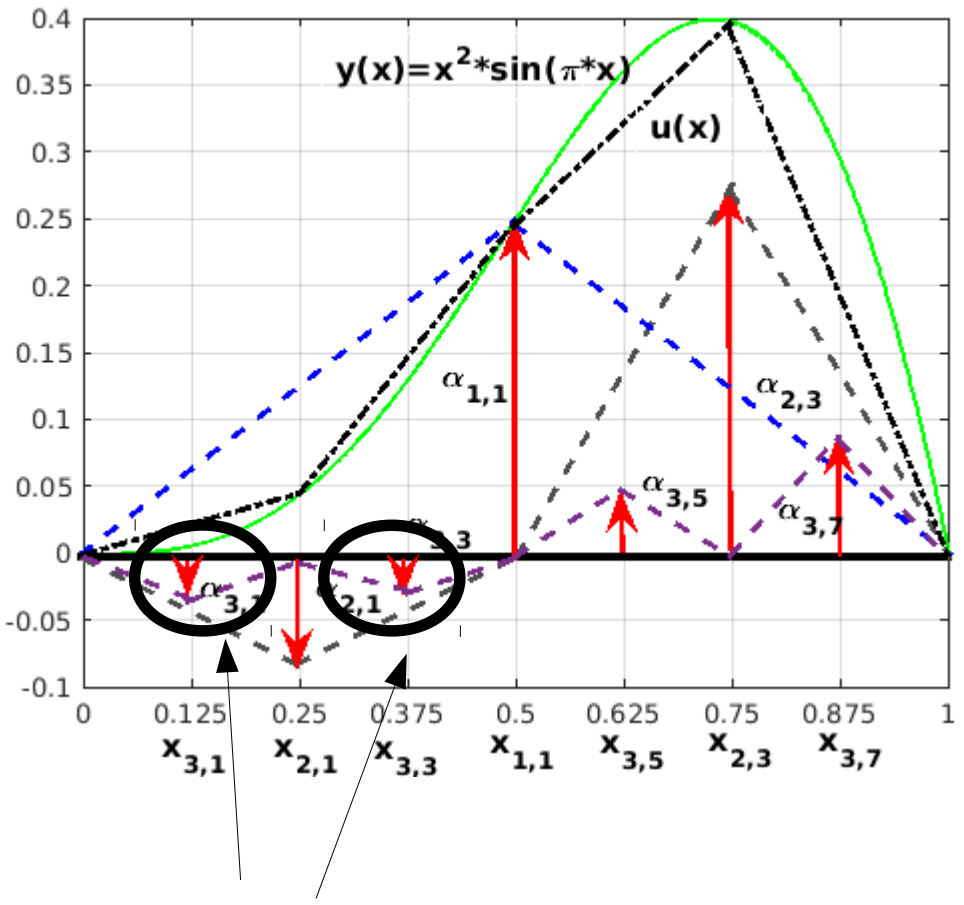
-Criterion: **e.g.** $|\alpha_{\vec{l},\vec{i}}| \geq \epsilon$





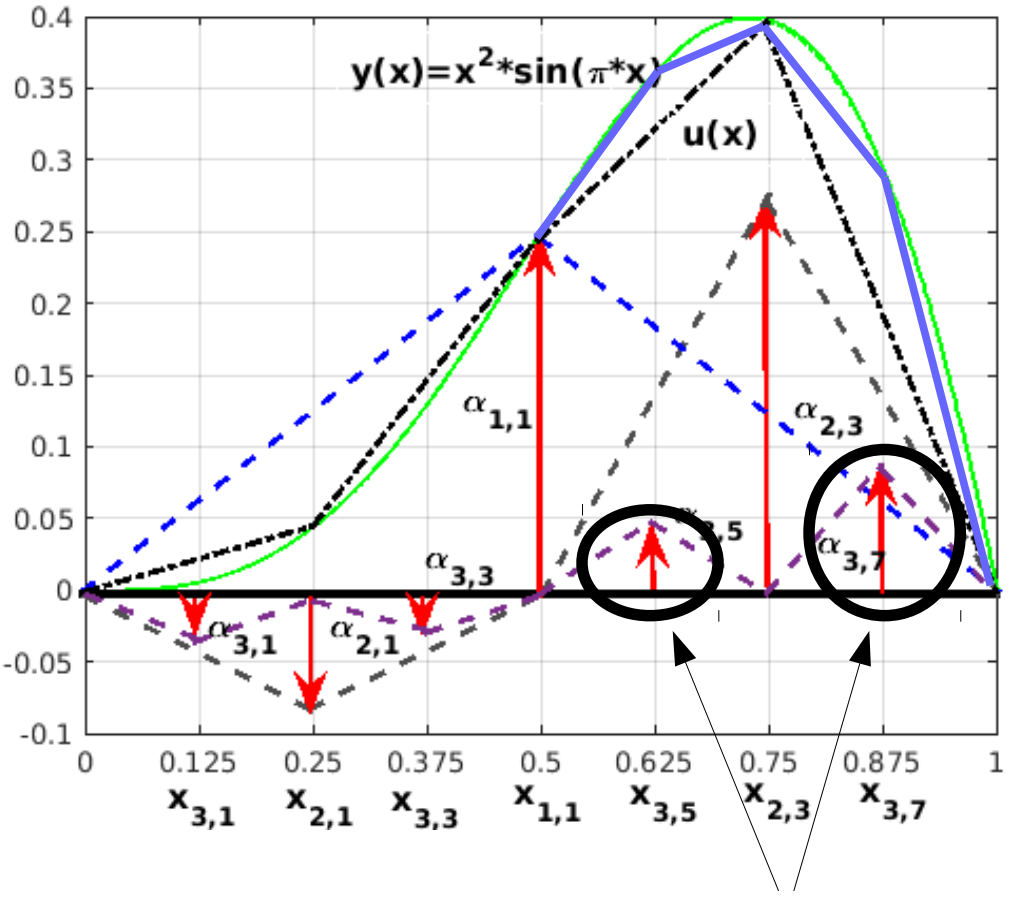
top panel: tree-like structure of sparse grid.
lower panel: locally refined sparse grid in 2D.

Example I



Small – below threshold

Example II



Add points – above threshold

Test in 1d

(See Genz (1984) for test functions)

Test function:

$$f(x) = \frac{1}{|0.5 - x^4| + 0.01}$$

Error both for full grid and adapt. sparse grid of $O(10^{-2})$.

Error measure:

→ 1000 random points from [0,1]

$$e = \max_{i=1,\ldots,1000} |f(\vec{x}_i) - u(\vec{x}_i)|$$

Full grid: **1023** points
Adaptive sparse grid: **109** points.

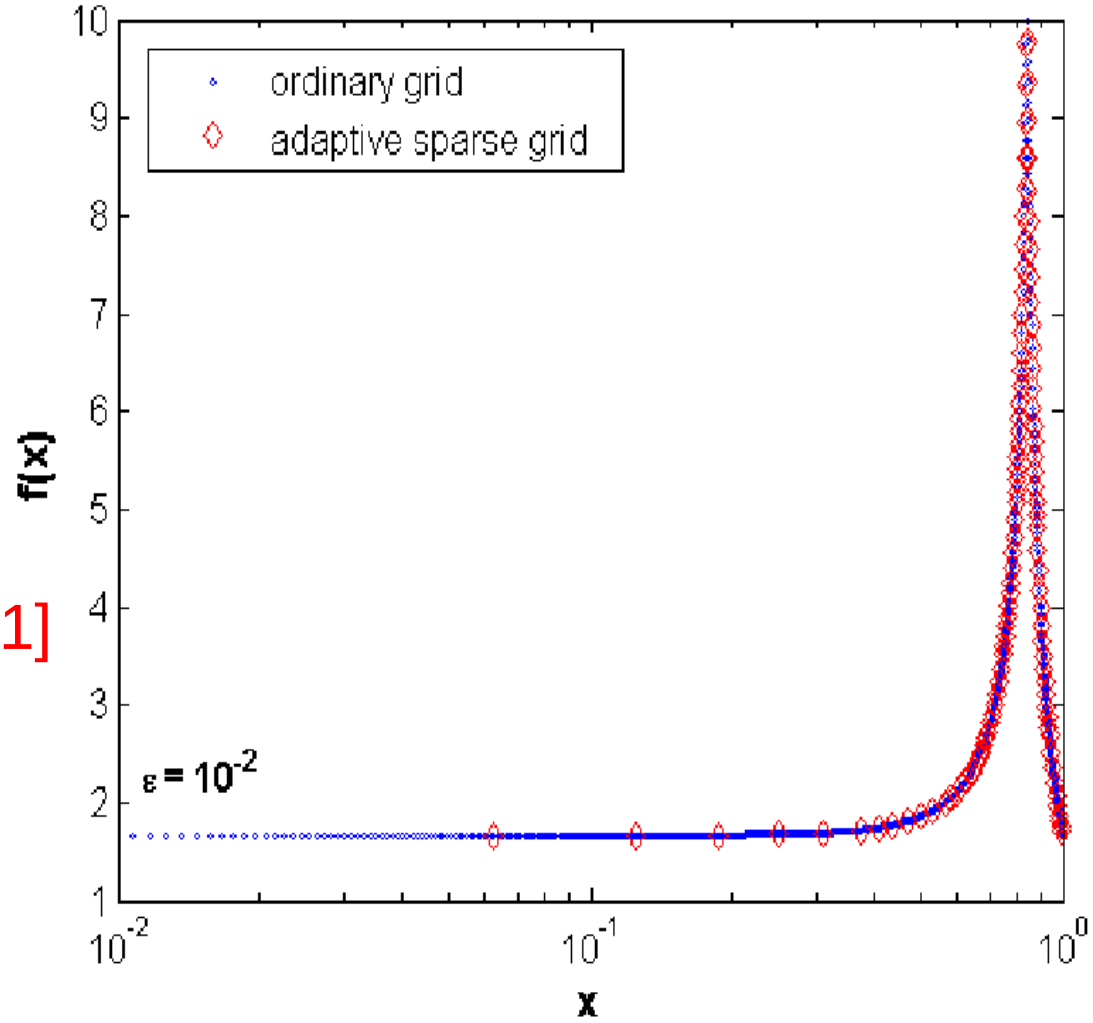


Fig.: Blue: Full grid;red: adaptive sparse grid.

Test in 2d

Test function: $$\frac{1}{|0.5 - x^4 - y^4| + 0.1}$$

Error: $O(10^{-2})$

Full grid:
→ $O(10^9)$ points

Sparse grid:
→ **311,297 points**

Adaptive sparse grid:
→ **4,411 points**

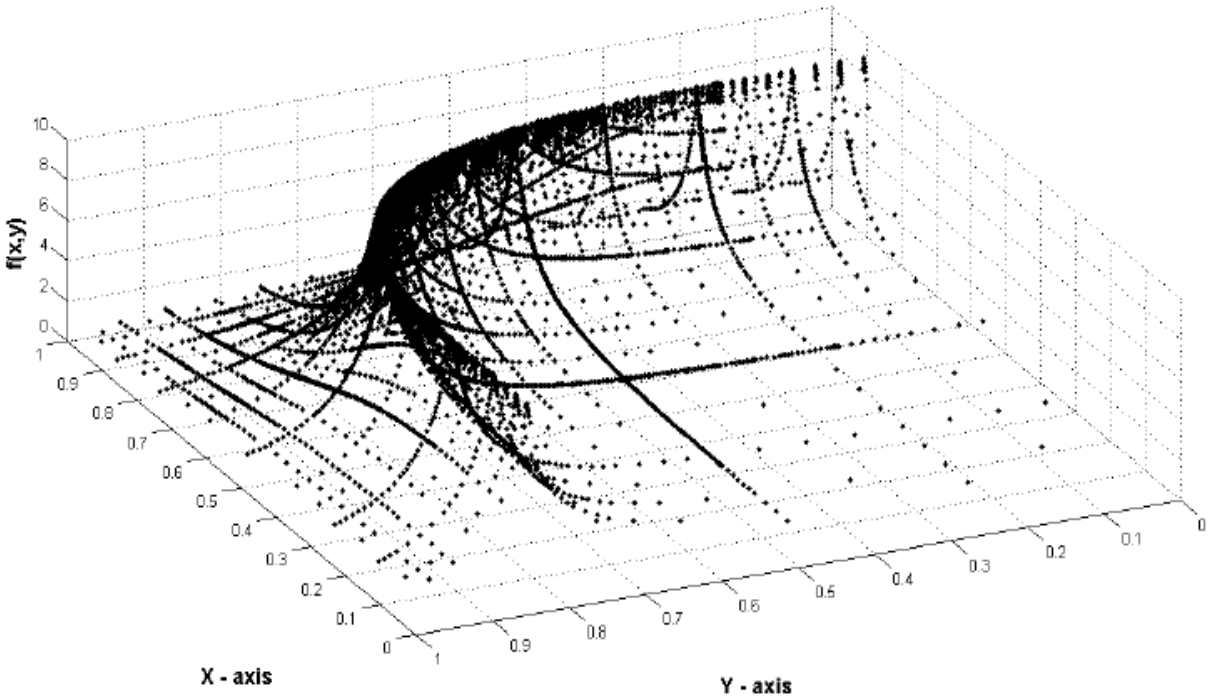


Fig.: 2d test function and its corresponding grid points after 15 refinement steps.

Movie

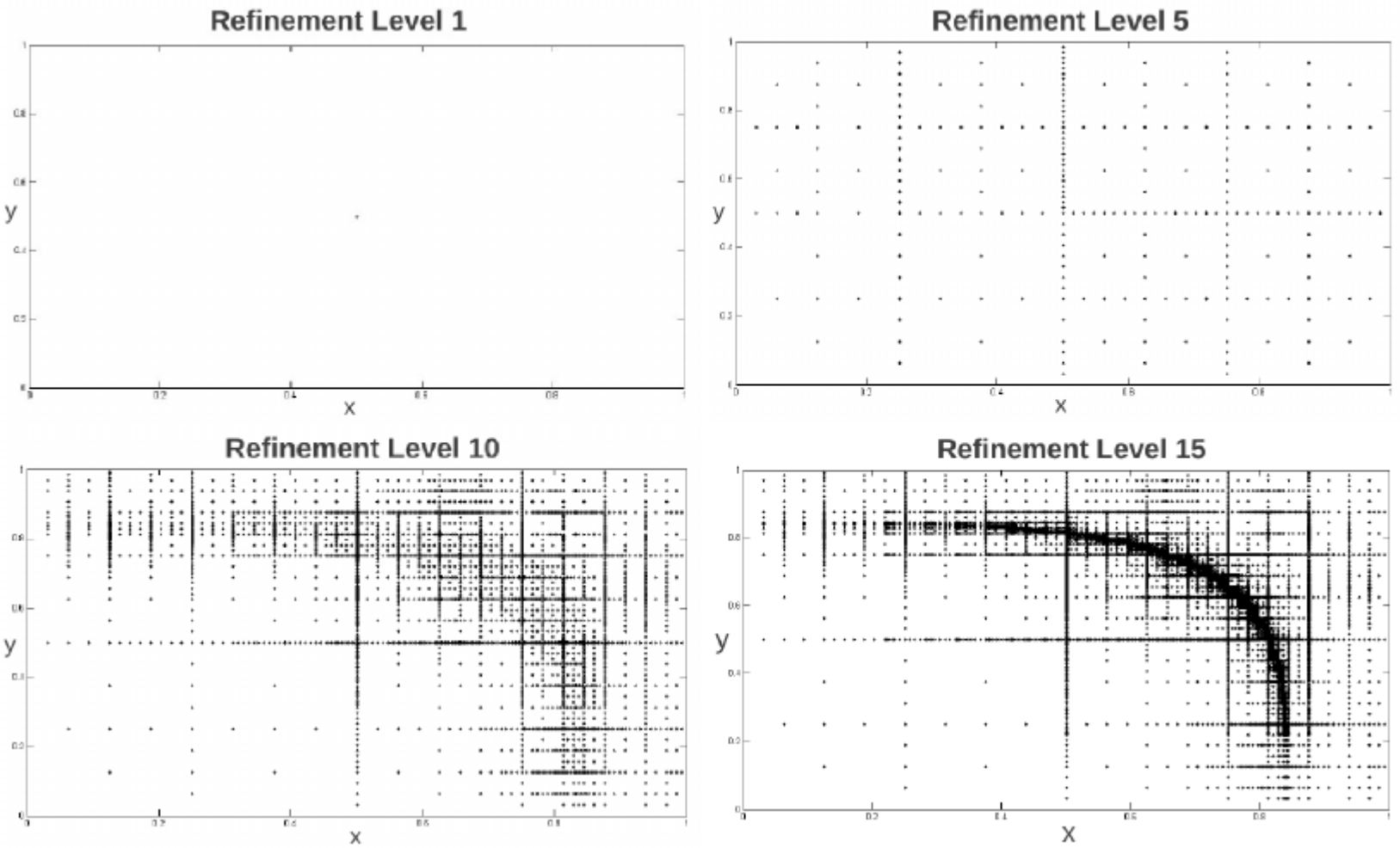
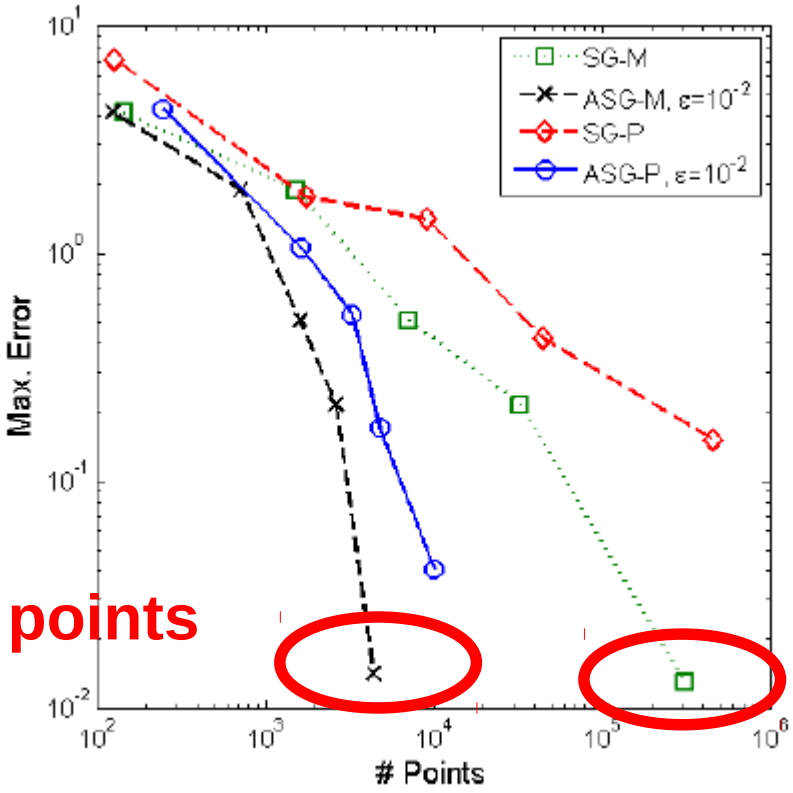


Fig.: Evolution of the adaptive sparse grid with a **threshold for refinement of 10^{-2}** . The refinement levels displayed are L = 1, 5, 10, 15.

Convergence



4,411 points

311,297 points

Fig.: Comparison of the interpolation error for **conventional and adaptive sparse grid interpolation** (two different adaptive sparse grid choices).

Time-iteration with adaptive sparse grids

Data: Initial guess $p_{next} = (p_{next}(1), \ldots, p_{next}(N_s = 16)$ for next period's policy function. Approximation accuracy $\bar{\eta}$. Maximal refinement level L_{max}. Starting refinement level $L_0 \leq L_{max}$. Refinement threshold ϵ.

Result: The (approximate) equilibrium policy function $p = (p(1), \ldots, p(N_s = 16))$.

while $\eta > \bar{\eta}$ **do**

  Set $z = 1$.

  for $z \leq N_s$ **do** → ***Stochastic states: independent adaptive sparse grids***

    Set $l = 1$, set $G(z) \subset S(z)$ to be the level 1 grid on $S(z)$, and set $G_{old}(z) = \emptyset, G_{new}(z) = \emptyset$.

    while $G(z) \neq G_{old}(z)$ **do**

      for $g(z) \in G(z) \setminus G_{old}(z)$ **do** → ***Grid points within a refinement level: independent***

        Compute the optimal policies $p(g(z))$ by evaluating (5) to (15) given next period's policy p_{next}.

        Define the policy $\tilde{p}(g((z)))$ by interpolating $\{p(g(z))\}_{g(z) \in G_{old}(z)}$.

        if $(l < L_{max}$ *and* $\|p(g(z)) - \tilde{p}(g(z))\|_\infty > \epsilon)$ *or* $l < L_0$, **then**

          Add the neighboring points (*sons*) of $g(z)$ to $G_{new}(z)$.

        end

      end

      Set $G_{old}(z) = G(z)$, set $G = G_{old}(z) \cup G_{new}(z)$, set $G_{new}(z) = \emptyset$, and set $l = l + 1$.

    end

    Define the policy function $p(z)$ as the sparse grid interpolation of $\{p(g(z))\}_{g(z) \in G(z)}$.

    Calculate (an approximation for) the error within a state:

    $\eta(z) = \|p(z) - p_{next}(z)\|$. Set $p_{next}(z) = p(z)$.

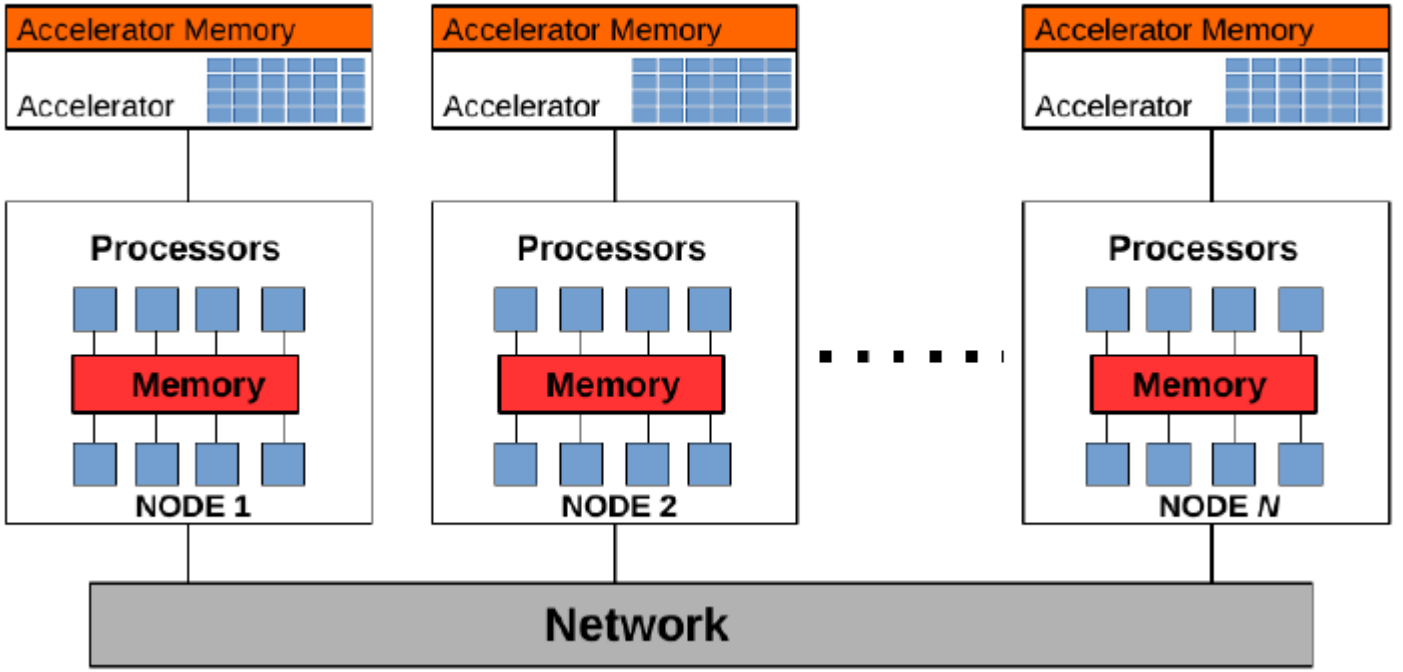
    set $z = z + 1$.

  end

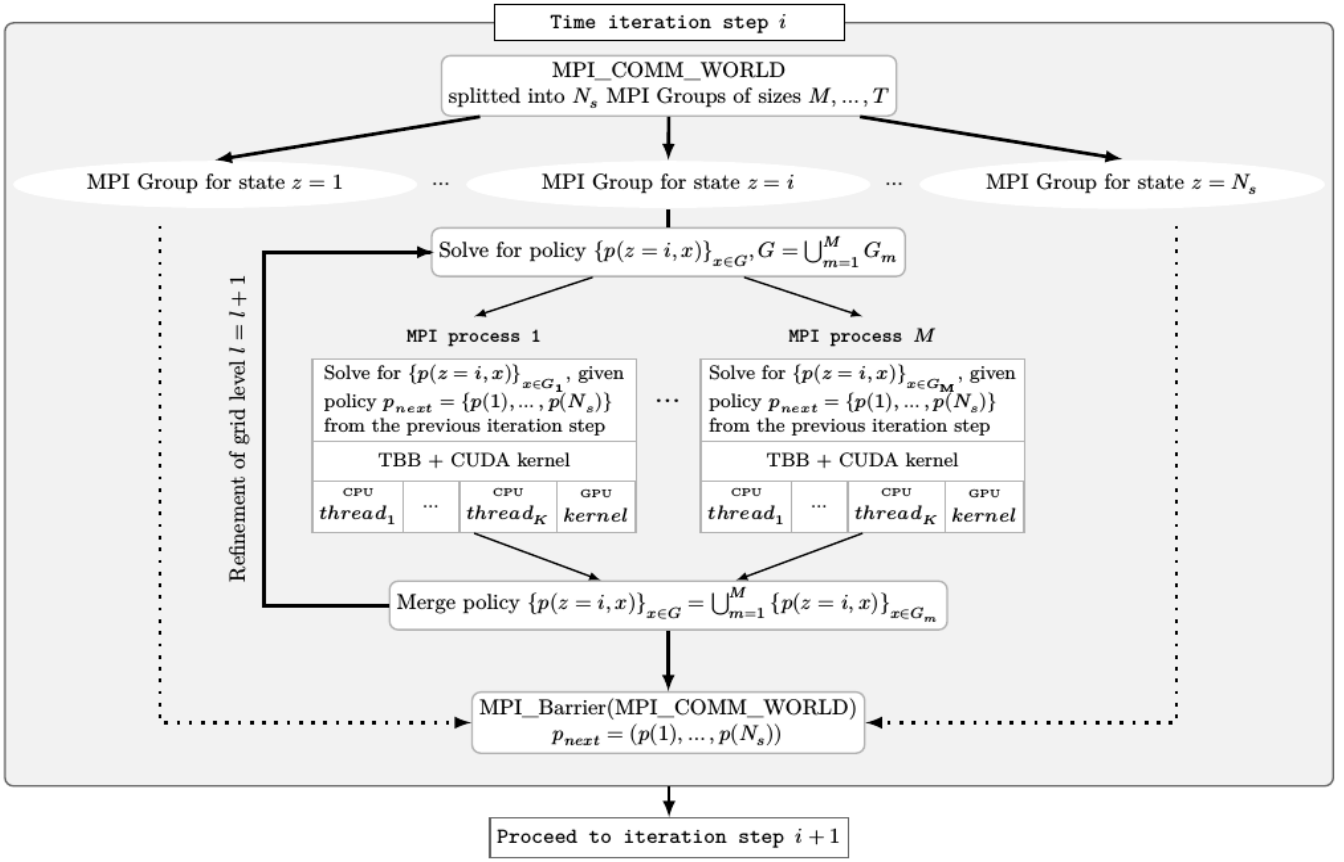
  error: $\eta = \max(\eta(1), \ldots, \eta(N_s))$

end

Today's HPC systems



Overall parallelization scheme



→ At every grid point in every state, a non-linear system of equations needs to be solved.

→ Sizes of the individual adaptive sparse grids may be very different.

→ We need to carefully ensure workload balance.

Single node parallelization

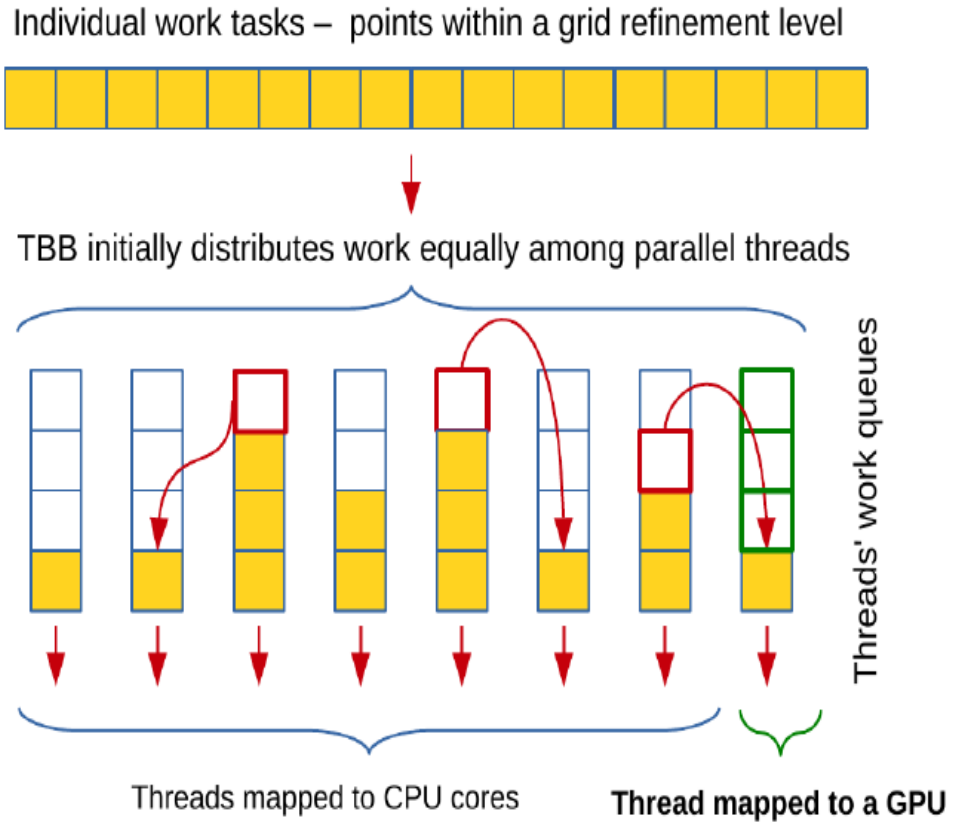
-TBB maps different threads, similar to OpenMP.

-Every thread is initially assigned an equal logical queue of tasks.

-However, different tasks may be processed faster or slower, due to differences between tasks and/or compute cores

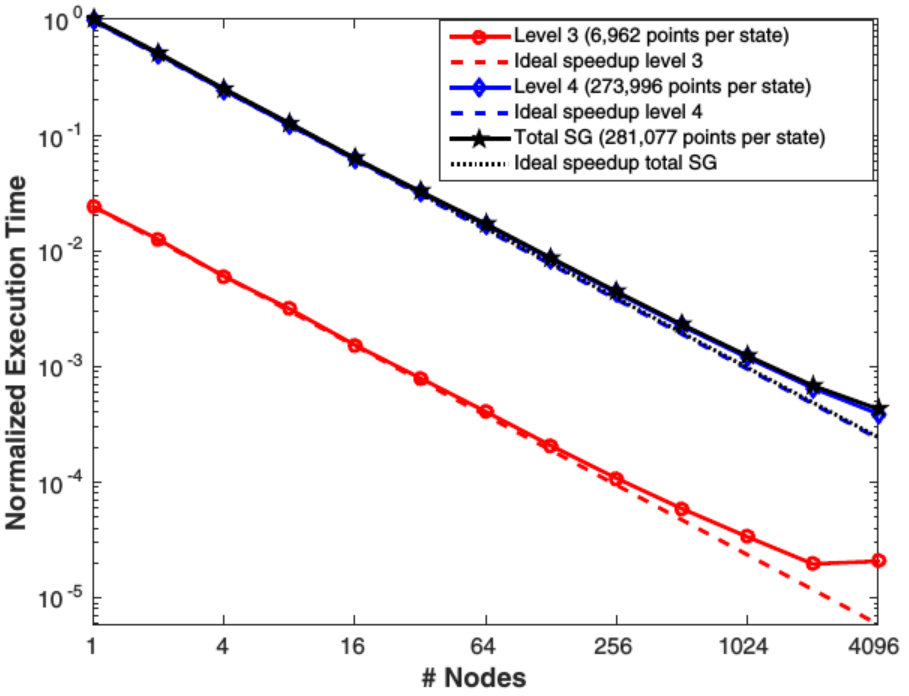
-TBB approach to work balancing: once one thread runs out of tasks, **“steal” a task from another thread, which makes slower progress**.

-We map one extra thread onto **GPU** if present.
→ **CPU** cores and **GPU** process interpolation tasks together.



Strong scaling on “Piz Daint” at CSCS

- Test on Cray XC50
- 16 x 281,077 = 4,497,232 points.
- 265,336,688 unknowns.
- 70% efficiency on 4,096 nodes.
- Speed-up limitations: few points in lower grid levels.



Piz Daint: ~25 Petaflop/s

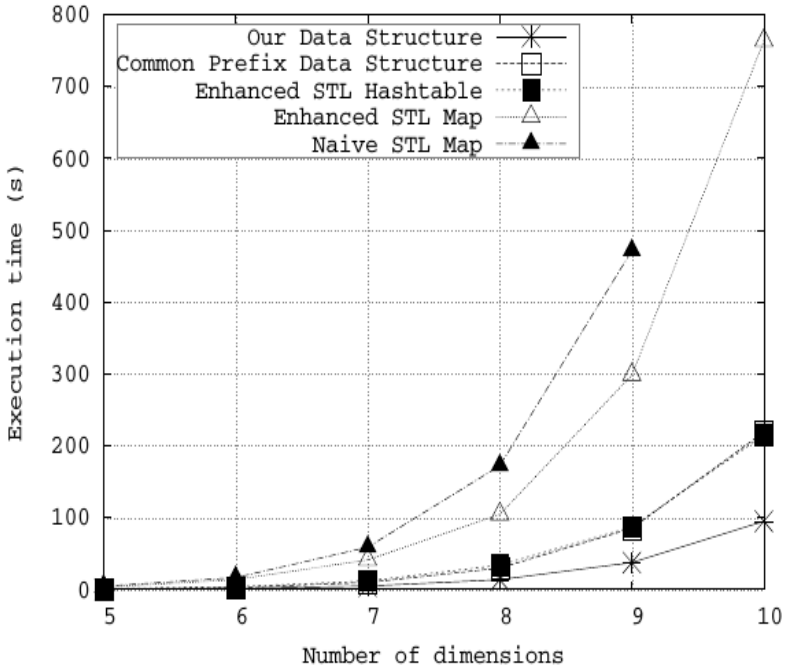
Results for Smooth IRBC Model

Non-adaptive sparse grid of fixed level produces stable accuracy when dimension is increased massively:

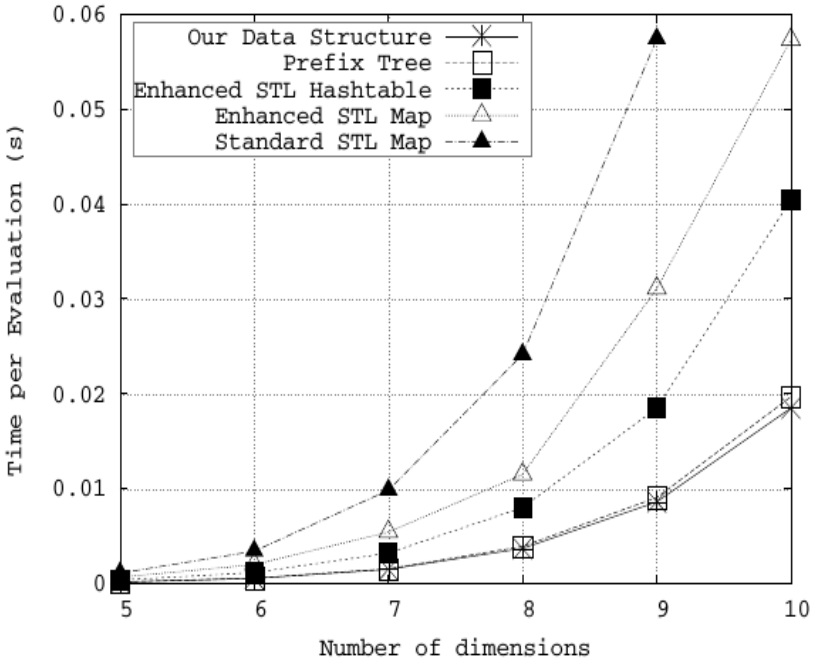
Dimension	Level	Points	Max. Error	Avg. Error
4	3	41	-2.95	-3.18
12	3	313	-2.81	-3.27
20	3	841	-2.93	-3.30
50	3	5,101	-2.64	-3.33
100	3	20,201	-2.79	-3.33
4	4	137	-3.04	-3.65
12	4	2,649	-3.04	-3.83
20	4	11,561	-3.00	-3.73

All errors are given in log 10 -scale.

Execution times on sparse grids



(a) Runtime for sequential hierarchization.



(b) Runtime for sequential evaluation.

→ going to higher dimensions gets polynomially harder

Figs. From Murarasu et al. (2011)

Limitations of sparse grids (II)

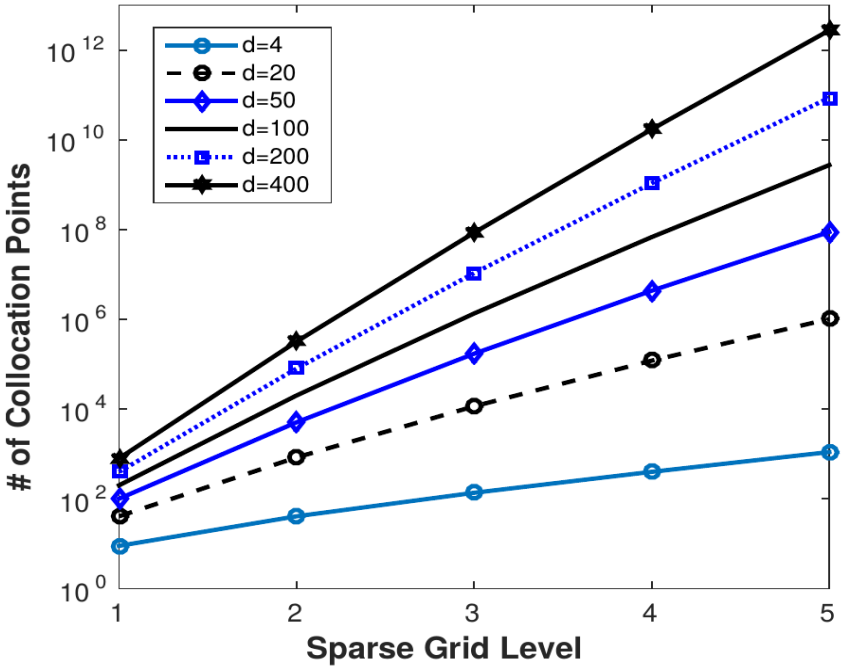


Fig.: classical sparse grids of varying dimension and increasing refinement level

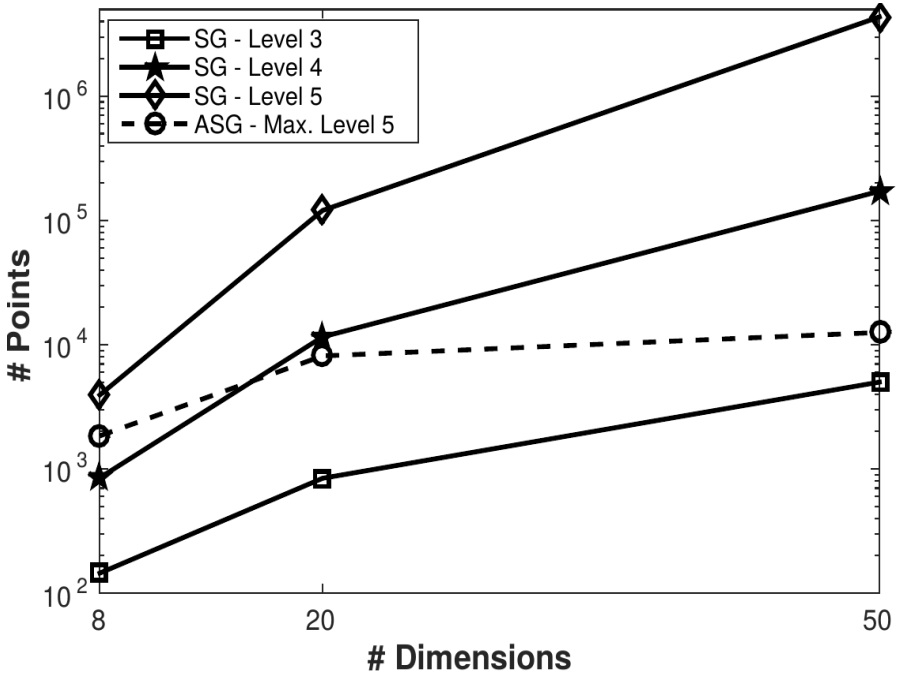
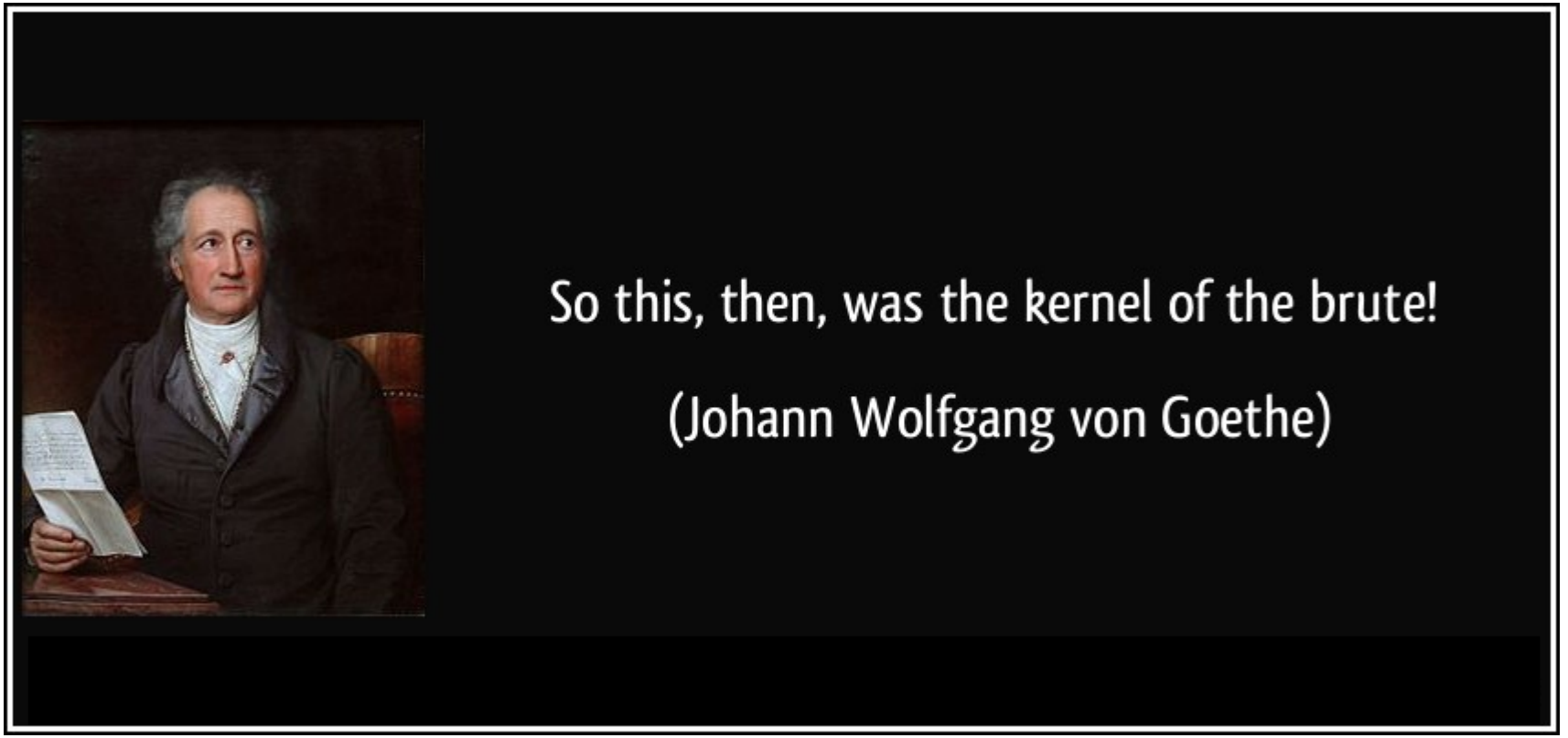


Fig.: IRBC model, solved both with classical sparse grids of varying dimension and increasing refinement level.

Major issue: a complex problem may require a high resolution in order to obtain a "reasonable" solution, i.e., a high sparse grid refinement level. For high-dimensional problems, the amount of points added to the sparse grid grow fast with the increasing level (still slower than exponential) but still make problems quickly intractable **(left panel)**. ASGs can alleviate this issue to some extend **(right panel)**.

Second issue: Geometric restrictions (we need to map our problem onto a hypercube→ **See Felix' talk for GPR**).

3. HDMR


So this, then, was the kernel of the brute!
(Johann Wolfgang von Goethe)

Dimensional Decomposition

see, e.g., Ma & Zabaras (2010), Yang et al. (2012), Rabitz & Alis (1999), Holtz (2011)...

Motivated by idea that for many situations, **only relatively low order correlations* among dimensions will significantly impact solution**.

→ Split multi-dimensional function of dimension ***N*** into its contributions from different groups of sub-dimensions.

→ We want to approximate $f(\mathbf{Y}) : \mathbb{R}^N \rightarrow \mathbb{R}$ by the following representation:

$$f(\mathbf{Y}) = f_0 + \sum_{s=1}^{N} \sum_{i_1 < \cdots < i_s}^{N} f_{i_1 \cdots i_s}(Y_{i_1}, \ldots, Y_{i_s})$$

Where the interior sum is over all sets of s integers $i_1, \ldots, i_s$, that satisfy $1 \leqslant i_1 < i_s \leqslant N$.

$$f(\mathbf{Y}) = f_0 + \sum_{i=1}^{N} f_i(Y_i) + \sum_{1 \leqslant i_1 < i_2 \leqslant N} f_{i_1 i_2}(Y_{i_1}, Y_{i_2}) + \cdots + \sum_{1 \leqslant i_1 < \cdots < i_s \leqslant N} f_{i_1 \cdots i_s}(Y_{i_1}, \ldots, Y_{i_s}) + \cdots + f_{12 \cdots N}(Y_1, \ldots, Y_N).$$

zeroth-order component "mean effect"

Univariate function; individual contribution to the output

s-th order component function

*Term "correlation" loosely used; namely in the context of the impact of input variables upon the output when acting together

Cut-HDMR

(e.g. , Sobol (2002), Ma & Zabaras (2010), Yang et al (2012), and references therein)

A computationally efficient way of constructing the HDMR expansion is the so-called “cut-HDMR”.

The notation $\mathbf{Y} = \overline{\mathbf{Y}} \setminus \mathbf{Y_u}$ means that the components of Y other than those indices that belong to the set **u** are set equal to those of the reference point.

→ Components are |u|-dimensional functions where the unknown variables are those dimensions whose indices belong to u.

→ The component functions of CUT-HDMR are explicitly given as follows:

$$f_0 = f(\overline{\mathbf{Y}}), \quad f_i(Y_i) = f(\mathbf{Y})|_{\mathbf{Y}=\overline{\mathbf{Y}}\setminus Y_i} - f_0$$
$$f_{ij}(Y_i, Y_j) = f(\mathbf{Y})|_{\mathbf{Y}=\overline{\mathbf{Y}}\setminus(Y_i,Y_j)} - f_i(Y_i) - f_j(Y_j) - f_0, \ldots$$

In general, the component functions for $\mathbf{x_u}$ are

$$f_{\mathbf{u}}(\mathbf{x_u}) = f(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}\setminus\mathbf{x_u}} - \sum_{\mathbf{v}\subset\mathbf{u}} f_{\mathbf{v}}(\mathbf{x_v}), \quad \text{with } f_0 = f(\bar{\mathbf{x}})$$

Choice of a reference point

see, e.g., Sobol (2002)

The convergence property of HDMR is rather sensitive to the choice of **the reference point, a so-called anchor point** $\overline{\mathbf{Y}}$

→ A good reference point should satisfy:

$$\min_{\mathbf{Y}\in\Gamma} |f(\overline{\mathbf{Y}}) - \mathbb{E}[f(\mathbf{Y})]|$$

Often, the mean of the output is **not** known a priori.

→ To this end, it was proposed to **sample a moderate number of random inputs** and compute the mean of the sample outputs.

→ Then the reference point is chosen as the one among the samples whose output is the closest to the above mean value.

HDMR expansion in large dimensions

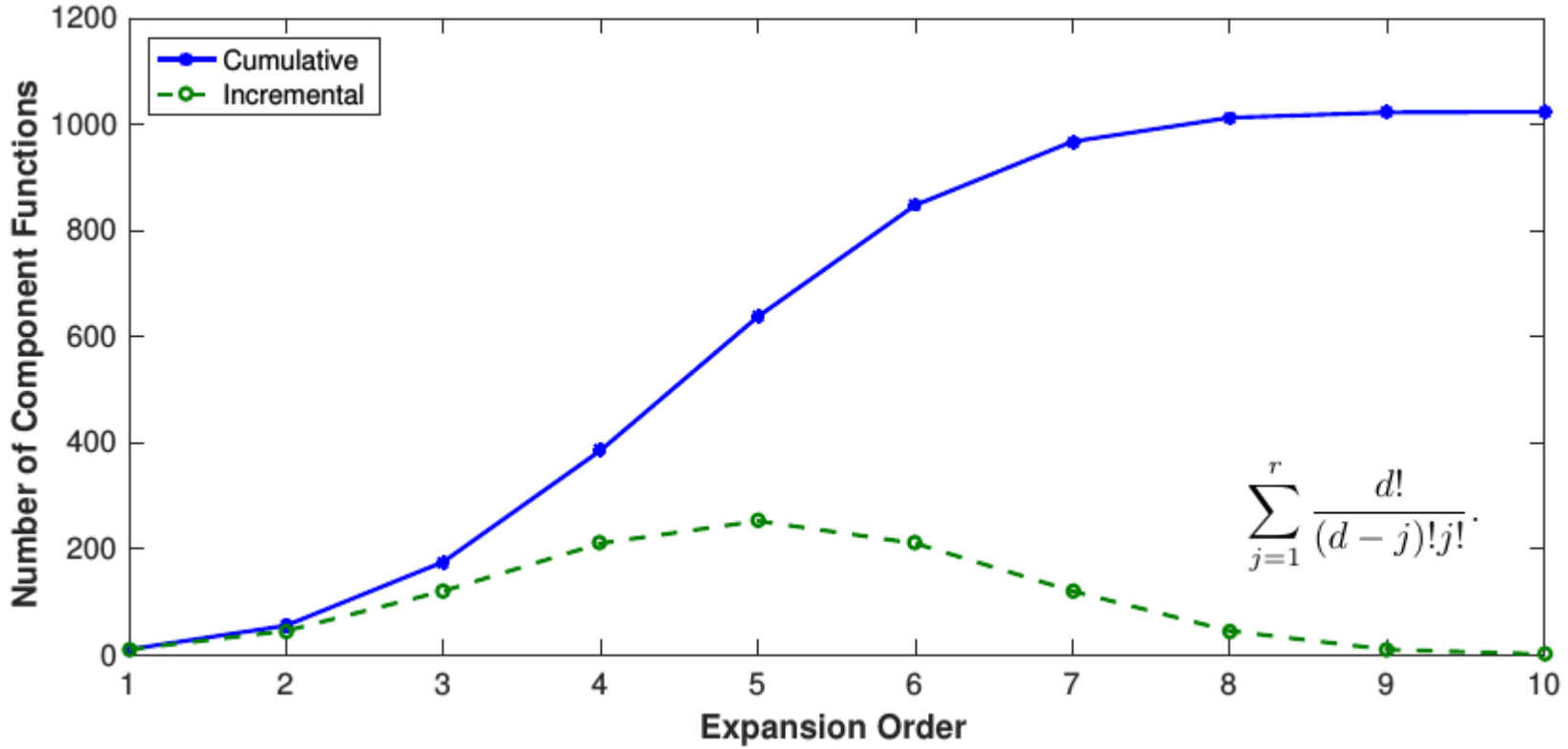


Fig.: Number of component functions for a 10-dimensional, at each given expansion order.

Up to this point we have discussed dimensional decomposition assuming that we will compute all component functions.

→ **Too expensive**

(**500d: 125,251 component functions are needed for a second-order expansion**)

Truncation of the HDMR expansion

-The **active dimensions** refers to the set of indices in **u** corresponding to the inputs that have significance in the input-output response of the system.

-We want to construct an approximated system output by **only considering the active dimensions and ignoring non-active ones**.

-From the figure in the previous slide we can see that the increase in component functions is highest at the start of the expansion.

→ It would be ideal to eliminate any combinations of insignificant inputs early on in the combinatorial tree.

Adaptive dimension selection

The metric for this assessment is based on the work in Ma & Zabaras (2010), however different variates have been implemented (see, e.g. Yang et al (2012)).

Adaptivity measure $\eta_{\mathbf{u}}$ and an appropriately selected cut-off $\epsilon_\eta \geq 0$.
We **reject component functions** containing all indices of u if $\eta_{\mathbf{u}} < \epsilon_\eta$.

$$\eta_{\mathbf{u}} = \frac{\left\| \int f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) d\mathbf{x} \right\|_2}{\left\| \sum_{\mathbf{v} \subset \mathcal{S}, |\mathbf{v}| \leq |\mathbf{u}|-1} \int f_{\mathbf{v}}(\mathbf{x}_{\mathbf{v}}) d\mathbf{x} \right\|_2},$$

e.g. in 1 dimension $\frac{|\mathbb{E}(f_j)|}{|f_0|}$ ...and in 2 dimension $\frac{|\mathbb{E}(f_{j_1,j_2})|}{\sum_{j=0}^{D_1} |\mathbb{E}(f_j)|}$

→ **scalar measure:** indicating the relative importance of the component function with index **u**.

→ The adaptivity coefficient corresponds to the relative integral of the current component function with respect to **approximated system integral of the previous expansion order.**

→ Only the integral of the current component function will be required, as the denominator will have already been computed in the last expansion.

Merging HDMR wirth ASGs

Recall: $$f(\mathbf{x}) = \sum_{\mathbf{u} \subseteq \mathcal{S}} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}), \quad f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) = f(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}\setminus\mathbf{x}_{\mathbf{u}}} - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x}_{\mathbf{v}}),$$

The component function now operations on a **|u|**-dimensional sparse grid and all subsets of the current component functions are also lower dimensional sparse grid interpolations.

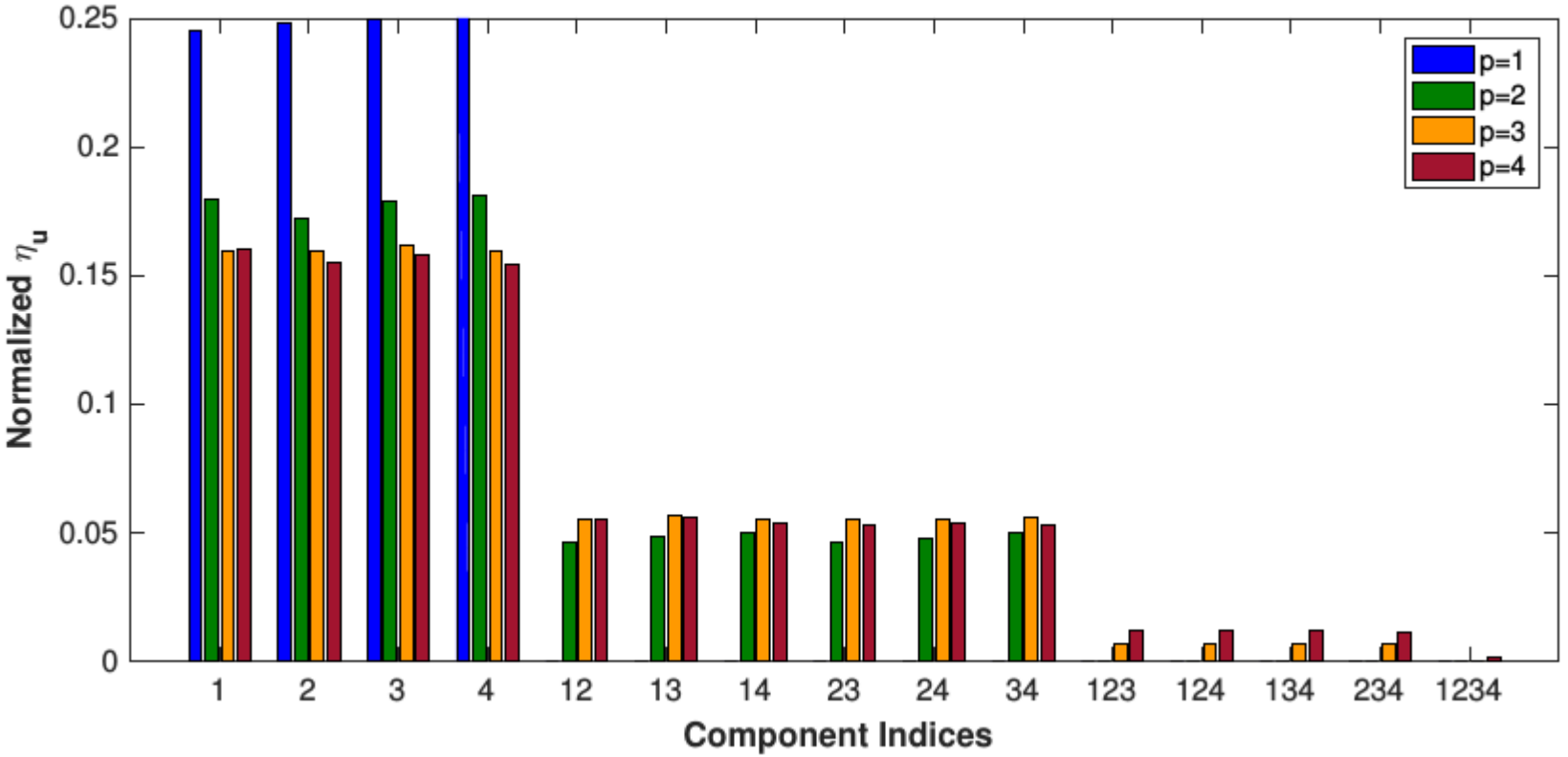
$$\begin{aligned} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) &\approx \mathcal{SG} f(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}\setminus\mathbf{x}_{\mathbf{u}}} - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x}_{\mathbf{v}}) \\ &\approx \sum_{\|\mathbf{k}\|_1 \leq \ell+d-1} \sum_{\mathbf{i} \in \mathbf{I}_{\mathbf{k}}} \alpha_{\mathbf{i},\mathbf{k}} \Phi_{\mathbf{i},\mathbf{k}}(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}} - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x}_{\mathbf{v}}) \end{aligned}$$

One nice feature: $$\mathbb{E}[f(\mathbf{x})] = \sum_{\mathbf{u} \subseteq \mathcal{S}} \mathbb{E}[f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}})]$$

→ Integration trivial on adaptive sparse grids)

NOTE: HDMR independent of choice of “basis” – SG is convenient.

Example I



Active dimension adaptivity measure $\eta_{\mathbf{u}}$ for a 4-dimensional polynomial $(x_1 + \ldots + x_4)^p$

ATTENTION!

In optimal situation where functions that need to be approximated are “somehow” additive separable, truncation can work very efficiently.

→ There are however situations where ASGs perform substantially better than HDMR+ASGs (e.g. “product-peak” functions).

→ Transformations or alternative adaption criteria might help.

Parallelization

All the component functions within a “HDMR level” are independent!
→ an additional layer of parallelism (MPI_Groups) on top of sparse grid parallelism

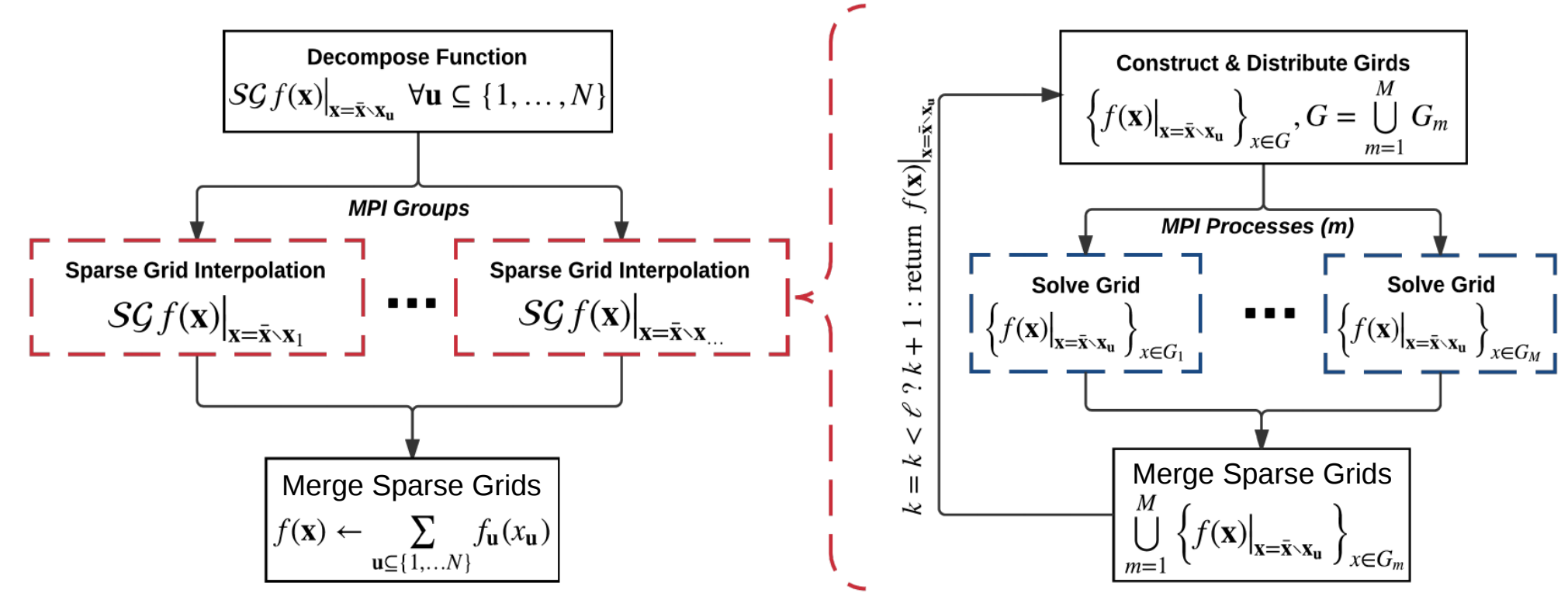


Fig: The red boxes indicate region of isolated (local to the communicator) communication.

HDMR as a whole

HDMR can provide 3rd layer of sparsity (SG – ASG – HDMR).

HDMR can provide significant performance gains:

→ for functions with low input-output correlation.

→ provides high degree of parallelization.

→ can scale efficiently on large scale systems.

→ allows for computability of otherwise uncomputable systems.

→ we were able to solve 400d smooth dynamic stochastic problems.

→ we were able to solve 60d dynamic models with kinks.

Questions