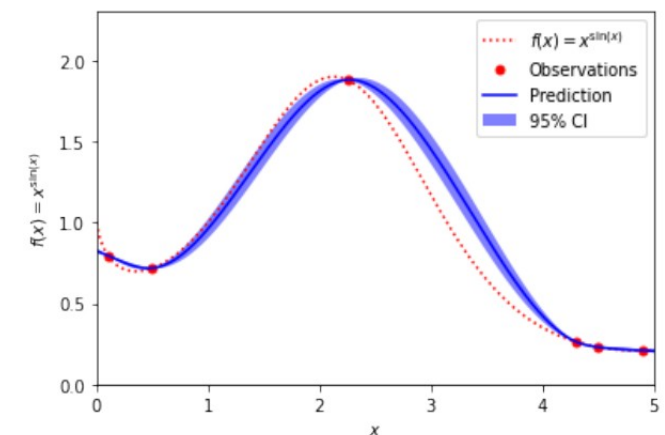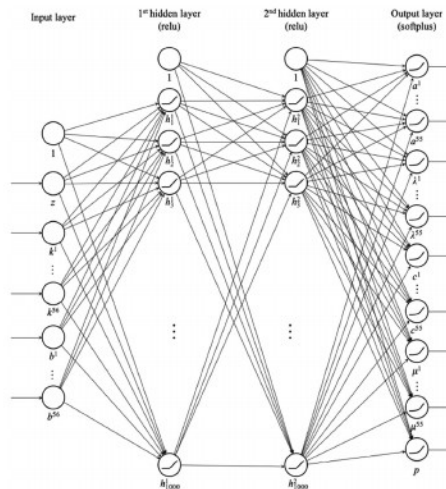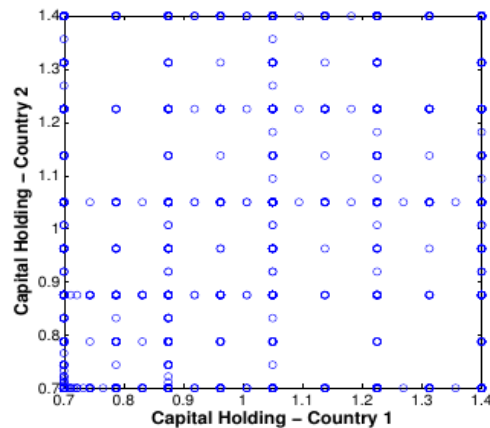# Advanced Methods in Computational Economics

Simon Scheidegger
simon.scheidegger@unil.ch
November 14th- 21st, 2022
CREST

https://github.com/sischei/crest_comp_econ

Who I am?

- Assistant Prof, University of Lausanne, Department of Economics
  (https://sites.google.com/site/simonscheidegger)

- Ph.D., theoretical Physics, University of Basel (2010)

Research interests:
Research focus on developing computational methods for solving and estimating
high-dimensional dynamic stochastic economic models and applying them
to climate economics, macro, and finance.

Teaching:
Machine Learning, Programming, numerical analysis, & computational methods,
software engineering, financial economics.

Who I are you?

- What is your background?

- Why are you here?

- What are your expectations of the mini-course?

What is this mini-course about?

Roadmap – fast forward:

Day 1, Monday – November 14th

1. (Adaptive) Sparse Grids

2. Hands-on Session (I)
    https://github.com/SparseGridsForDynamicEcon/SparseGrids_in_econ_handbook.
    Play with adaptive sparse grid code ($\rightarrow$ analytical functions).

3. Hands-on Session (II)
    https://github.com/SparseGridsForDynamicEcon/HDMR.
    Dynamic Programming & Time Iteration with Sparse Grids.

4. High-dimensional dynamic stochastic model representation (HDMR).

5. Hands-on Session (III)
    Analytical examples.
    Dynamic Programming & Time Iteration with HDMR.

Roadmap – fast forward (2):

Day 2, Thursday – November 17th

1. A brief recap on Machine Learning Basics
    Deep Learning Basics (The multi-layer perceptron, Feed-forward networks
        Network training – SGD, Error back-propagation, Some notes on over-fitting).

2. Throughout the lecture – hands-on:
        Perceptron.
        Gradient descent.
        If time permits: a simple multi-layer perceptron implementation & several examples.
        Basics on Tensorflow & Keras.

3. Deep Surrogate (https://github.com/DeepSurrogate/OptionPricing).

4. Deep Equilibrium Nets (https://github.com/sischei/DeepEquilibriumNets).

Roadmap – fast forward (3):

Day 3, Monday – November 21th (via Zoom)

1. Basics on Gaussian Process Regression (supervised machine learning).
   Noise-free kernels, Kernels with noise.

2. The curse of dimensionality and how to deal with it (e.g., active subspaces).

3. Bayesian Active Learning.

4. Dynamic Programming/optimal control with GPs.
   https://github.com/GaussianProcessesForDynamicEcon/DynamicIncentiveProblems

6. An outlook to frontier topics of GPs (Limitations of GPs and "big data"/scalable GPs

Some useful materials

Deep Learning
Ian Goodfellow and Yoshua Bengio and Aaron Courville
MIT Press 2016
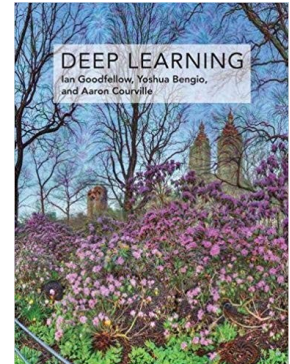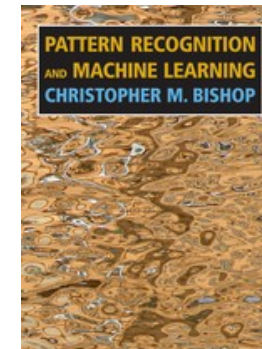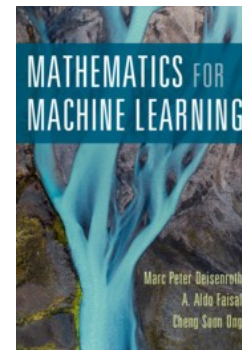http://www.deeplearningbook.org/

Pattern Recognition and Machine Learning
C. M Bishop, Springer 2006
(pdf freely available)

Mathematics for Machine Learning
Deisenroth, A. Aldo Faisal, and Cheng Soon Ong.
Cambridge University Press 2020

→ **There is a great community out there (use your browser and Google around...)**

Lecture Slides & Codes on Git

I will post the slides and codes for this course here:

https://github.com/sischei/crest_comp_econ

We will use the Nuvolos Cloud services

https://nuvolos.cloud/   **nuvolos**

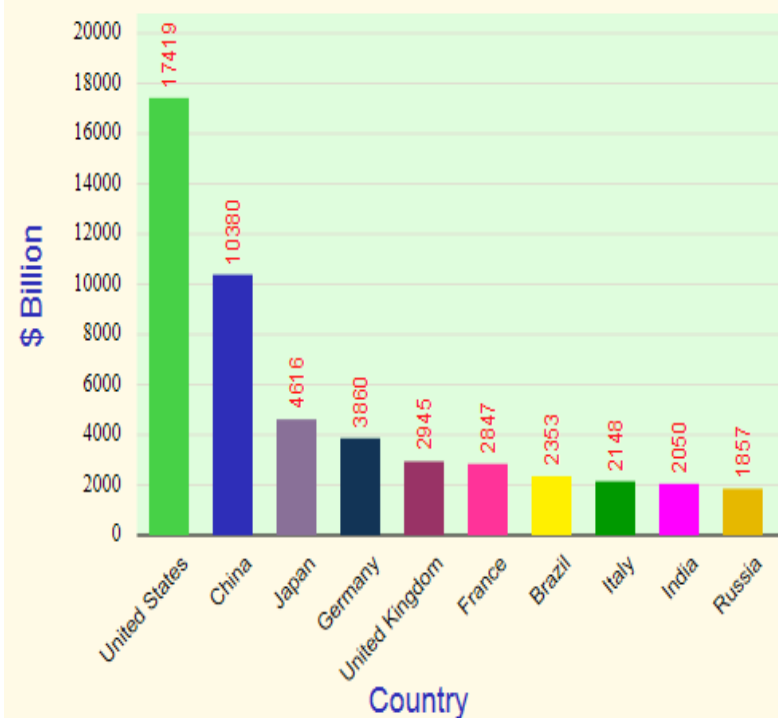→ please enroll the class by clicking on this link:

1. (Adaptive) Sparse Grids

I.   Motivation – "the curse of dimensionality"

II. From Full (Cartesian) Grids to Sparse Grids

III. Adaptive Sparse Grids

IV. How to integrate ASGs in dynamic economic model

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 – Heterogeneity in OLG* models
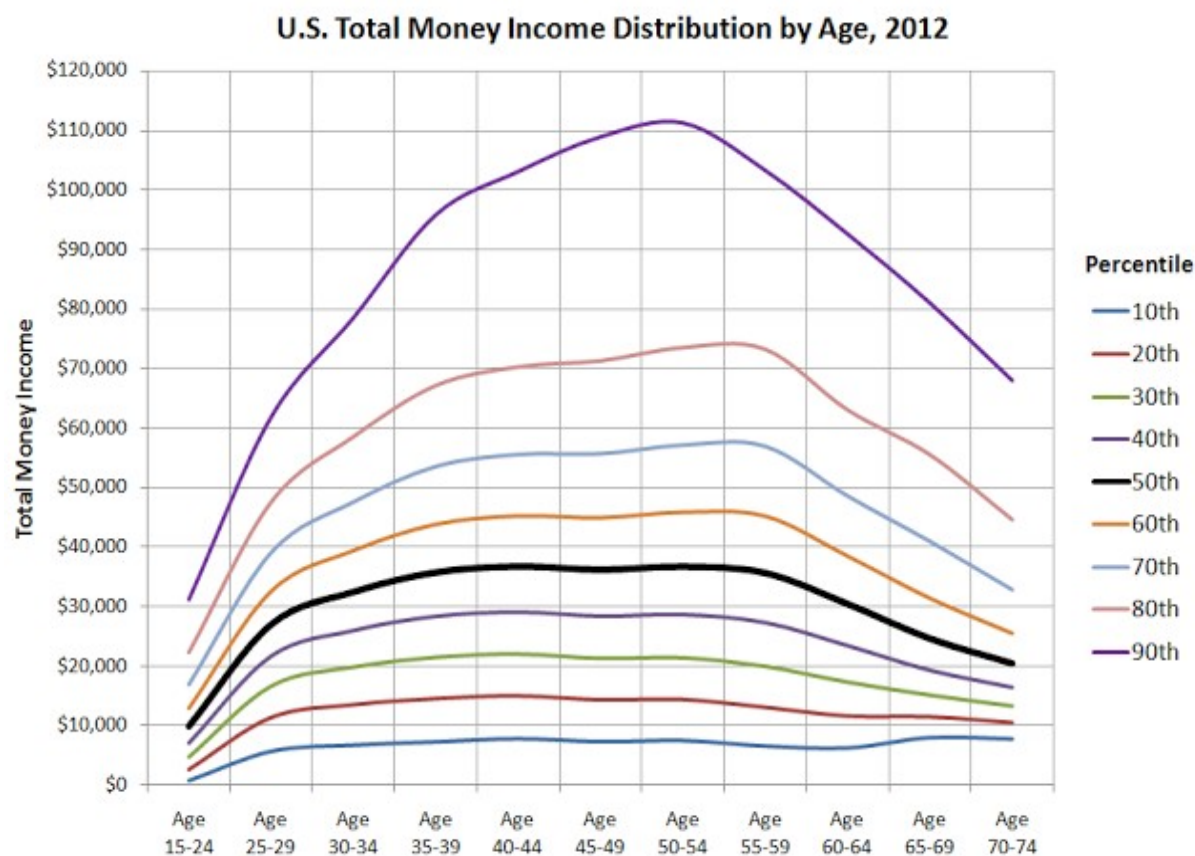
*Overlapping generation models


To model e.g. social security:

- How many age groups?

- borrowing constraints?

- aggregate shocks?

- …

→ **Model: heterogeneous & high-dimensional**

U.S. Total Money Income Distribution by Age, 2012

Financial markets: non-Gaussian returns

- Derivative contracts giving a <u>right to buy or sell an underlying security</u>.
  - *European* if exercise at expiration only.
  - **American** if exercise any time until expiration.

- American options are <u>extremely challenging</u>:
  - → Dynamic optimization problem.

- Basic models do not describe dynamics accurately (e.g., Hull (2011)).

- Financial returns are often not Gaussian.

- Realistic models are hard to deal with, as they need many factors.
  - → Curse of dimensionality.
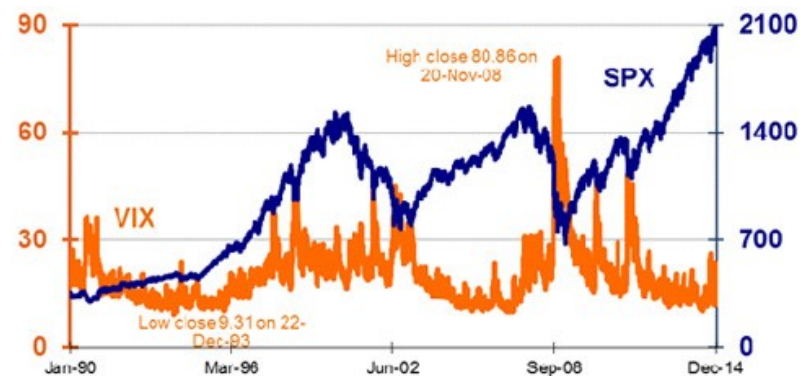
Dynamic Programming/Value Function Iteration

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

Dynamic programming seeks a time-invariant policy function **p** mapping a state $\mathbf{x_t}$ into the control $\mathbf{u_t}$ such that for all $t \in \mathbb{N}$ $u_t = p(x_t)$
The solution is approached in the limit as $j \to \infty$ by iterations on:

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

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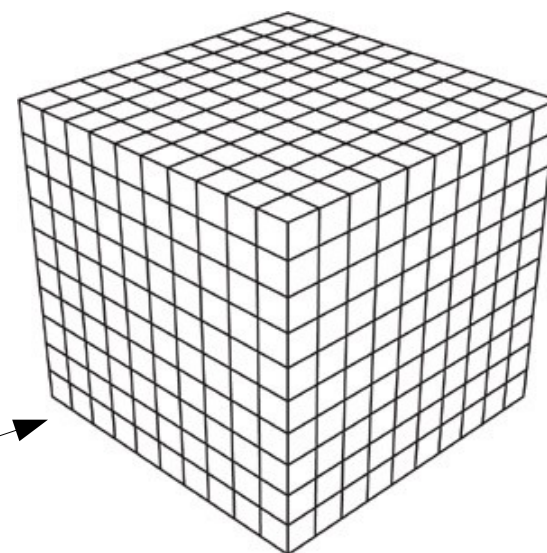
$$\tilde{x} = g(x,u)$$

x: grid point, describes your system.
State-space potentially **high-dimensional**.

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

→ $\mathbf{N^d}$ points in ordinary discretization schemes.

→ Use-case for (adaptive) sparse grids.

How many is dimensions is high dimensions?

?

CREST

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?

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
Exploit symmetries,...

Deal with #Points
Adaptive Sparse Grids

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
Exploit symmetries,...

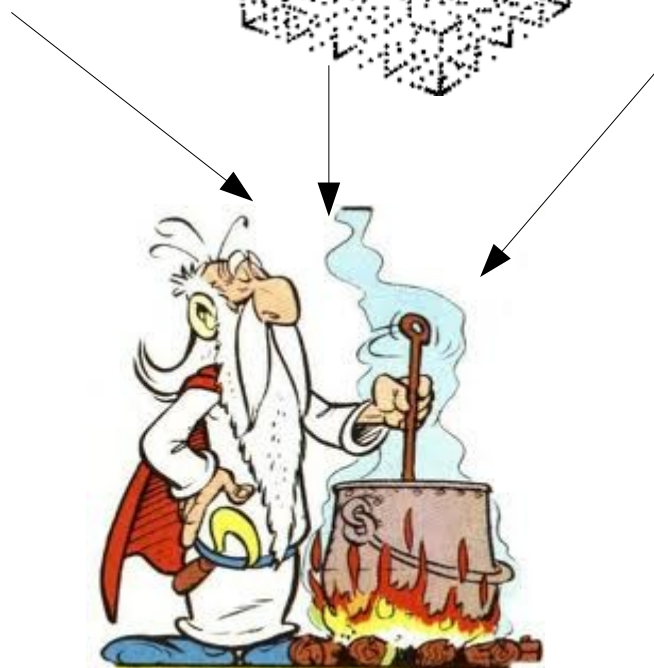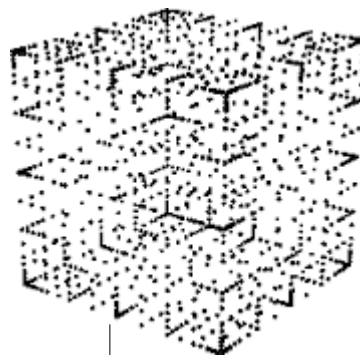**Deal with #Points**
Adaptive Sparse Grids

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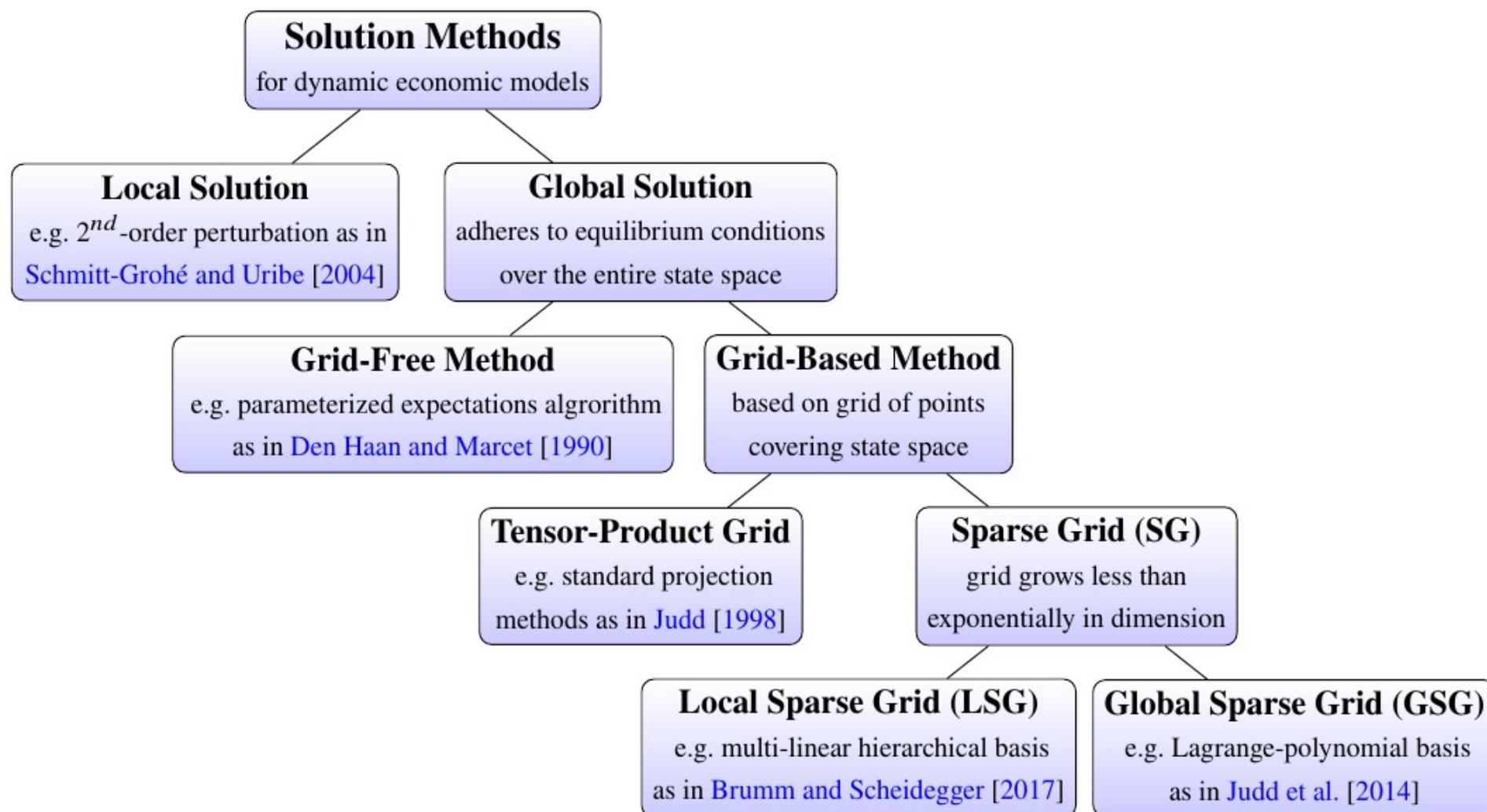
Computational modelling

Numerical Solution Methods

Solution Methods
for dynamic economic models

Local Solution
e.g. 2nd-order perturbation as in
Schmitt-Grohé and Uribe [2004]

Global Solution
adheres to equilibrium conditions
over the entire state space

Grid-Free Method
e.g. parameterized expectations algrorithm
as in Den Haan and Marcet [1990]

Grid-Based Method
based on grid of points
covering state space

Tensor-Product Grid
e.g. standard projection
methods as in Judd [1998]

Sparse Grid (SG)
grid grows less than
exponentially in dimension

Local Sparse Grid (LSG)
e.g. multi-linear hierarchical basis
as in Brumm and Scheidegger [2017]

Global Sparse Grid (GSG)
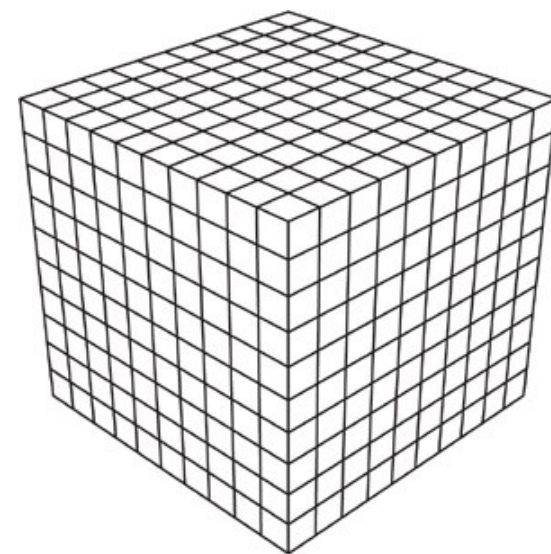e.g. Lagrange-polynomial basis
as in Judd et al. [2014]

Abstract Problem Formulation

i) Dynamic models: heterogeneous & high-dimensional

ii) Want to solve dynamic stochastic models with high-dimensional state spaces

→   Have to approximate and interpolate
    high-dimensional functions

Problem: curse of dimensionality
→ N^d points in ordinary discretization schemes

iii) **Want to overcome curse of dimensionality**

iv) **Want locality & adaptivity of interpolation scheme**

v) **Speed-up*** → access hybrid HPC systems (MPI, OpenMP, TBB, GPU)

Models where high-dim. state spaces show up

e.g. Stokey, Lucas & Prescott (1989), Ljundquist & Sargent (2004), Krüger & Kübler (2004), Judd et. al. (2013), Brumm & Scheidegger (2017),...

- International Real Business Cycle (IRBC) Models:
  Exchange Rates, Global Trade Imbalances

- Dynamic Stochastic General Equilibrium (DSGE) Models:
  Monetary Policy, Business Cycle Fluctuations

- Overlapping Generations (OLG) Models:
  Demographic Change, Social Security

- Mathematical Finance:
  Option pricing,...

II. From Full Grids to Sparse Grids

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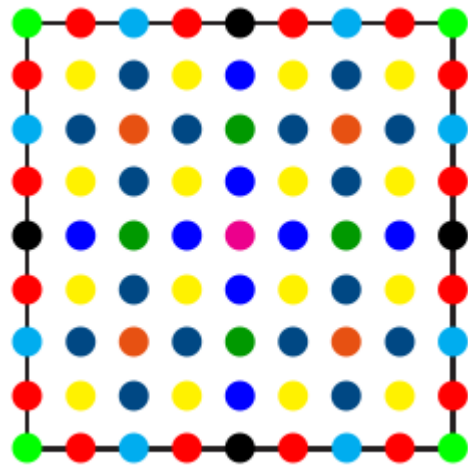
Cartesian Grid
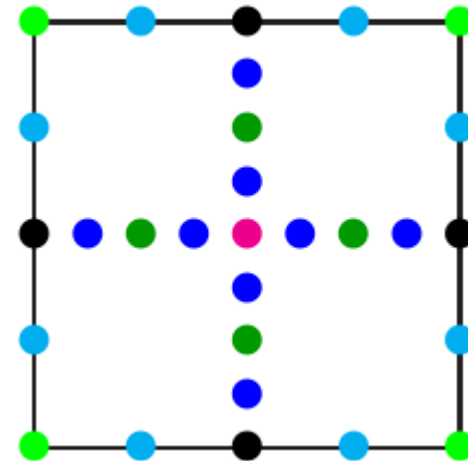
Sparse Grid

Interpolation on a Full Grid

-Consider a **1-dimensional function** $f : \Omega \rightarrow \mathbb{R}$ **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:

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

with the **index set**

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

The corresponding function space:

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

The **1d-interpolant:**

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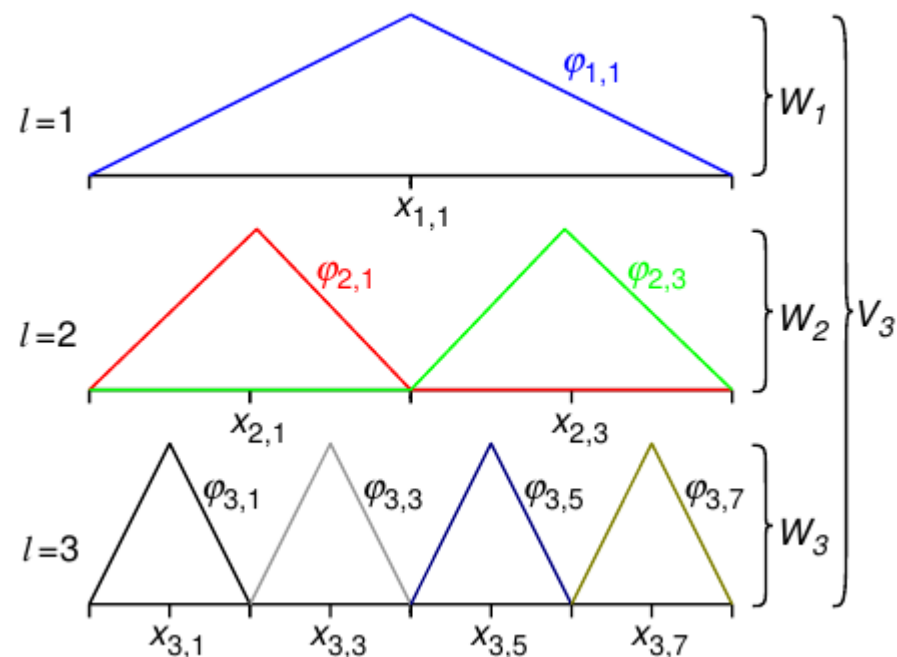

Fig.: 1-d basis functions $\phi_{l,i}$ and the corresponding **grid points** up level *l* = 3 in the hierarchical basis.
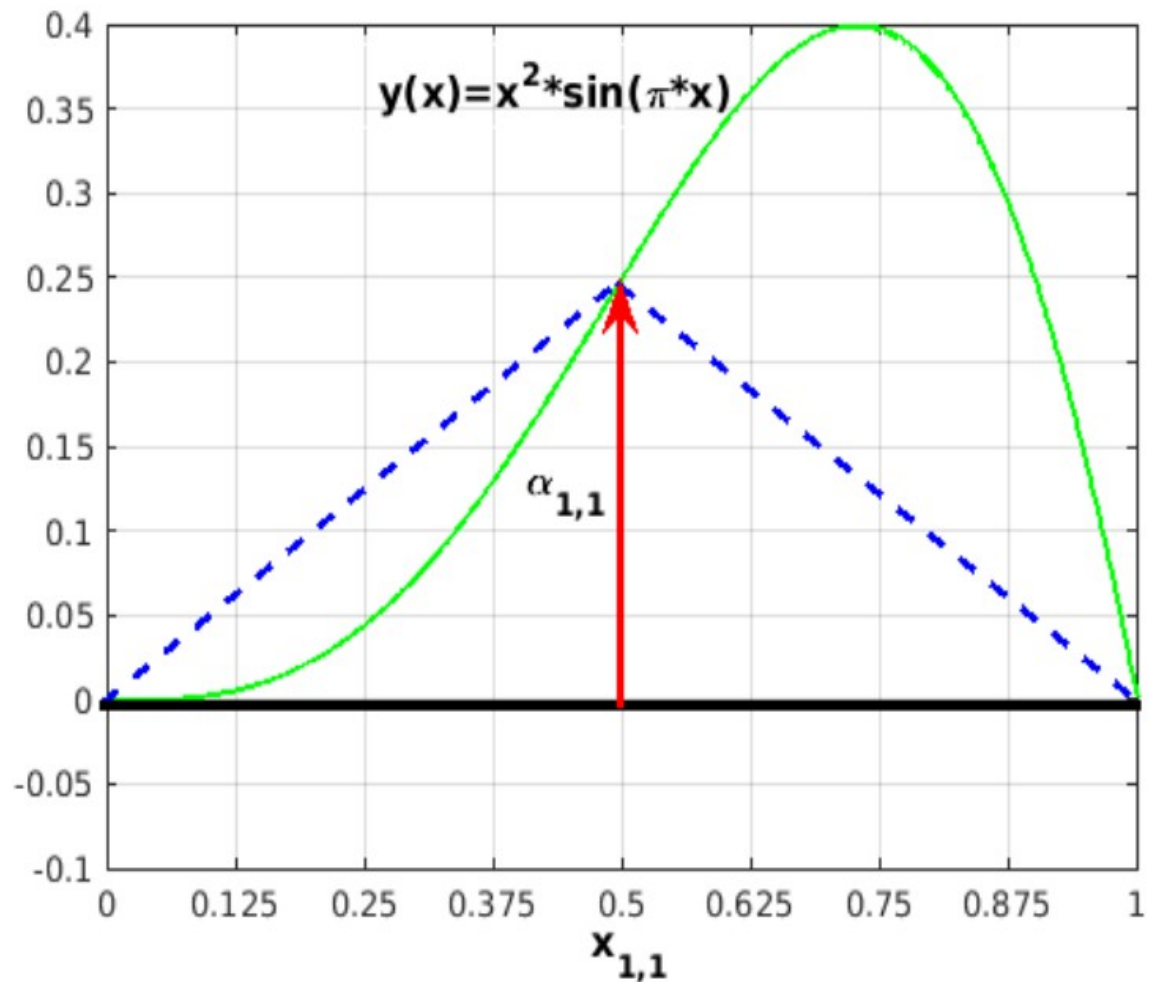
Note: supports of all basis functions of W_k mutually disjoint!

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.

$y(x)=x^2*\sin(\pi*x)$

$\alpha_{1,1}$

$x_{1,1}$

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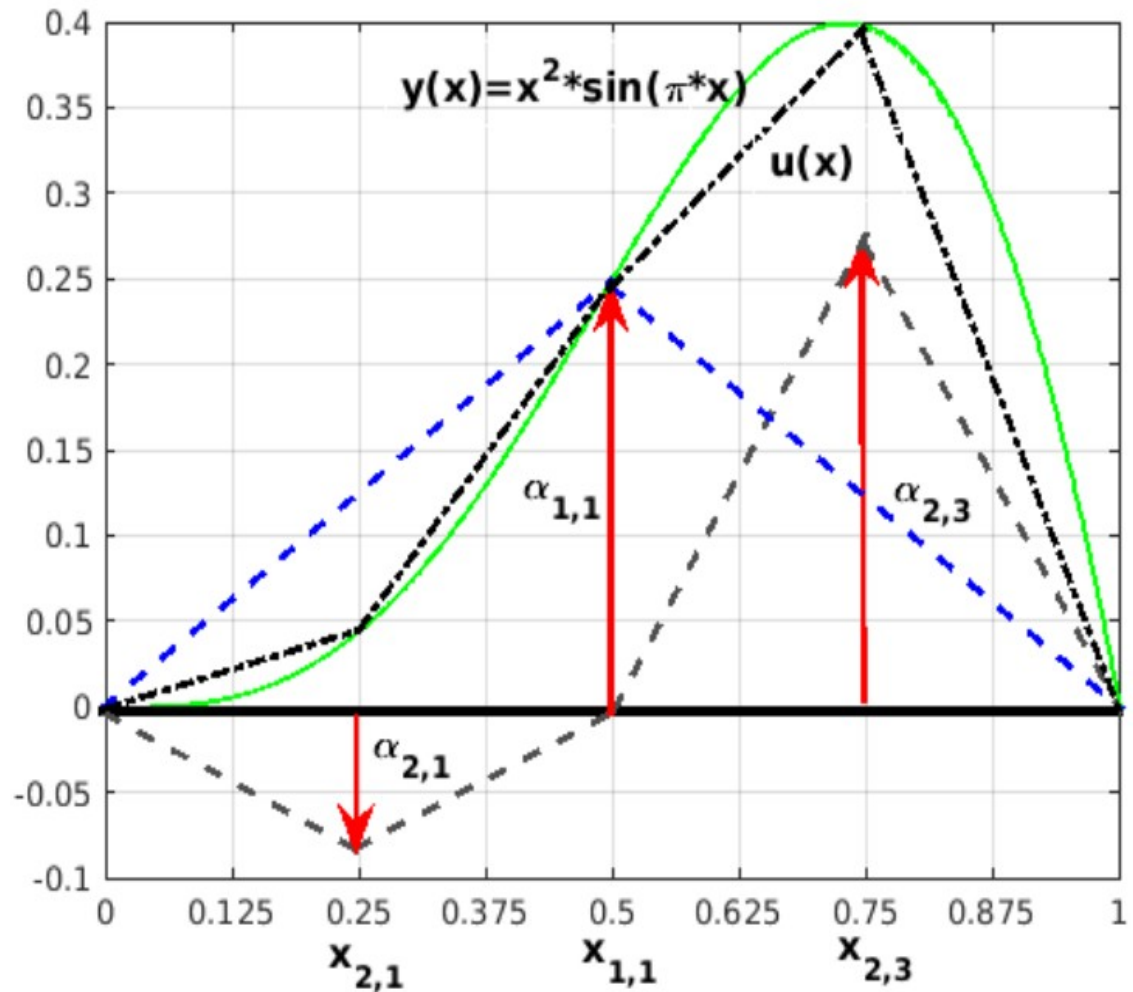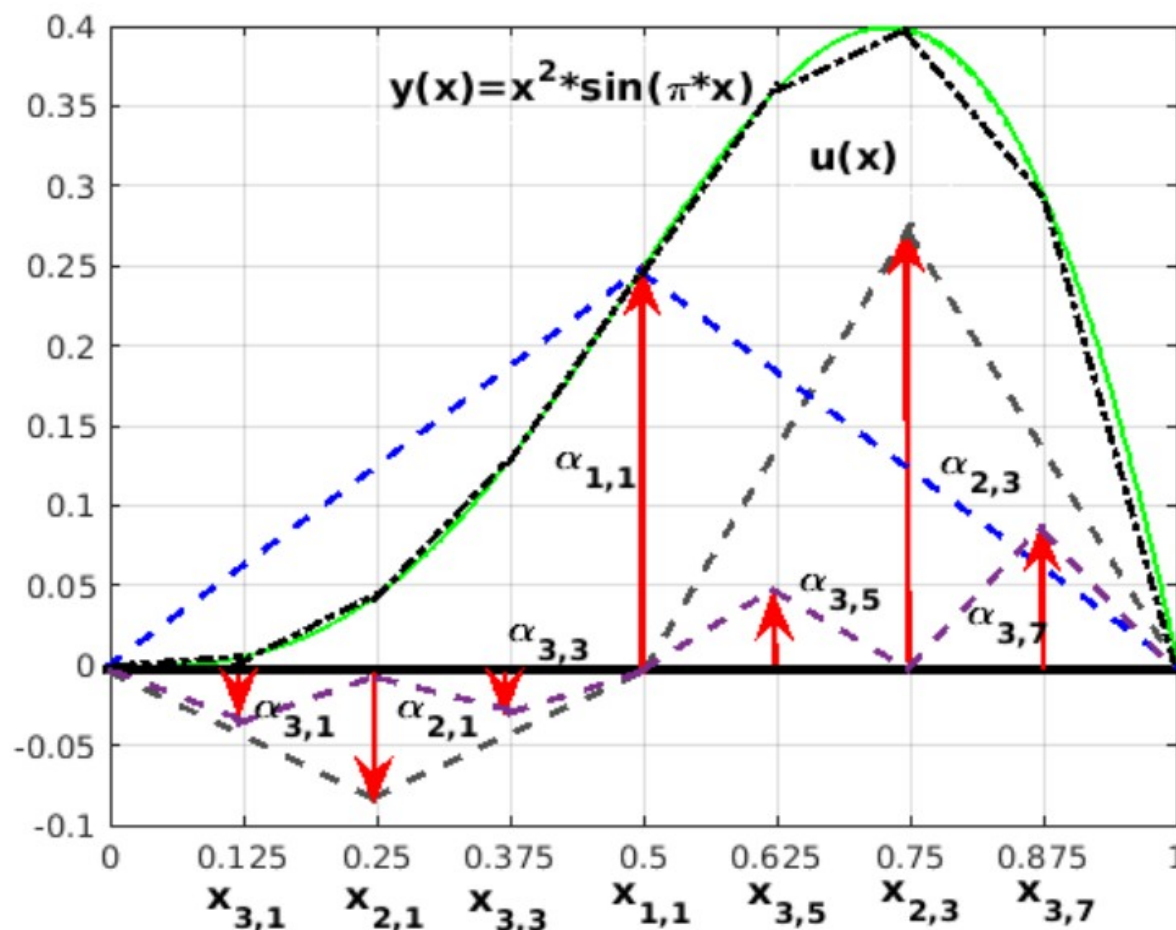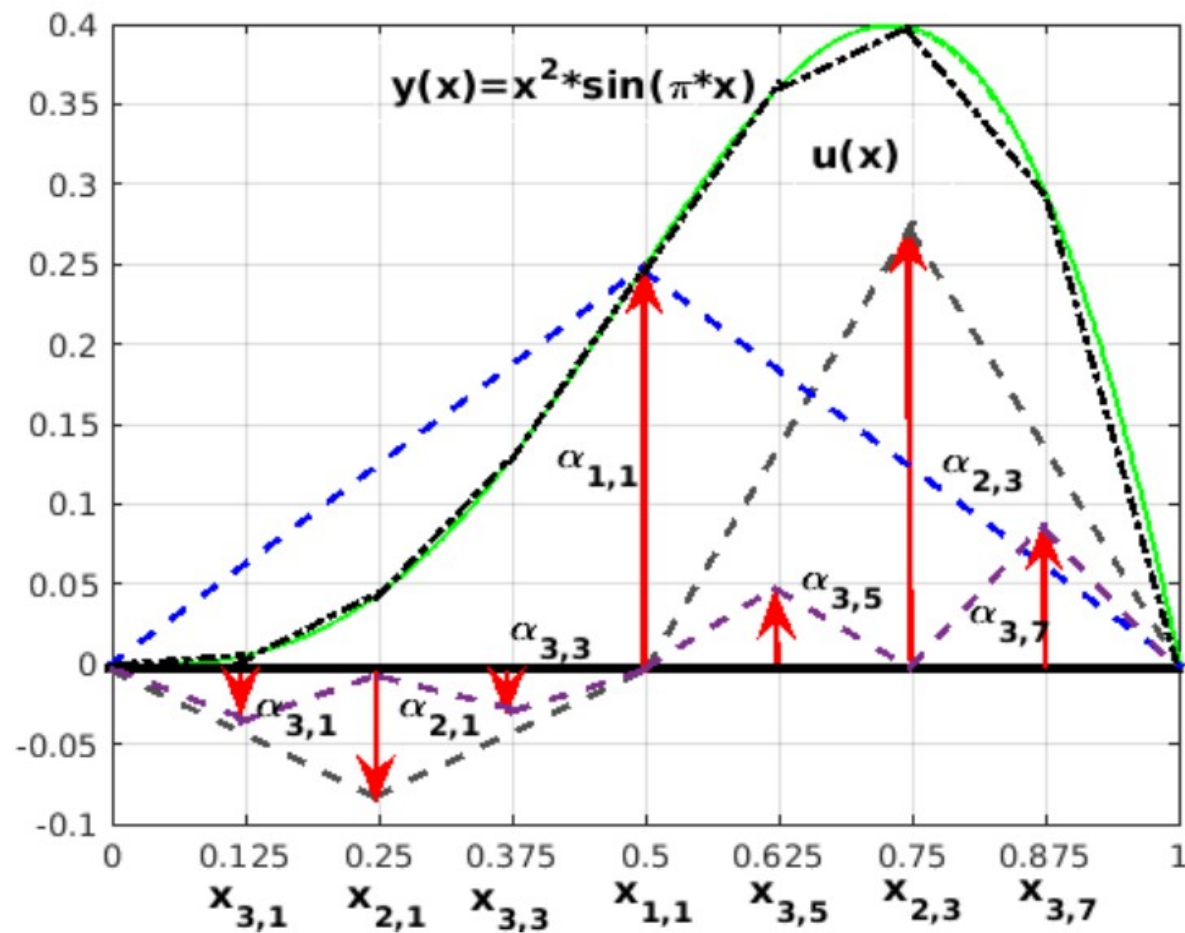**Evaluate function only at points that are unique to the new level**.

MOVIE

Non-zero Boundary Conditions

Want to be able to handle non-zero boundaries:

$$f|_{\partial\Omega} \neq 0$$

If we add naively points at boundaries, **3^d support nodes** will be added.

Numerically cheapest way: **Modify basis functions and interpolate towards boundary.** Various choices possible!
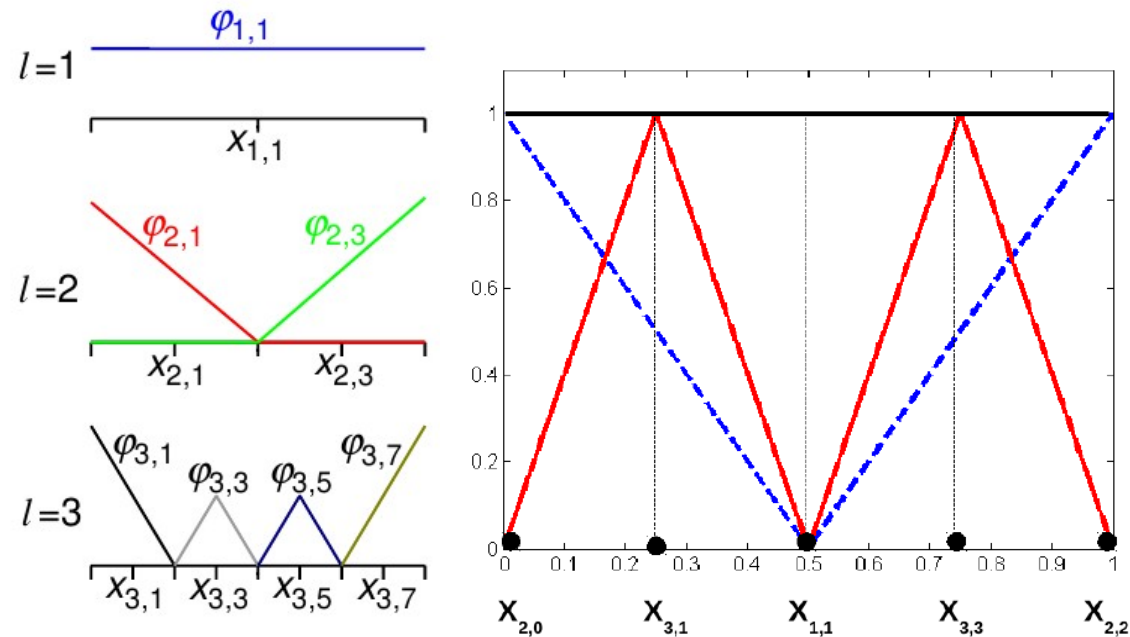
Fig.: Example of modified 1d-basis functions According to Pflüger (2010), which are extrapolating towards the boundary (**left**). They are constant on level 1 and **"folded-up"** if adjacent to the boundary on all other levels. **Right**: "Modified" hat basis.
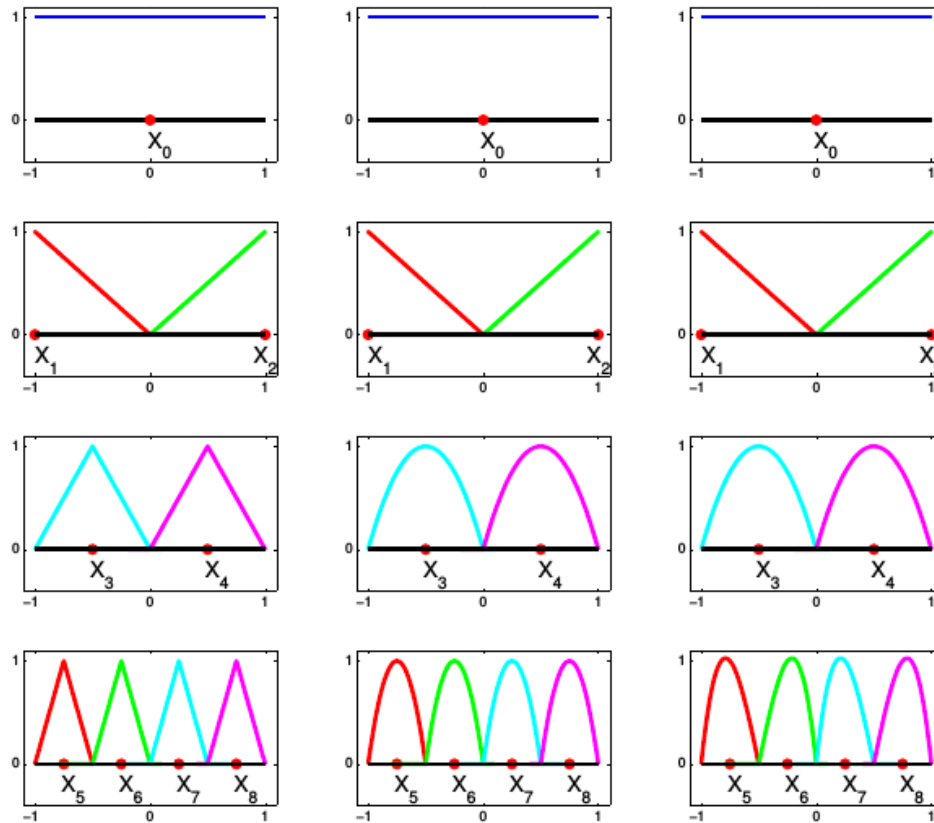
Examples for basis functions

Figure 1: Local polynomial points (*rule_localp*) and functions, left to right: linear, quadratic, and cubic functions.

Fig. taken from Tasmanian User Manual

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:
$$W_{\vec{l}} := \operatorname{span}\{\phi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\}$$

Multi-d interpolant:

$$f(\vec{x}) \approx u(\vec{x}) = \sum_{|l|_\infty \le n} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x})$$
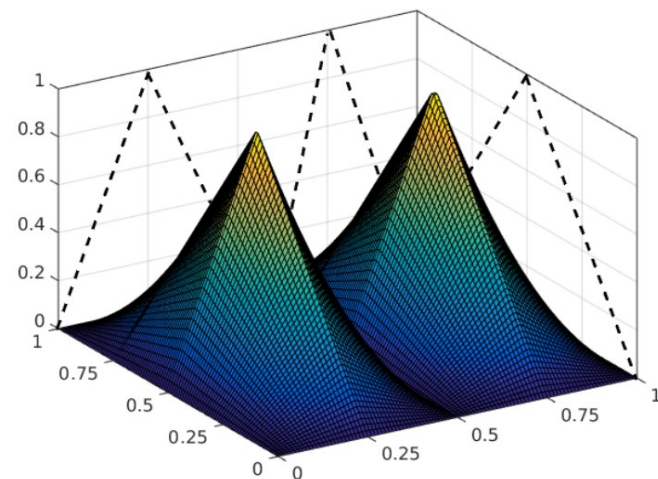
Fig.: Basis functions of the **subspace W$_{2,1}$**

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

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
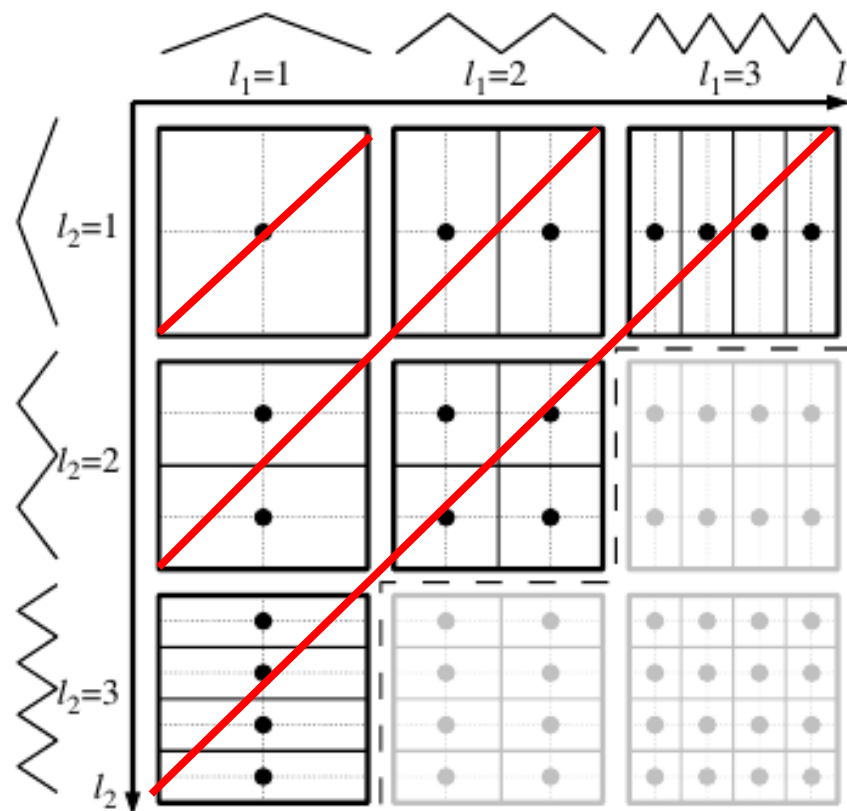
$$V_{0,n}^S := \bigoplus_{|\vec{l}|_1 \le n+d-1} W_{\vec{l}}$$
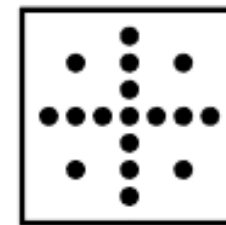
Sparse grid: 17 pt.
Full grid   : 49 pt.

Fig.: Two-dimensional subspaces **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.

Basis Functions of $W_{2,2}$ — Included in V_3
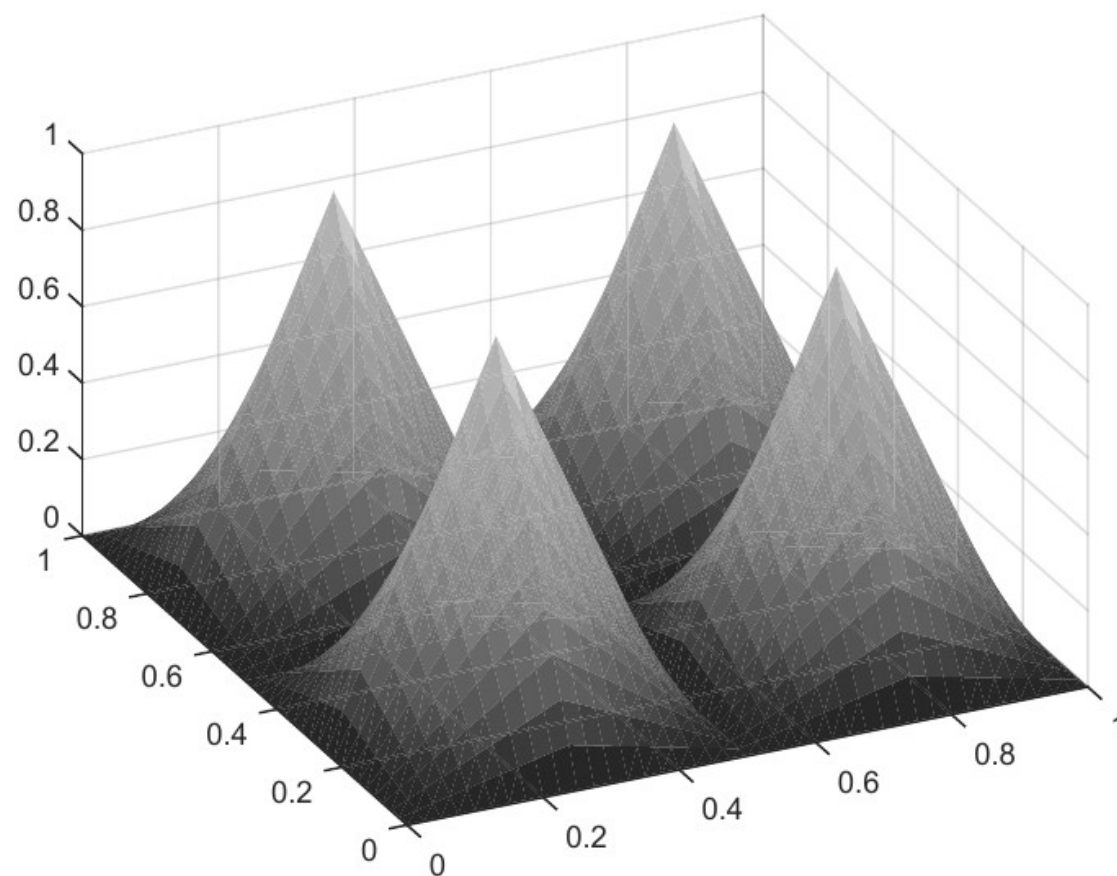
Basis Functions of $W_{3,3}$ — not Included in V_3
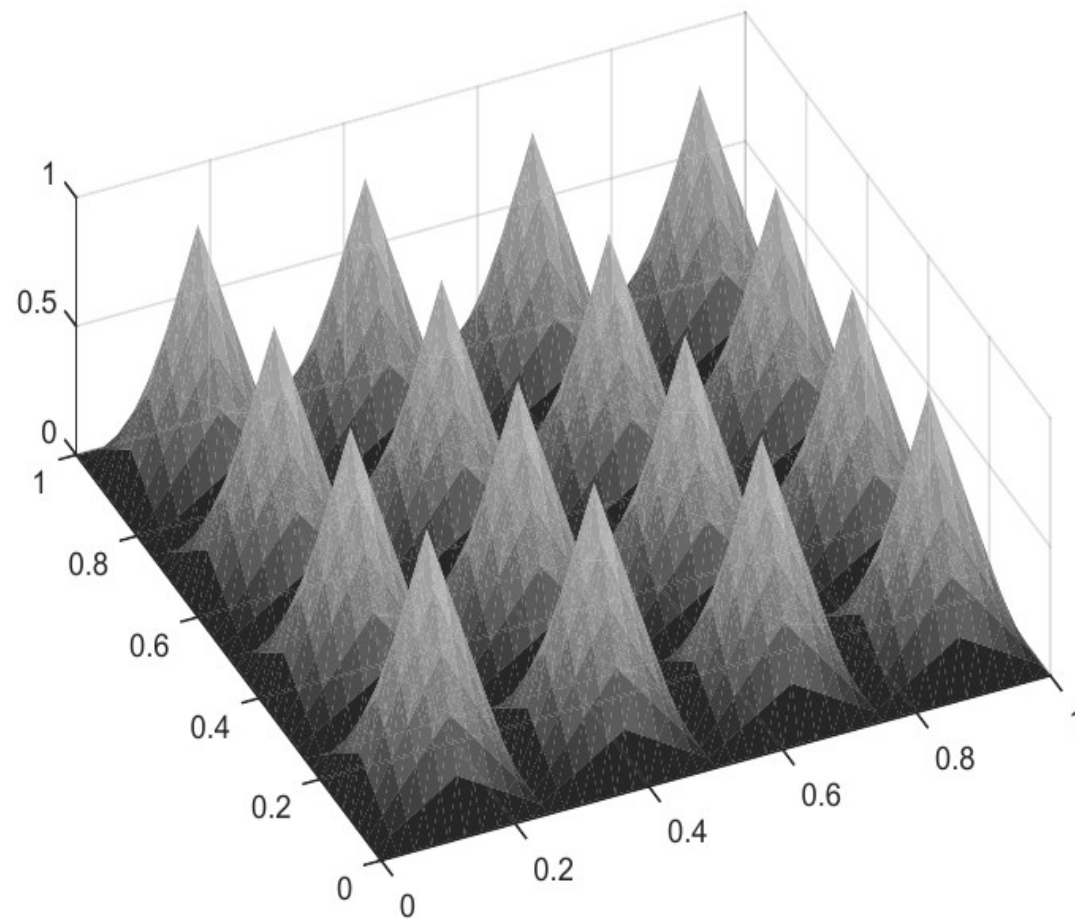
Sparse Grid of Increasing level
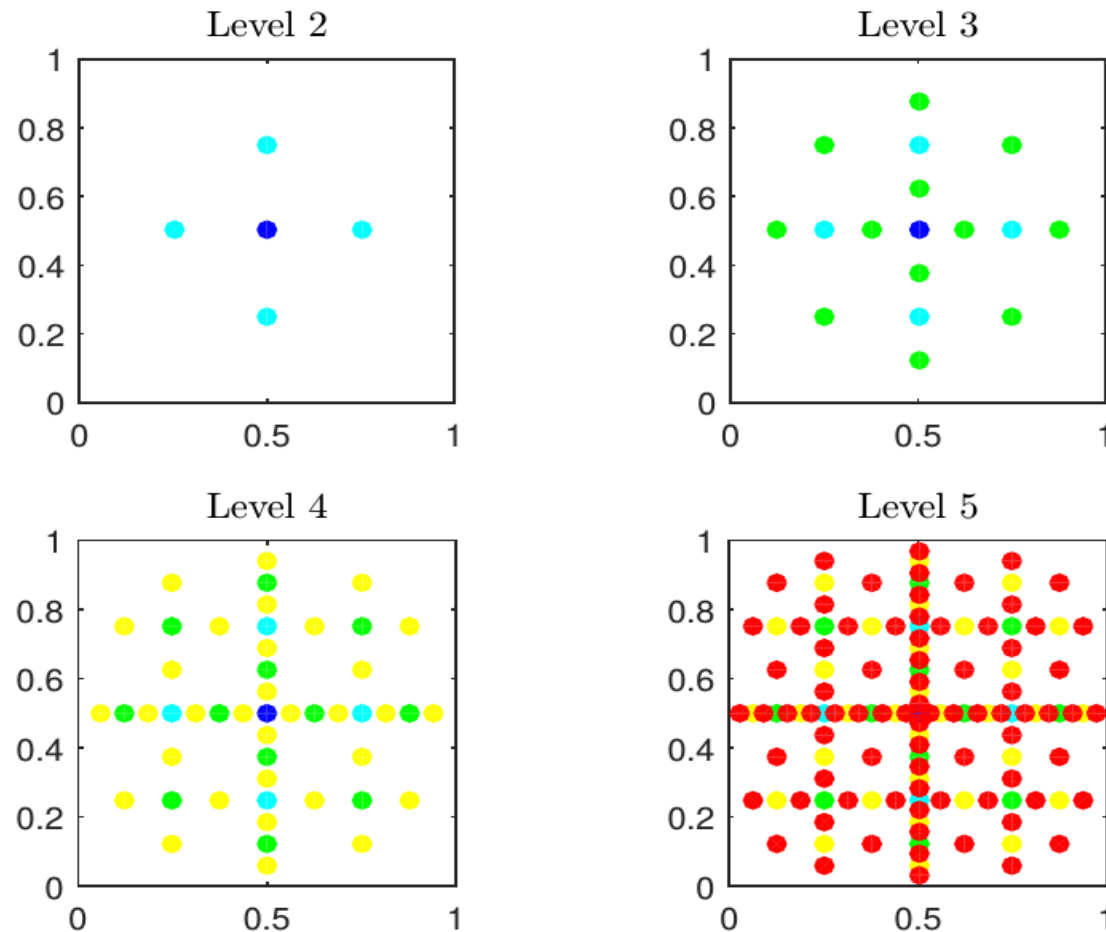
Grid Points

d	$|V_n|$	$|V_{0,n}^S|$
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
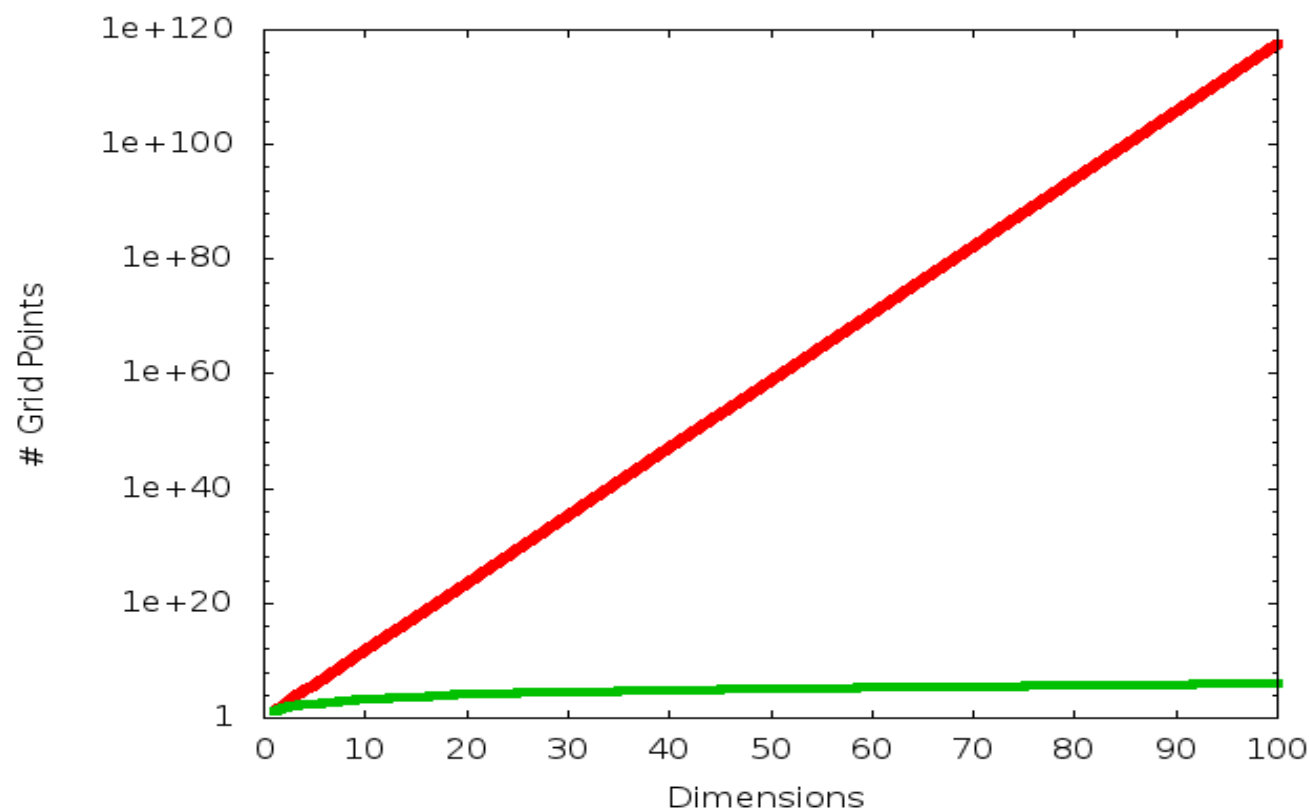
Tab.: Number of grid points for several types of sparse grids of level $n = 4$.
Middle: Full grid; **right: classical sparse grid** with **no points at the boundaries.**
Fig.: Number of grid points growing with dimension (full grid vs. sparse grid).

Sparse Grid with non-zero boundaries

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

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

Sparse grid: 29 pt.
Full grid : 81 pt.
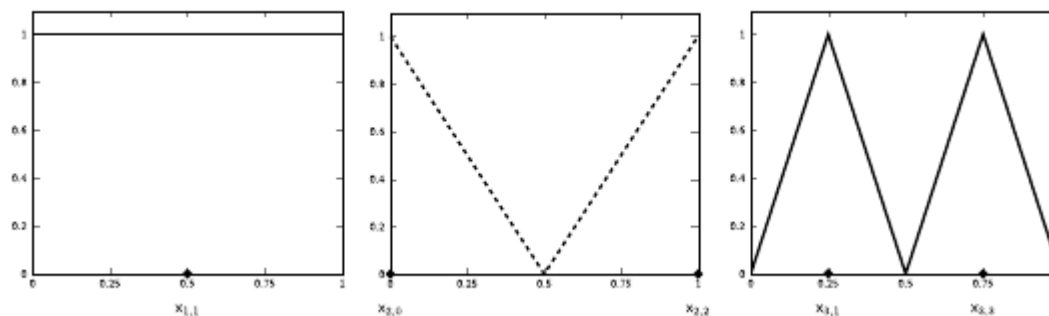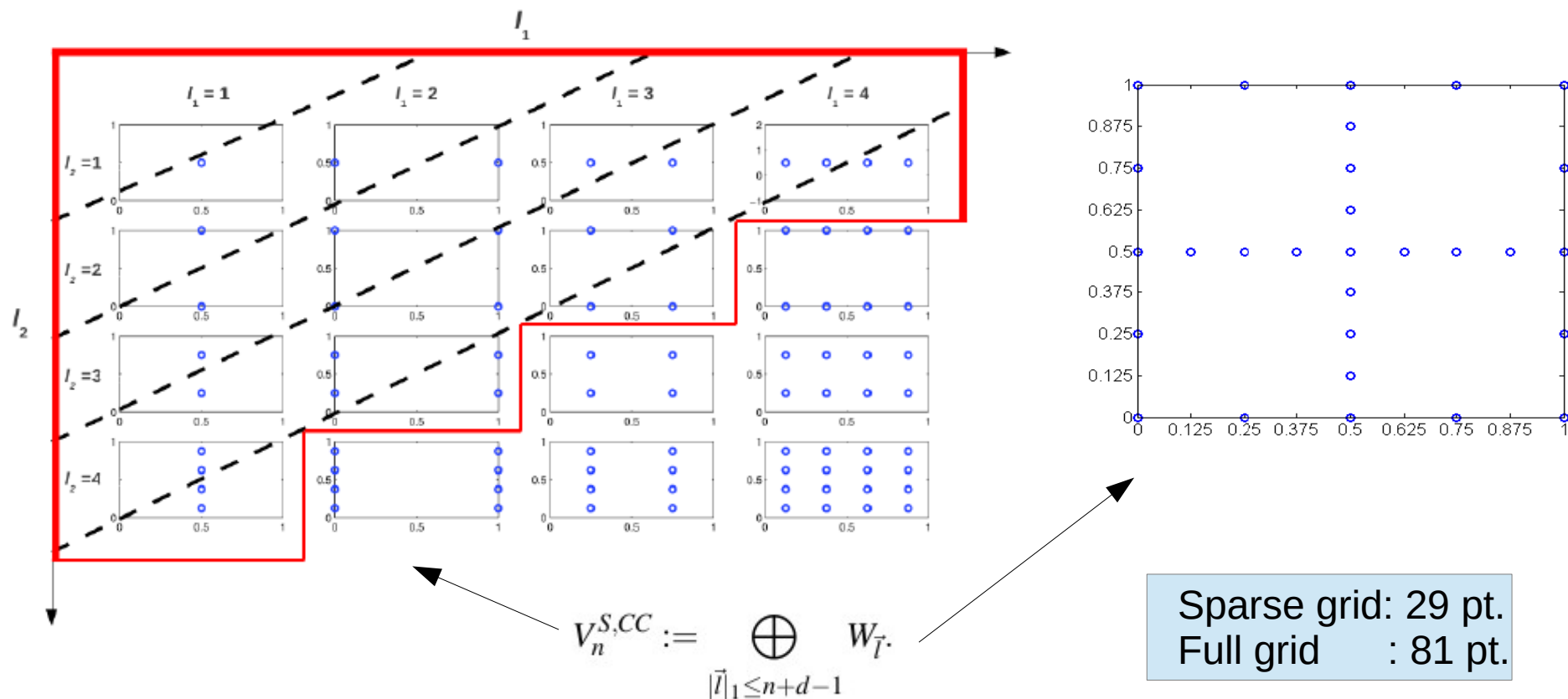
Hierarchical Integration

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

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

Where are Sparse Grids used?

For a review, see, e.g. Bungartz & Griebel (2004)

Sparse grid methods date back to Smolyak(1963)

BUT: Smolyak used global polynomials!

So far, methods applied to:

-High-dimensional integration

e.g. Gerstner & Griebel (1998), Bungartz et al. (2003),...

-Interpolation

e.g. Barthelmann et al. (2000), Klimke & Wohlmut (2005),...

-Solution of PDEs

e.g. Zenger (1991), Griebel (1998),...

More fields of application: regressions, data mining, likelihood estimations, option pricing, data compression, dynamic economic models...

e.g. Kubler & Kruger (2004), Winschel & Kraetzig (2010), Judd et al. (2013) → Smolyak; global basis functions.

III. Adaptive Sparse Grids
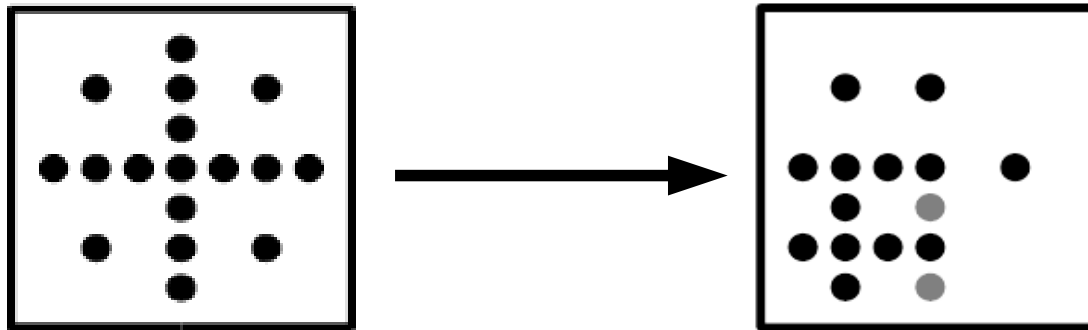
Sketch of adaptive refinement

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.** $|\alpha_{\vec{l},\vec{i}}| \geq \epsilon$

L=1   0.5

L=2   0.25   0.75

L=3   0.125   0.375   0.625   0.875
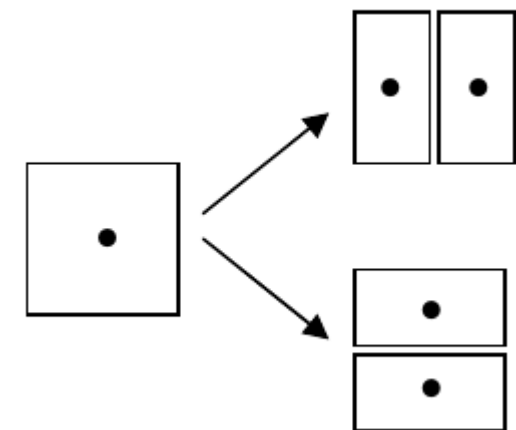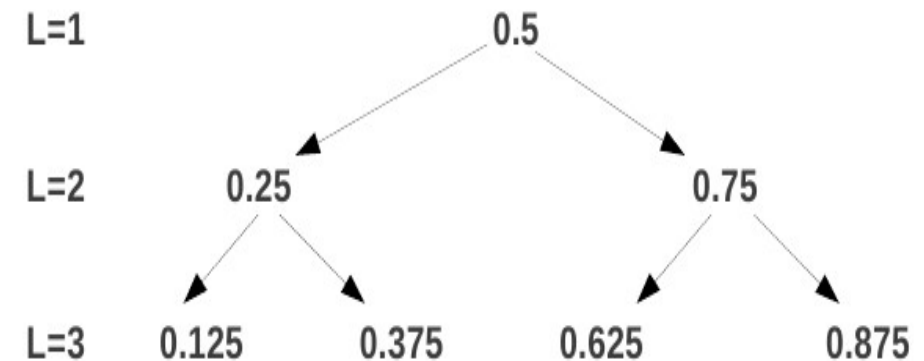
top panel: tree-like structure of sparse grid.
lower panel: locally refined sparse grid in 2D.

Example I
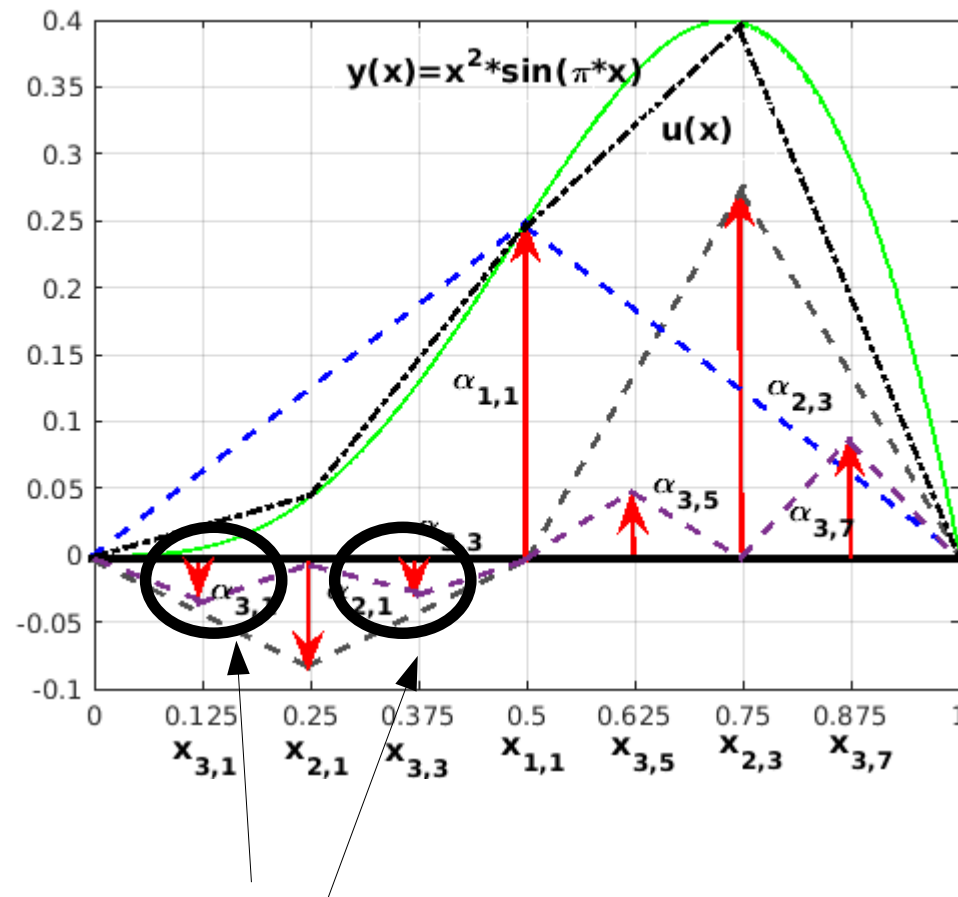
Small – below threshold

Example II
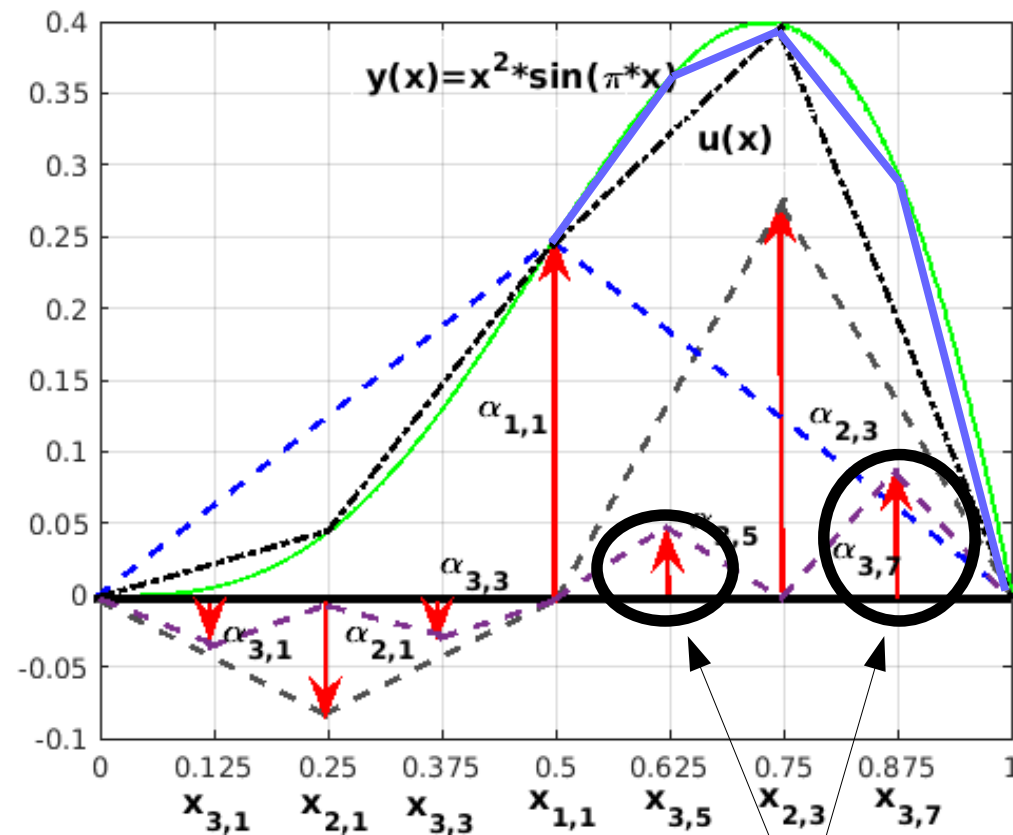
Add points – above threshold

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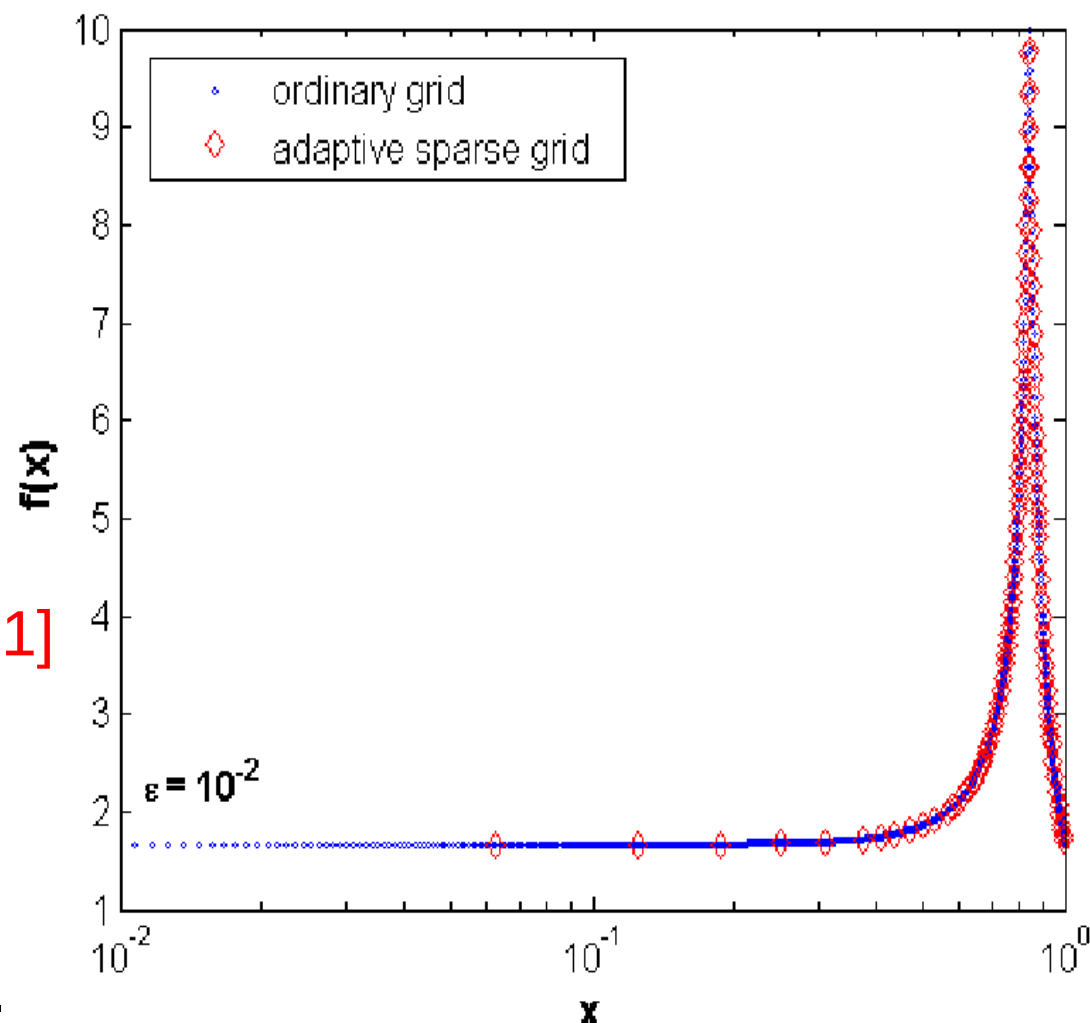
Fig.: Blue: Full grid; red: adaptive sparse grid.

Test in 2d

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

Error: *O(10⁻²)*

Full grid:
→ *O(10⁹)* points

Sparse grid:
→ **311,297 points**

Adaptive sparse grid:
→ **4,411 points**
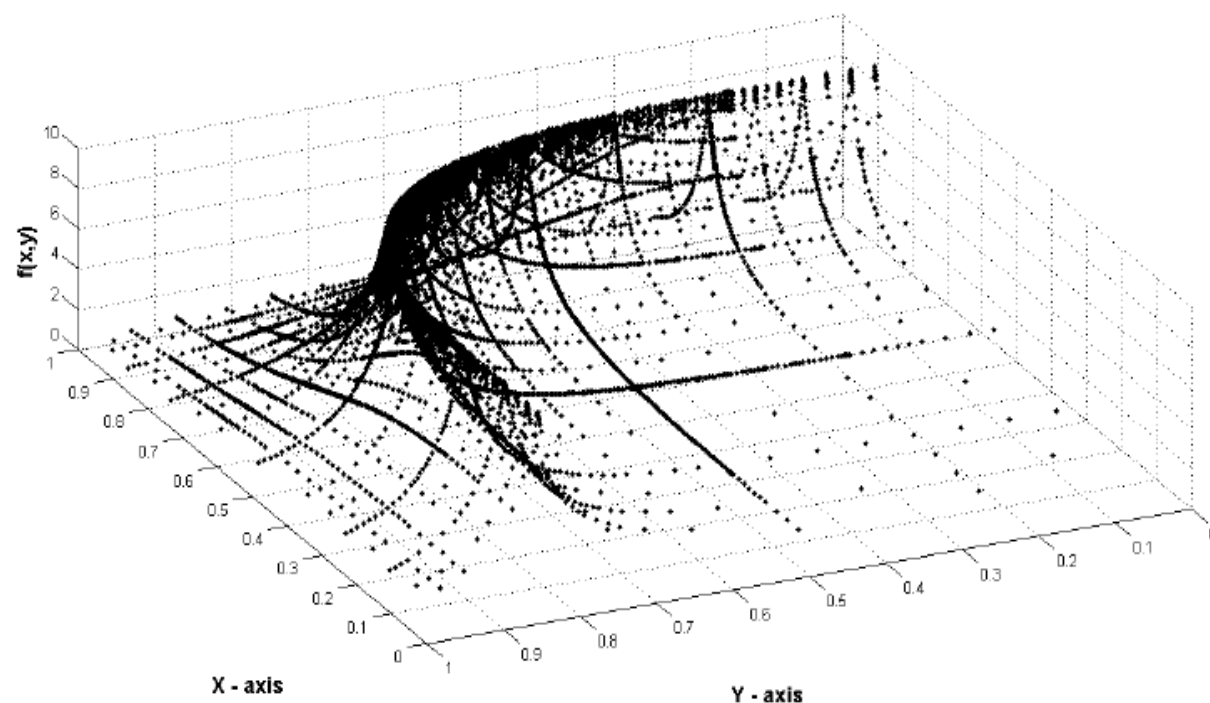
Fig.: 2d test function and its corresponding grid points after 15 refinement steps.

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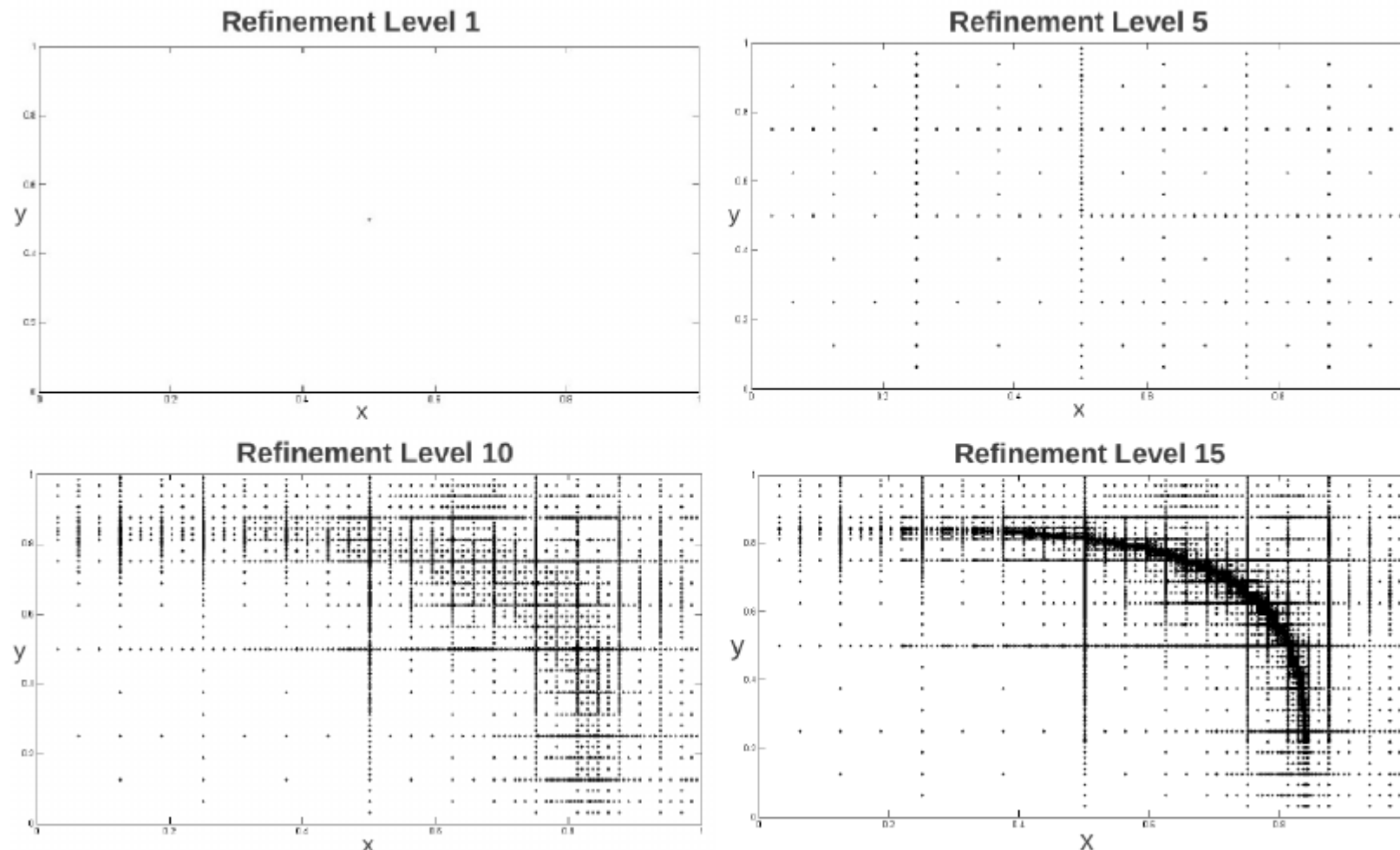
Fig.: Evolution of the adaptive sparse grid with a **threshold for refinement of 10^{-2}** . The refinement levels displayed are L = 1, 5, 10, 15.

Convergence

4,411 points

311,297 points
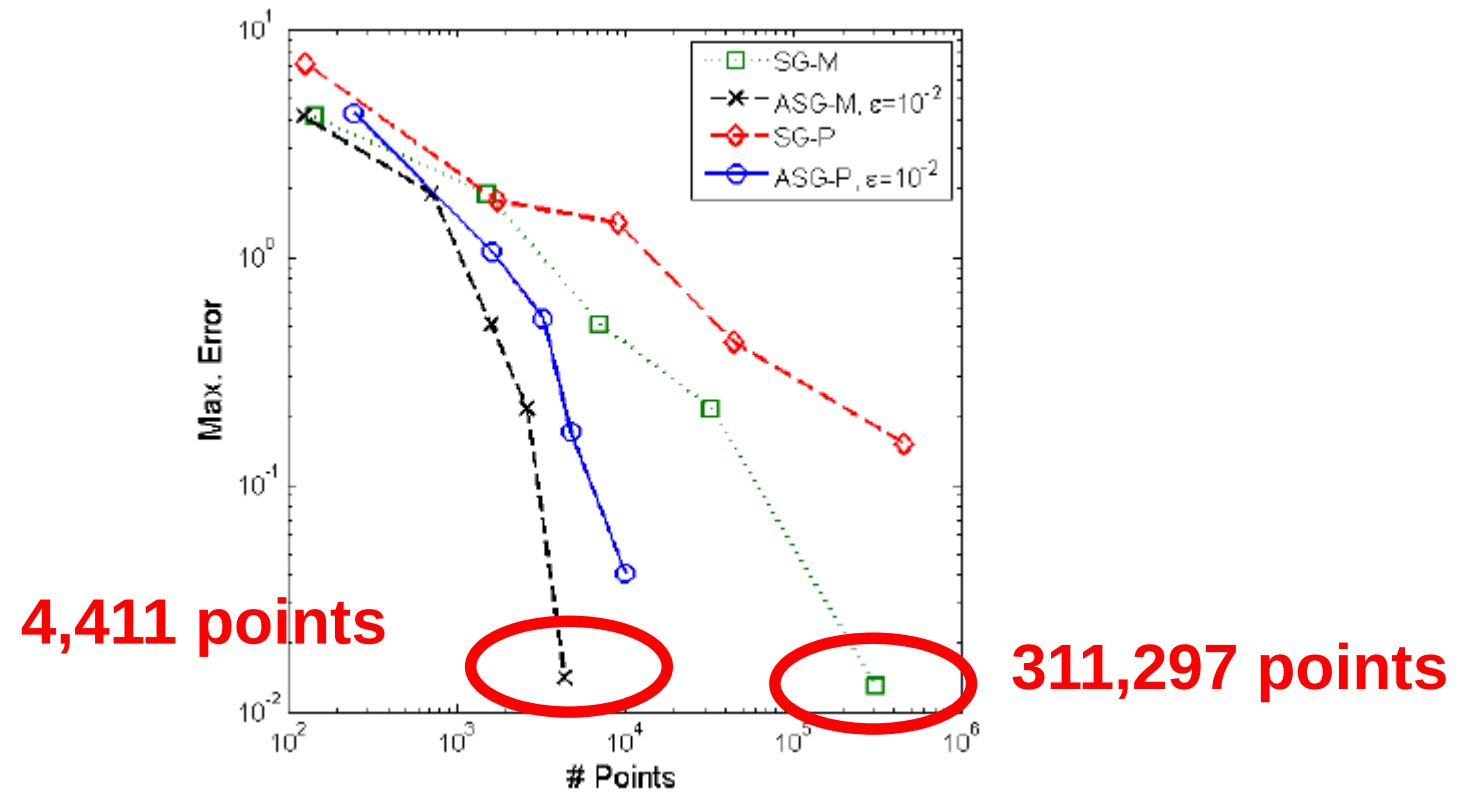
Fig.: Comparison of the interpolation error for **conventional and adaptive sparse grid interpolation** (two different adaptive sparse grid choices).

From Cartesian to adaptive sparse grids
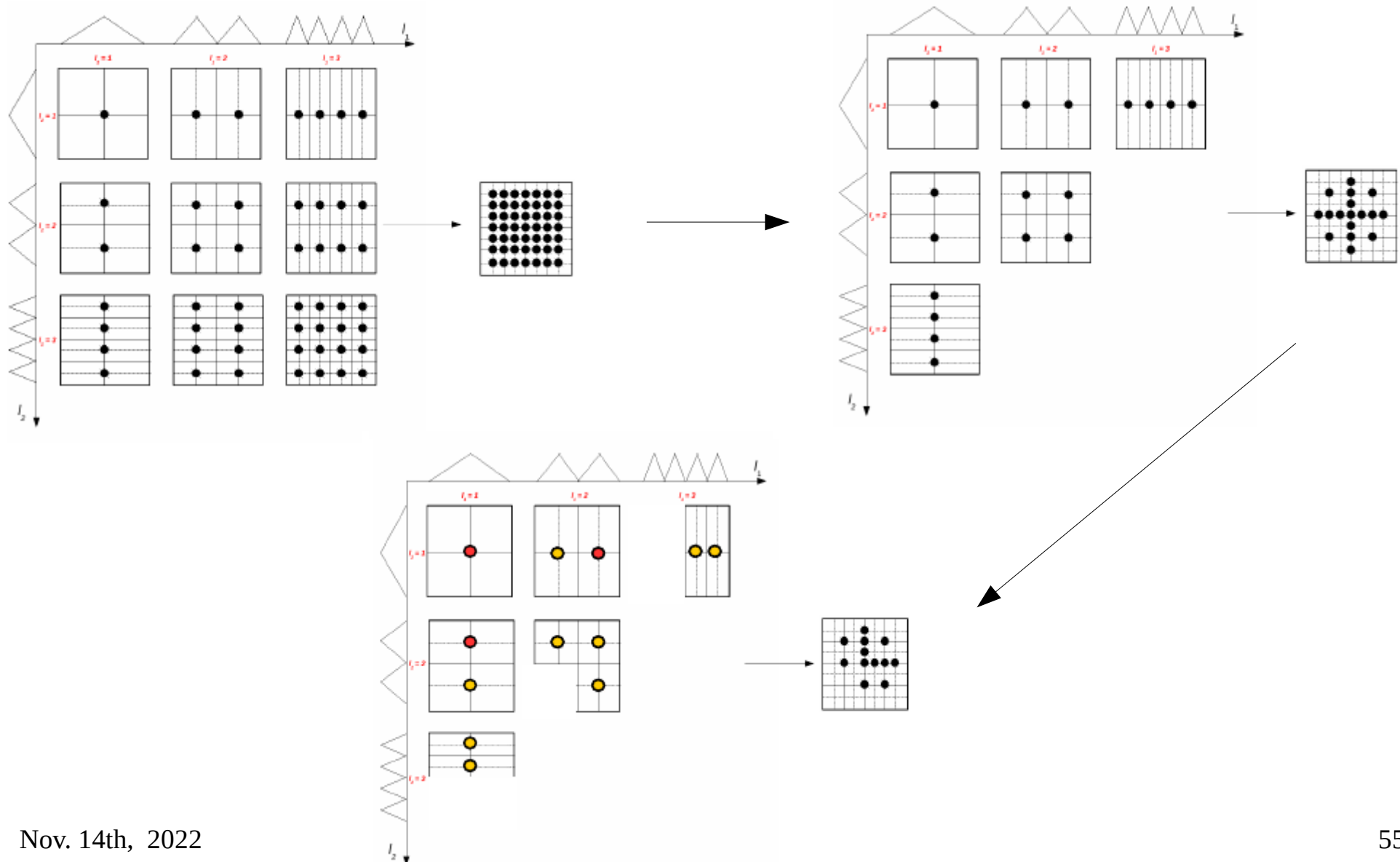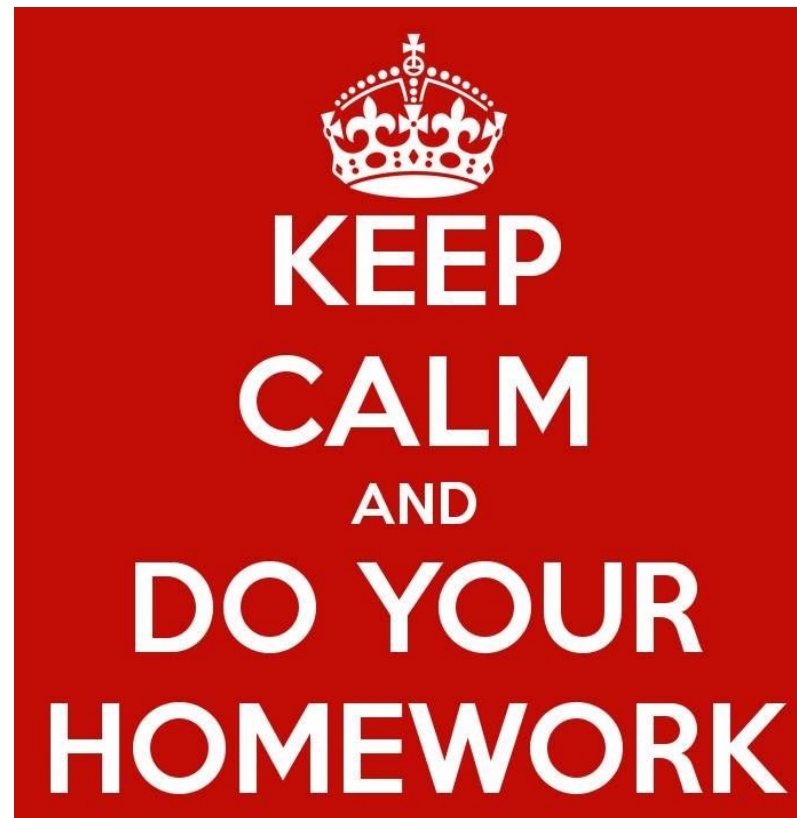
crest_comp_econ/lectures/lecture_1/code/SparseGrids/Analytical_example

Analytical examples

Create sparse grids based on different analytical test functions, e.g. Genz (1984).

→ different test functions can be obtained by varying $c = (c_1, \ldots, c_d)$ (c> 0) and $w = (w_1, \ldots, w_d)$

→ difficulty of functions is monotonically increasing with c.

→ randomly generate 1,000 test points and compute error(s): $e = \max_{i=1,\ldots,1000} |f(\vec{x}_i) - u(\vec{x}_i)|$.

→ **play with adaptive/non-adaptive sparse grids/refinement level and criterion.**

→ generate convergence plots (number of points versus error – as done above).

Genz (1984) test functions

1. OSCILLATORY: $f_1(x) = \cos\left(2\pi w_1 + \sum_{i=1}^{d} c_i x_i\right),$

2. PRODUCT PEAK: $f_2(x) = \prod_{i=1}^{d} \left(c_i^{-2} + (x_i - w_i)^2\right)^{-1},$

3. CORNER PEAK: $f_3(x) = \left(1 + \sum_{i=1}^{d} c_i x_i\right)^{-(d+1)},$

4. GAUSSIAN: $f_4(x) = \exp\left(-\sum_{i=1}^{d} c_i^2 t(x_i - w_i)^2\right),$

5. CONTINUOUS: $f_5(x) = \exp\left(-\sum_{i=1}^{d} c_i |x_i - w_i|\right),$

6. DISCONTINUOUS: $f_6(x) = \begin{cases} 0, & \text{if } x_1 > w_1 \text{ or } x_2 > w_2, \\ \exp\left(\sum_{i=1}^{d} c_i x_i\right), & \text{otherwise.} \end{cases}$

Nov. 14th, 2022

IRBC with Adjustment Costs and Irreversible Investment

- Use test case from JEDC project on computational methods
  (Den Haan et al. 2011, Juillard and Villemot 2011, Kollmann et al. 2011)

- *N* countries facing productivity shocks and capital adjustment costs

  ⇒ countries differ in productivity, *a* (stochastic and exogen.), and capital stock, *k* (endogen.)
  ⇒ dimension of the state space is *2N*
  ⇒ recursive equilibrium is characterized by
     policy $p:R_+^{2N} \Rightarrow R_+^{N+1}$
     mapping state into *N* capital choices and *1* Lagrange multiplier (as markets are complete)

- Extension: Investment in each country is irreversible:

  ⇒ recursive equilibrium is characterized by policy
     $p:R_+^{2N} \Rightarrow R_+^{2N+1}$
     mapping state into *N* capital choices, *1* Lagrange multiplier,
     and *N* Karush-Kuhn-Tucker multipliers

IRBC model (I)

To demonstrate the capabilities of ASGs, consider an IRBC model where investment in each country's capital stock is irreversible as in Brumm and Scheidegger [2017]. As these occasionally binding constraints induce non-differentiabilities in policy functions, they represent a challenge that adaptivity can address. To focus on the dimensionality of the state space and the occasionally binding constraints, the model is kept simple in many other ways.

IRBC model. In the model, there are M countries, $j = 1, \ldots, M$, each using its capital stock, k_j, to produce output, which can be used for investment, χ_j, or for consumption, c_j, generating utility through an additive separable utility function with discount factor β and per-period utility function

$$u_j(c_j) = c_j^{1-\gamma_j}/(1-\gamma_j), \quad \gamma_j > 0. \tag{27}$$

Investment is subject to adjustment costs equal to $k_j \cdot \phi/2 \cdot g_j^2$, where $g_j \equiv k_j'/k_j - 1$ and $\phi > 0$. Furthermore, investment is irreversible, $\chi_j \geq 0$, and depreciates at a rate $\delta > 0$, thus following the law of motion:

$$k_j' = k_j \cdot (1-\delta) + \chi_j. \tag{28}$$

The amount produced by each country is given by $a_j \cdot A \cdot (k_j)^\kappa$, where $A > 0$, and a_j is the country specific productivity level. The latter follows this law of motion:

$$\ln a_j' = \rho \cdot \ln a_j + \sigma \cdot (e_j + e_{M+1}). \tag{29}$$

IRBC model (II)

The parameters ρ and σ determine the persistence and volatility in productivity. The country-specific shocks, $e_j \sim \mathcal{N}(0,1)$, as well as the global shock, $e_{M+1} \sim \mathcal{N}(0,1)$, are assumed to be independent from each other and across time. We follow the comparison study by Kollmann et al. [2011] in assuming complete markets,[13] which implies that the decentralized competitive equilibrium allocation can be obtained as the solution to a social planner's problem: Subject to aggregate resource constraints and irreversibility constraints, maximize the weighted sum of country utility, weighted by welfare weights, τ_j, which depend on the initial capital stocks of the countries.

Recursive structure. In the following, the recursive structure of the model is briefly presented, while the reader is referred to Brumm and Scheidegger [2017] for the derivation of the equilibrium conditions. The state variables of the IRBC model with M countries consist of

$$\mathbf{x} = (a_1, \ldots, a_M, k_1, \ldots, k_M) \in X \subset \mathbb{R}^{2M}, \tag{30}$$

where a_j and k_j are the productivity and capital stock of country j, respectively. The policy function $p : \mathbb{R}^{2M} \rightarrow \mathbb{R}^{2M+1}$ maps the current state into investment choices, χ_j, the multipliers for the irreversibility constraints, μ_j, and the multiplier of the aggregate resource constraint, λ:

$$p(\mathbf{x}) = (\chi_1, \ldots, \chi_M, \mu_1, \ldots, \mu_M, \lambda). \tag{31}$$

IRBC model (III)

The investment choices determine the next period's capital stock in a deterministic way through (28). In contrast, the law of motion of productivity (cf. equation (29)) is stochastic. Taken together, (28) and (29) specify the distribution of x' (corresponding to (25) in the general problem above). The period-to-period equilibrium conditions of this model (corresponding to (26)) consist of three types of equations. First, the optimality conditions for investment in capital in each country j:

$$\lambda \cdot \left[1 + \phi \cdot g_j\right] - \mu_j - \beta \cdot \mathbb{E}_t \left\{\lambda' \left[a'_j \cdot A \cdot \kappa \cdot (k'_j)^{\kappa-1} + 1 - \delta + \frac{\phi}{2} \cdot g'_j \cdot \left(g'_j + 2\right)\right] - \mu'_j \cdot (1 - \delta)\right\} = 0. \quad (32)$$

Second, the irreversibility constraint and the associated complementary condition for each country j,

$$\chi_j \geq 0, \mu_j \geq 0, \chi_j \cdot \mu_j = 0. \quad (33)$$

Finally, using $c_j = (\lambda/\tau_j)^{-\gamma_j}$, the aggregate resource constraint is

$$\sum_{j=1}^{M} \left(a_j \cdot A \cdot (k_j)^{\kappa} - k_j \cdot \frac{\phi}{2} \cdot (g_j)^2 - \chi_j - c_j\right) = 0. \quad (34)$$

→ **crest_comp_econ/lectures/lecture_1/code/SparseGrids/IRBC_model**

Time Iteration Algorithm

Algorithm 1 Overview of the crucial steps of the time-iteration algorithm.

Time-Iteration Algorithm:

1. Make an initial guess p_{init} for next period's policy function. Set $p_{next} = p_{init}$. Choose an approximation accuracy $\bar{\eta}$.

2. Make one time-iteration step:
   (a) Choose a maximal refinement level L_{max} and a fixed grid level $L_0 \leq L_{max}$. Set $l = 1$, set $G \subset X$ to be the level 1 grid on X, and set $G_{old} = \emptyset$.
   (b) For each $g \in G \setminus G_{old}$ compute the optimal policies $p(g)$ by solving the system of equilibrium conditions

   $$0 = \mathbb{E}\left\{f\left(g, x_{t+1}, p(g), p_{next}(x_{t+1})\right) \middle| g, p(g)\right\},$$
   $$x_{t+1} \sim F(\cdot|g, p(g)),$$

   given next period's policy p_{next}. Construct the hierarchical surpluses of level l.
   (c) Generate G_{new} from G by adding for each $g \in G_l \setminus G_{old}$ its $2d$ neighbouring points, if either $l < L_0$ or

   $$\|p(g) - \tilde{p}(g)\|_\infty > \varepsilon,$$

   where the policy $\tilde{p}(g)$ is given by interpolating between $\{p(g)\}_{g \in G_{old}}$. Note that if G_{old} is of level 2, then each point does not have $2d$ but only d neighbouring points (cf. Fig. 3.4).
   (d) If $G_{new} = G$ or $l = L_{max}$, then set $G = G_{new}$ and go to (e), else set $G = G_{new}$, $l = l + 1$ and go to (b).
   (e) Define the policy function p as the sparse grid interpolation of $\{p(g)\}_{g \in G}$.
   (f) Calculate (an approximation for) the error, e.g.

   $$\eta = \|p - p_{next}\|_\infty.$$

   If $\eta > \bar{\eta}$, set $p_{next} = p$ and go to (a), else go to step 3.

3. The (approximate) equilibrium policy function is given by p.

Time Iteration Algorithm Details

N Euler equations

$$g_{t+1}^j := k_{t+1}^j / k_t^j - 1, \quad g_{t+2}^j := k_{t+2}^j / k_{t+1}^j - 1,$$

$$\forall j : \lambda_t \cdot \left[1 + \phi \cdot g_{t+1}^j \right] - \beta \cdot \mathbb{E}_t \left\{ \lambda_{t+1} \cdot \left[a_{t+1}^j \cdot A \cdot \alpha \cdot (k_{t+1}^j)^{\alpha-1} + (1-\delta) + \frac{\phi}{2} \cdot g_{t+2}^j \cdot \left(g_{t+2}^j + 2 \right) \right] \right\} = 0,$$

$$\sum_{j=1}^{N} \left(a_t^j \cdot A \cdot (k_t^j)^\alpha + k_t^j \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g_{t+1}^j)^2 \right) - k_{t+1}^j - \left(\frac{\lambda_t}{\tau_j} \right)^{-\gamma^j} \right) = 0.$$

1 aggregate resource constraint

$$\left(a_t^1, \ldots, a_t^N, k_t^1, \ldots, k_t^N \right) \longrightarrow \text{\textbf{Current grid points}}$$

solve for the unknown policy variables

$$\left(k_{t+1}^1, \ldots, k_{t+1}^N, \lambda_t \right) \longrightarrow \text{\textbf{Solution of system of Eqs. At the current iteration}}$$

given the known policy functions from last iteration

$$\left(k_{t+2}^1(a_{t+1}, k_{t+1}), \ldots, k_{t+2}^N(a_{t+1}, k_{t+1}), \lambda_{t+1}(a_{t+1}, k_{t+1}) \right) \longrightarrow \text{\textbf{Interpolant}}$$

evaluated at next periods state

$$\left(a_t^1, \ldots, a_t^N, k_t^1, \ldots, k_t^N \right), \text{ where } a_{t+1}^j = (a_t^j)^\rho \cdot e^{\sigma(e_t + e_t^j)}.$$

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.

Results for Non-Smooth IRBC Model: Finer Grids

Non-adaptive sparse grid has problems with non-smooth model:

Dimension	Level	Points	Max. Error	Avg. Error
4	5	401	-2.11	-2.93
4	7	2,929	-2.32	-3.12
4	9	18,945	-2.45	-3.32

Adaptive sparse grid can overcome problems with non-smooth model
→ they provide higher accuracy with less points:

ϵ	Max. Level Reached	Points	Max. Error	Avg. Error
0.01	7 (2,929)	245	-2.23	-2.88
0.005	9 (1,945)	559	-2.42	-2.98
0.0025	13 (643,073)	2,346	-2.68	-3.32
0.001	14 (3,502,081)	14,226	-2.91	-3.73

All errors are given in log 10 -scale.

IRBC with irreversible investment

Capital in country 1 low:
→ investment opportunities better than in country 2
→ irrev. constraint binding for country 2.

Capital in country 1 high:
→ irrev. constraint binding for country 1.
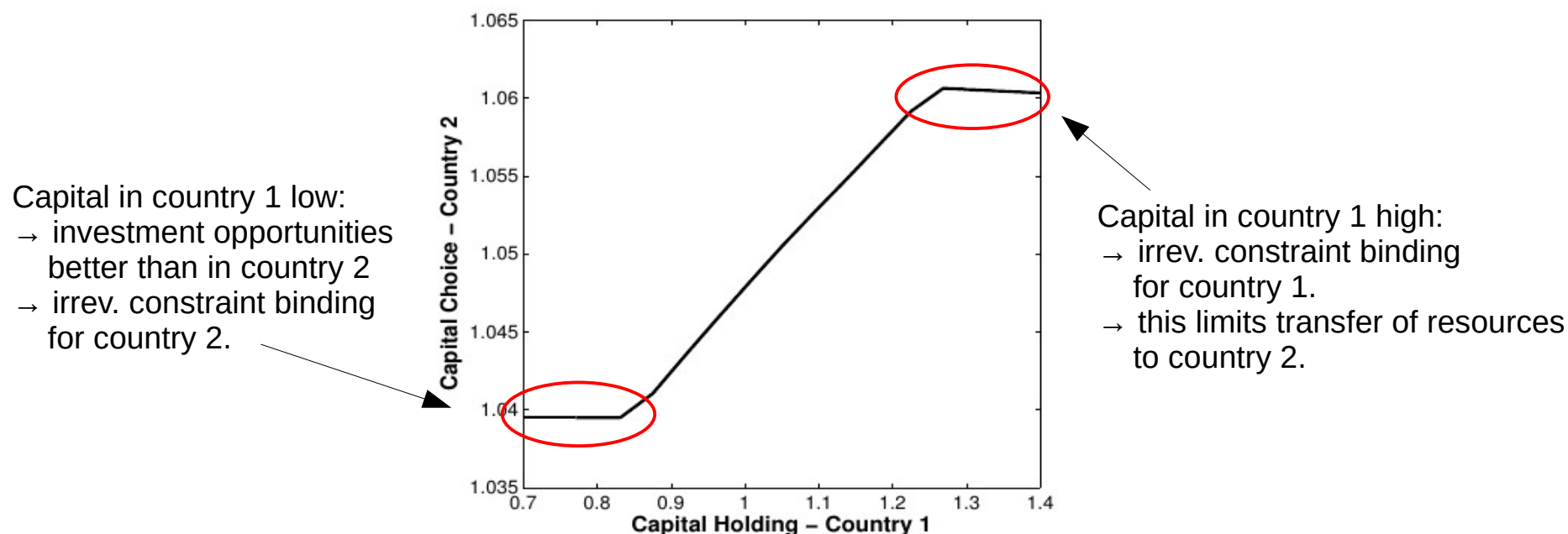→ this limits transfer of resources to country 2.

Fig.: Capital choice of country 2 as a function of capital holding of country 1. All other state variables of this model are kept fixed at steady state (2N = 4d). The 4-d policy function was interpolated on an adaptive sparse grid (ε = 0.0033).

Note: **kink is (2N -1) - dimensional hypersurface in 2N - dim state space.**
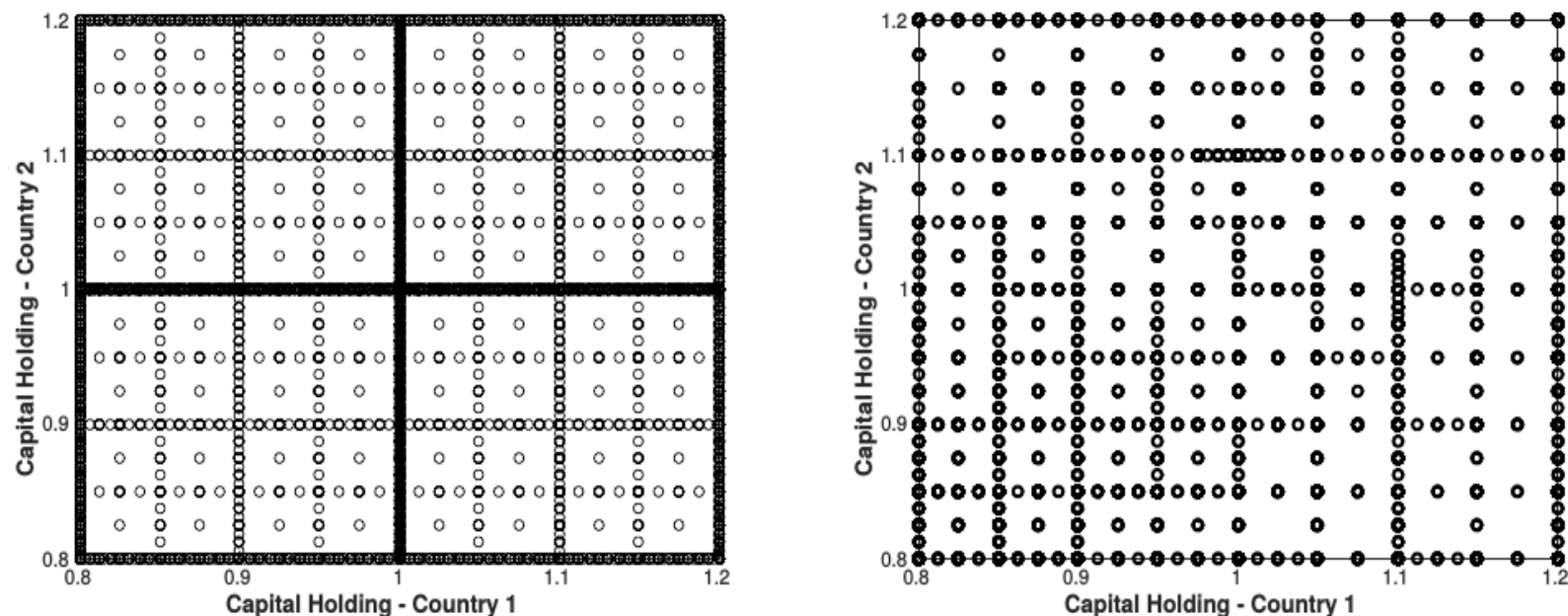
IRBC with binding constraints

Fig.: 2-d projections of two different grids.
Left: `classical' sparse grid of level 9 (18,945 points),
Right: adaptive grid with refinement threshold ε = 0.001 (14,226 points).
The x-axis shows capital holding of country 1, the y-axis shows capital holding
of country 2, while the productivities of the two countries are kept fixed at
their unconditional means.

Models with binding constrains: massive speedup due to adaptivity

Dimension	Points	Max. Error	Avg. Error	$|V_8^{S,CC}|$
4	245	-2.22	-2.88	18,945
6	684	-2.26	-2.73	127,105
8	931	-2.02	-2.66	609,025
10	2,790	-1.97	-2.54	2,148,960
12	4,239	-1.81	-2.48	7,451,394
16	8,569	-1.94	-2.36	52,789,761
20	9,098	-1.96	-2.35	$\gg 10^8$

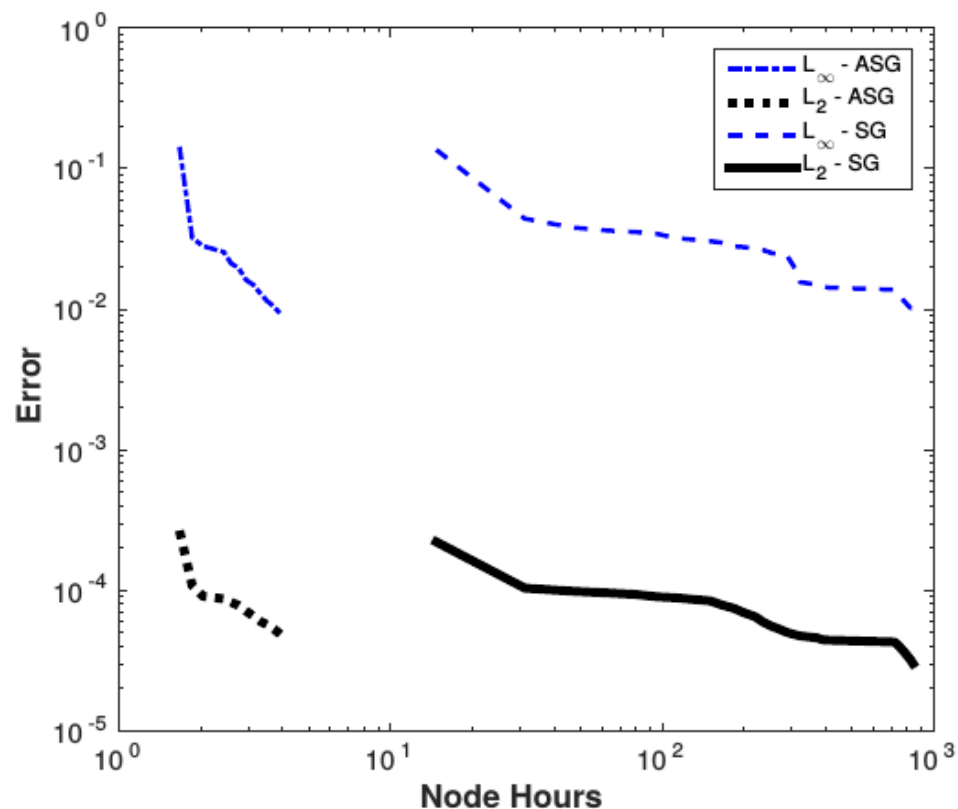**Tab.**: Comparison of a sparse and adapt. sparse grid of comparable accuracy.

Fig.: 8d model with binding constraints.
model run with/without adaptive sparse grids.
Relative error among two consecutive time-steps.
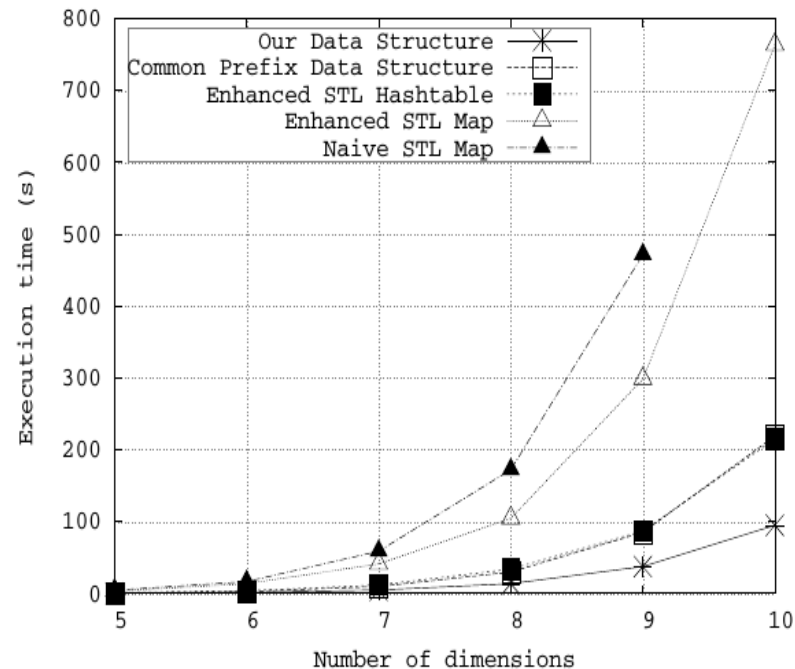10k points drawn from uniform distribution.

Questions

3. HDMR

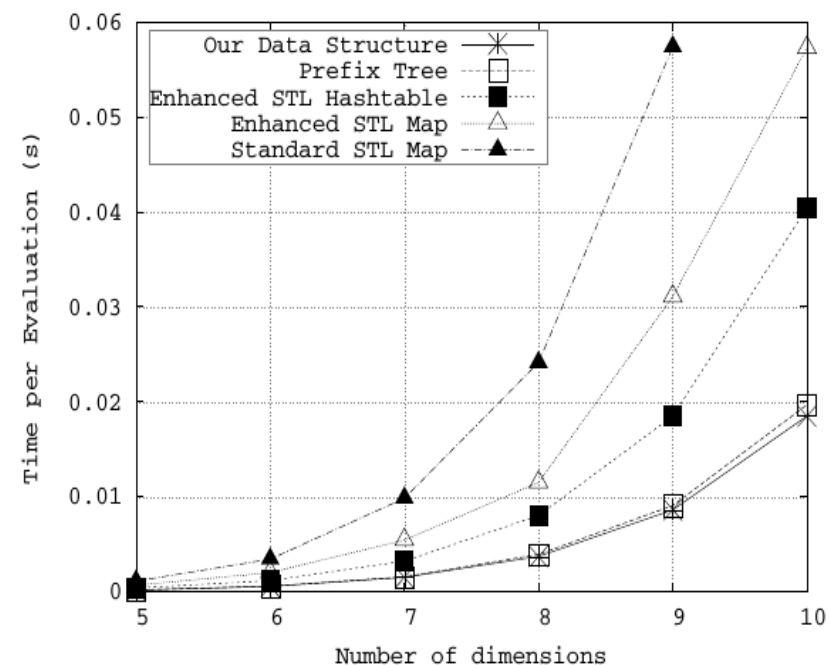So this, then, was the kernel of the brute!

(Johann Wolfgang von Goethe)

Limitation of sparse grids: Execution times in higher dimension

(a) Runtime for sequential hierarchization.

(b) Runtime for sequential evaluation.

going to higher dimensions gets polynomially harder → we need parallel programming

Figs. From Murarasu et al. (2011)
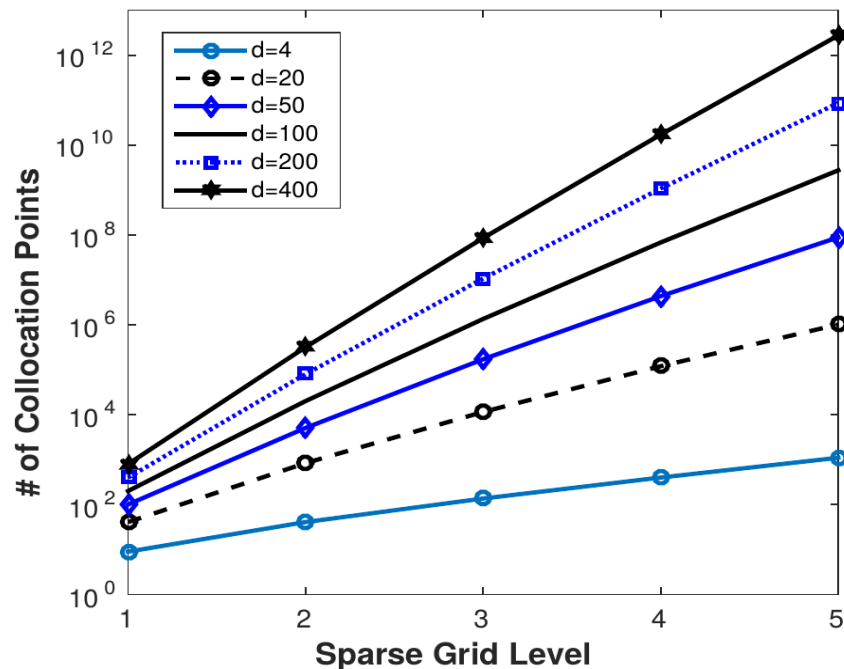
Limitations of sparse grids (II)

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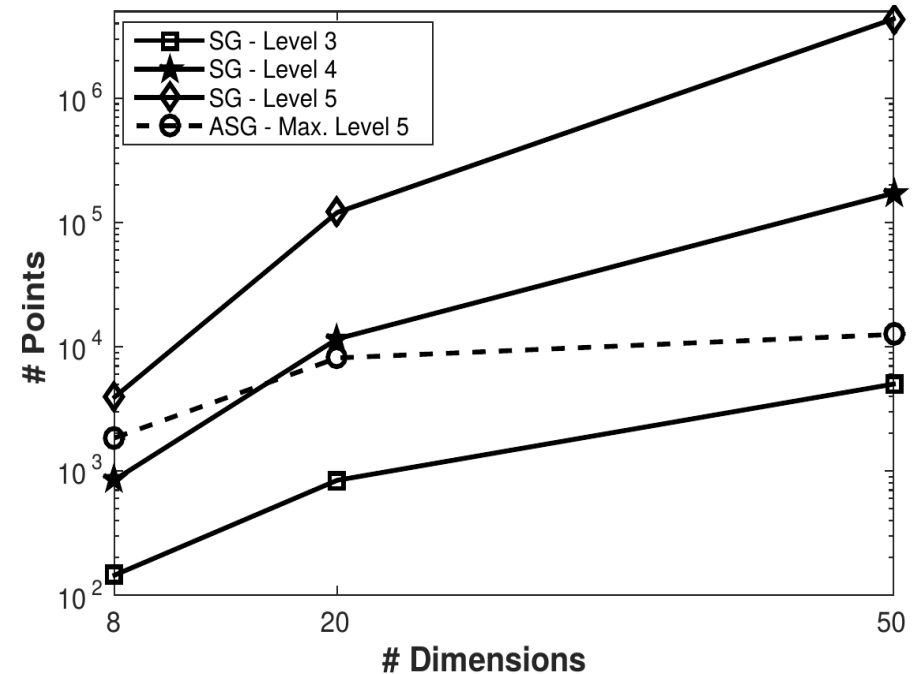
Fig.: IRBC model, solved both with classical sparse grids of varying dimension and increasing refinement level.

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)**.

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 $f(\mathbf{Y}) : \mathbb{R}^N \to \mathbb{R}$ 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

<u>Cut-HDMR</u>

(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 x_u are

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

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.
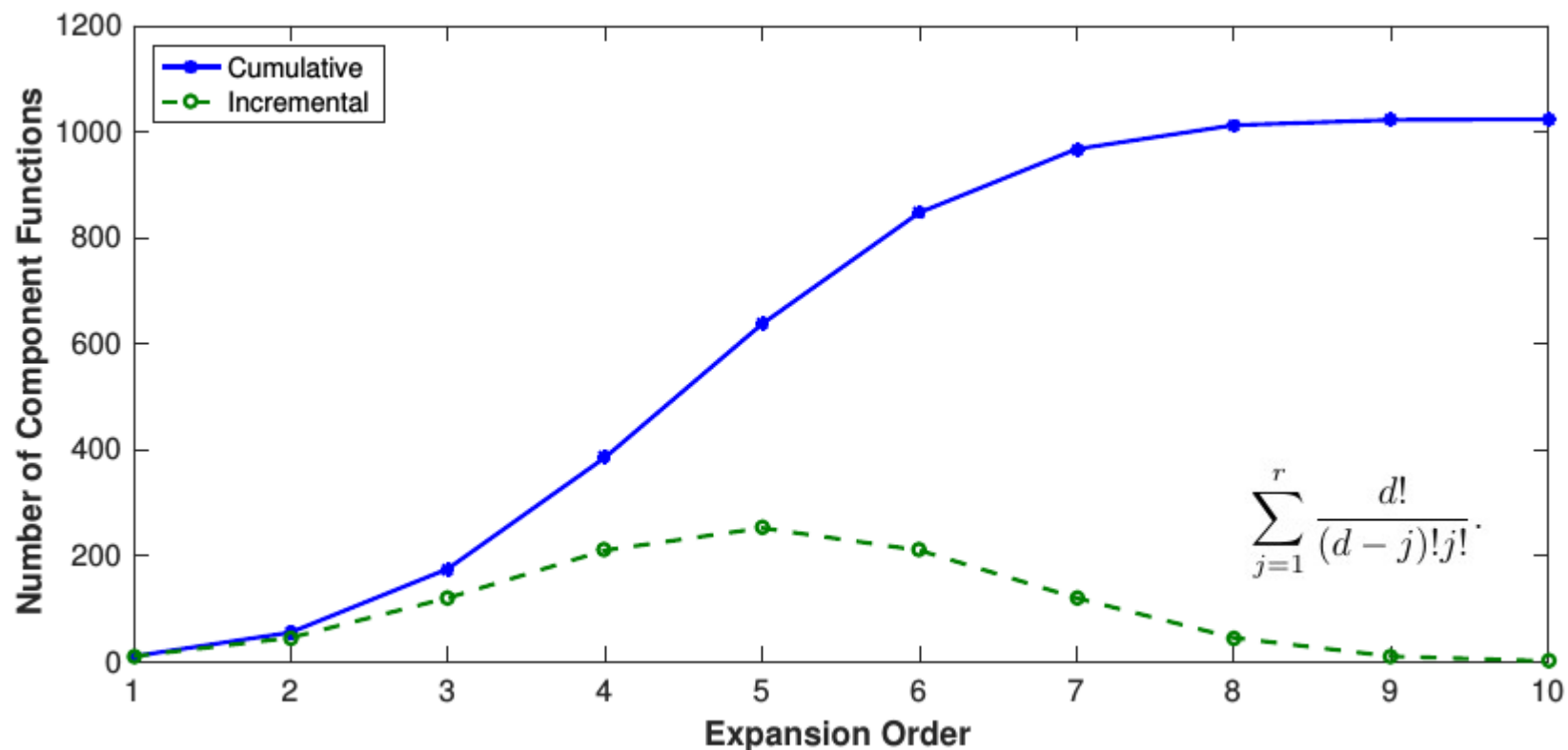
HDMR expansion in large dimensions

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 $\epsilon_n \geq 0$.
We **reject component functions** containing all indices of u if $\eta_{\mathbf{u}} < \epsilon_{\eta}$.

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

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

→ **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:
$$f(\mathbf{x}) = \sum_{\mathbf{u} \subseteq \mathcal{S}} f_{\mathbf{u}}(\mathbf{x_u}), \quad f_{\mathbf{u}}(\mathbf{x_u}) = f(\mathbf{x})|_{\mathbf{x} = \bar{\mathbf{x}} \backslash \mathbf{x_u}} - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x_v}),$$

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.

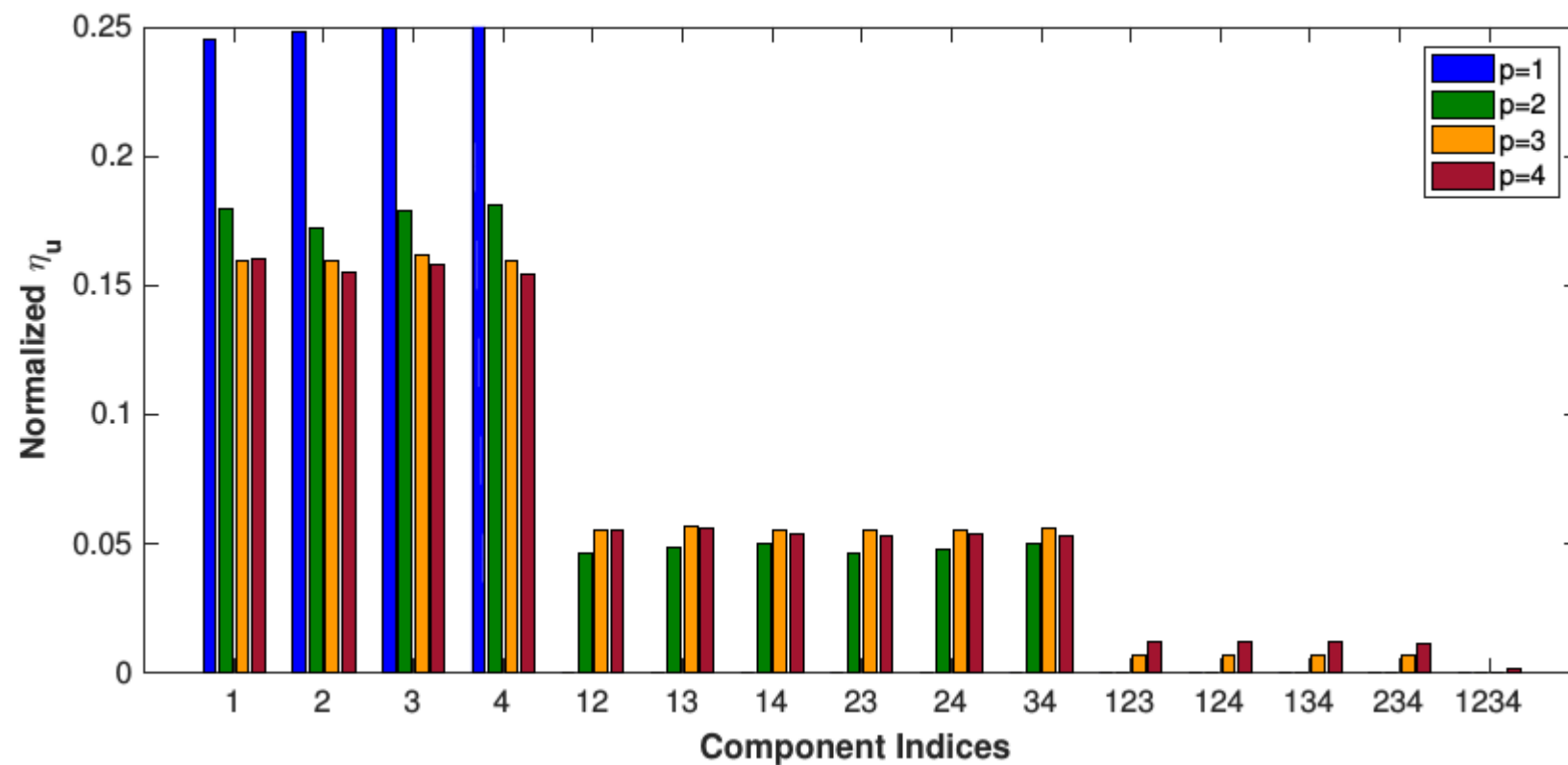$$f_{\mathbf{u}}(\mathbf{x_u}) \approx \mathcal{SG} f(\mathbf{x})|_{\mathbf{x} = \bar{\mathbf{x}} \backslash \mathbf{x_u}} - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x_v})$$

$$\approx \sum_{\|\mathbf{k}\|_1 \leq \ell + d - 1} \sum_{i \in \mathbf{I_k}} \alpha_{i,k} \Phi_{i,k}(\mathbf{x})|_{\mathbf{x} = \bar{\mathbf{x}}} - \sum_{\mathbf{v} \subset \mathbf{u}} f_{\mathbf{v}}(\mathbf{x_v})$$

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

→ 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.

lectures/crest_comp_econ/lectures/lecture_1/code/HDMR

Parallelization

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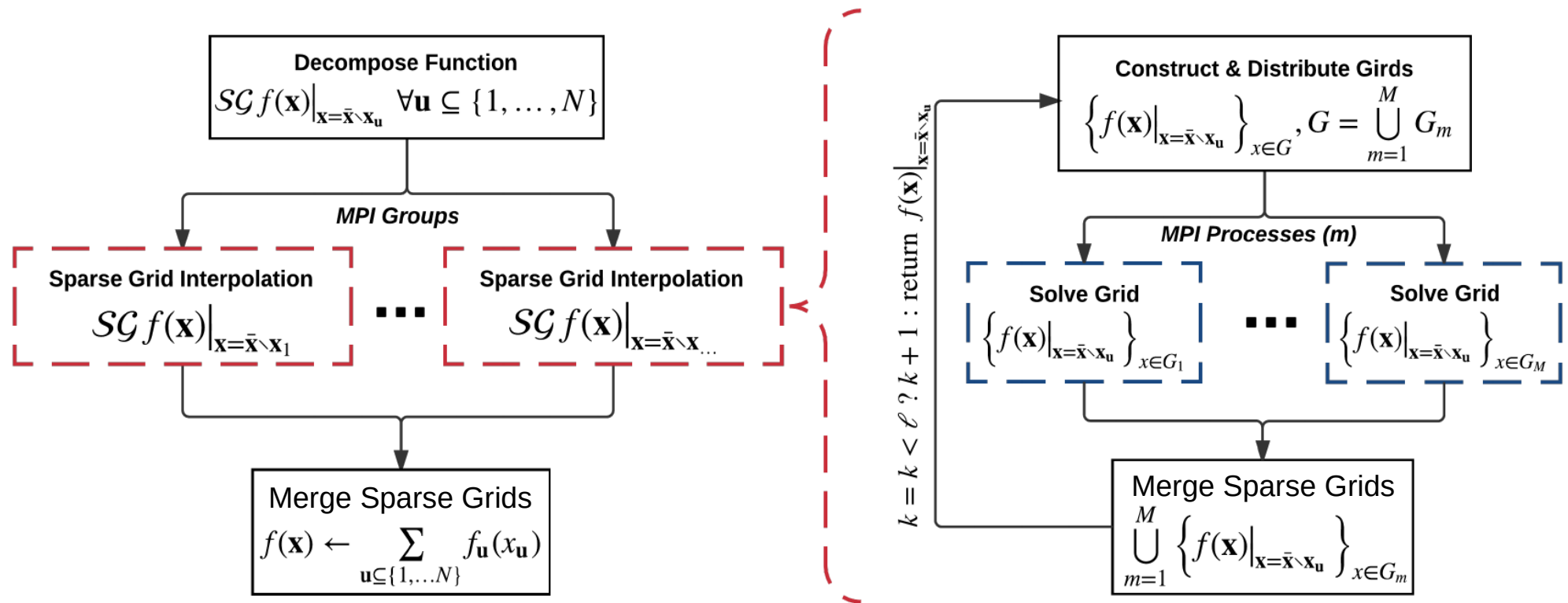→ an additional layer of parallelism (MPI_Groups) on top of sparse grid parallelism

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.