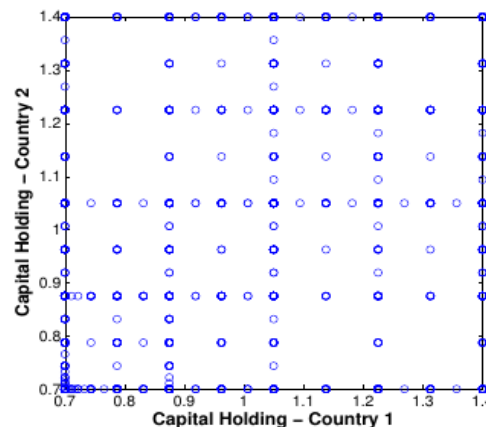


# Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models

Simon Scheidegger  
simon.scheidegger@gmail.com

July 16<sup>th</sup>, 2019

Open Source Economics Laboratory – BFI/UChicago  
(joint work with J. Brumm, Econometrica)



# Who I am?

- Assistant Prof, University of Lausanne, Department of Finance  
(<https://sites.google.com/site/simonscheidegger>)
- Ph.D., theoretical Physics, University of Basel (2010)
- Dissertation title:  
*“Gravitational waves from 3D MHD core-collapse supernova simulations with neutrino transport”*

## Research interests:

Research focus on developing computational methods for high-dimensional dynamic stochastic economic models and applying them to optimal tax policy, monetary policy and option pricing.

## Teaching:

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

# Roadmap – fast forward:

## Day 1, Tuesday – July 16<sup>th</sup>

1. (Adaptive) Sparse Grids (8.00 – 9.00)
2. Hands-on Session (I) (9.30 – 10.30):
  - log onto MIDWAY cluster (→ more details in parallel programming lecture).
  - install & play with open source sparse grid codes (→ analytical functions).
3. Hands-on Session (II)
  - Dynamic Programming & Time Iteration with Sparse Grids (10.45 – 11.45).
4. An exercise sheet related to the day (11.45 – 12.00).
  - Analytical examples.
  - Introduction to a stochastic growth model, solved with DP and sparse grids.

# Roadmap – fast forward (2):

## Day 2, Thursday – July 18<sup>th</sup> \_

1. Introduction to parallel and high-performance computing (8.00-9.30).
2. An ultra-short introduction to C++ (10.00-10.30 – hands on).
3. Basics on code optimization & OpenMP session I (10.45-11.45 – hands on).
4. Introduction to Projects (11.45-12.00 – hands on).
5. Exercise sheet related to the day's topics (12-00-..).

# Roadmap – fast forward (3):

## Day 3, Tuesday – July 23<sup>th</sup>

1. OpenMP session II (8.00-9.00 – hands on).
2. MPI session I (9.30-10.30 – hands on).
3. MPI session II (10.45-11.45 – hands on).
4. Exercise sheet related to the day's topics (11.50-12.00).

## Day 4, Thursday – July 25<sup>th</sup>

1. Hybrid parallelism – OpenMP & MPI (8.00-9.00 – hands on).
2. Hybridize some of the projects together (9.15-10.00 – hands on).
3. Advanced topics (10.15-11.00).
4. Start to present results from the projects (11.10 – 11.50).
5. Exercise sheet related to the day's topic (11.50-12.00 – hands on).

What are these lectures about?



# What are my goals for the next 4 days?

- You know how to deal with high-dimensional state spaces.
- You understand the basic concepts of parallel computing.
- You understand the basics of the available HPC hardware.
- Know which parallel programming paradigms are available.
- Be aware which paradigm and which hardware fits your problem.
- Gain hands-on expertise with exercises.

# Lecture Slides & Codes on Git

I will post the slides and codes for this session on sparse grids as well as for the parallel programming session here:

**<https://github.com/sischei/OSE2019.git>**





KEEP  
CALM  
AND  
LET'S GET  
STARTED

# Today – (Adaptive) Sparse Grids

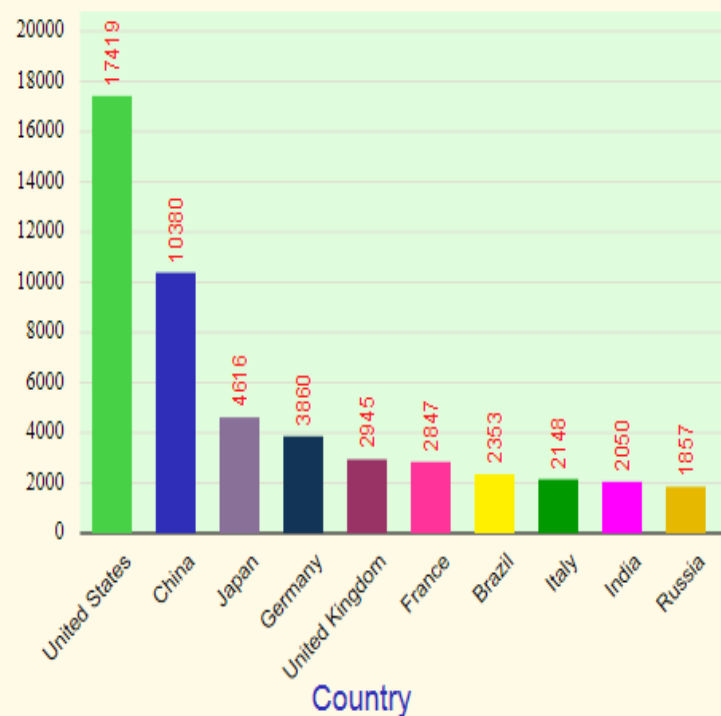
- I. Motivation – “the curse of dimensionality”
- II. From Full (Cartesian) Grids to Sparse Grids
- III. Adaptive Sparse Grids
- IV. How to integrate ASGs in dynamic economic model

# Example – Heterogeneity in IRBC models

- Model trade imbalance
- FX rates
- ...



Top 10 countries by GDP (Nominal) 2014



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

→ **Model heterogeneous & high-dimensional**

# Example – Heterogeneity in OLG\* models

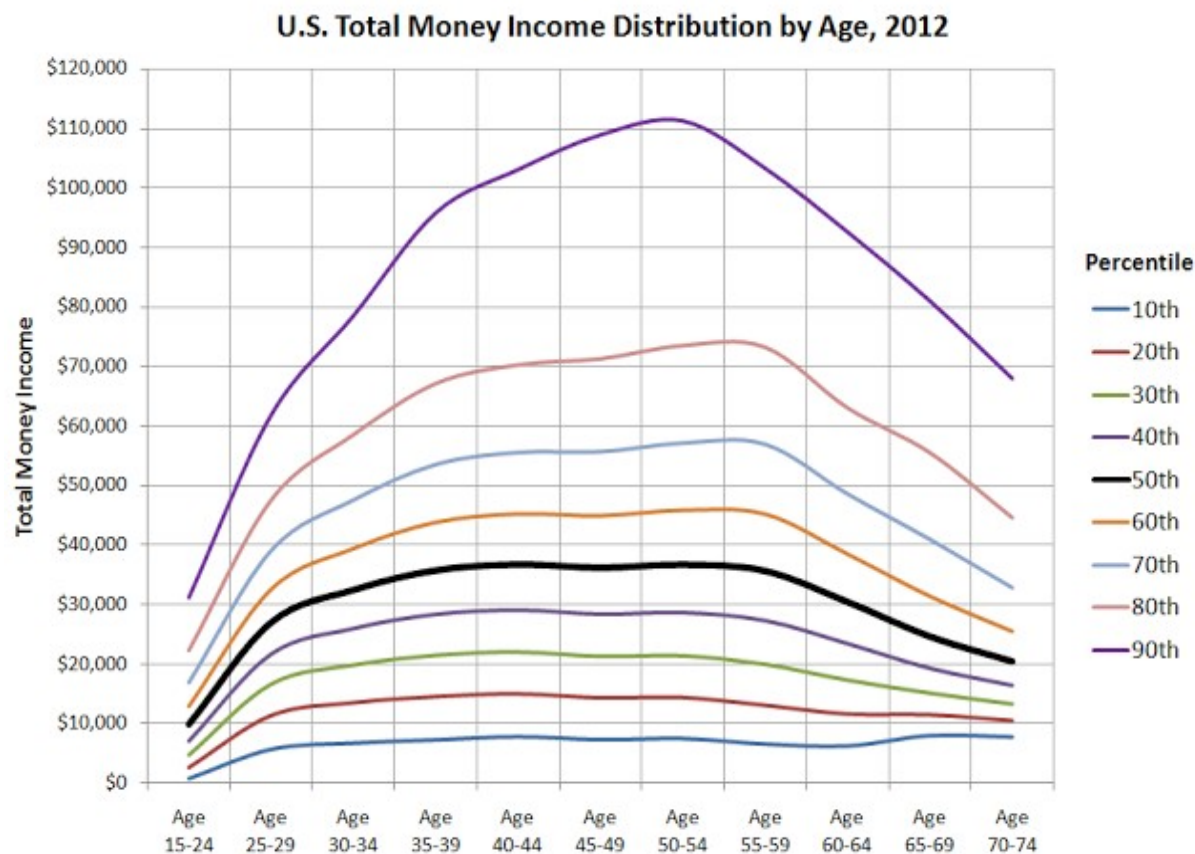
\*Overlapping generation models



To model e.g. social security:

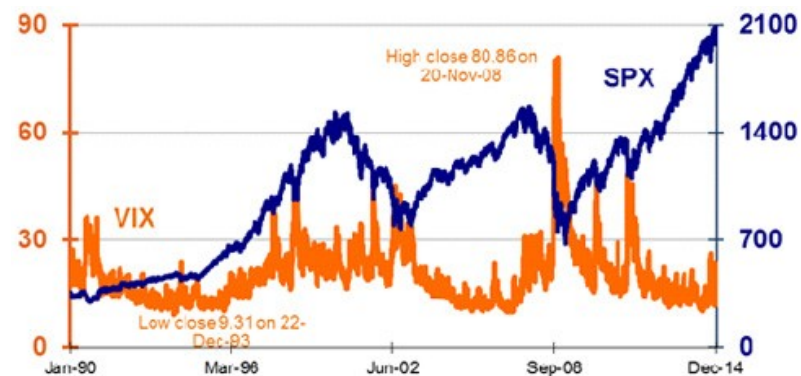
- How many age groups?
- borrowing constraints?
- aggregate shocks?
- ...

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



# Financial markets: non-Gaussian returns

- Derivative contracts giving a right to buy or sell an underlying security.
  - *European* if exercise at expiration only.
  - **American** if exercise any time until expiration.
- American options are extremely challenging:
  - **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  $\mathbf{p}$  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  $j \rightarrow \infty$  by iterations on:

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

s.t.

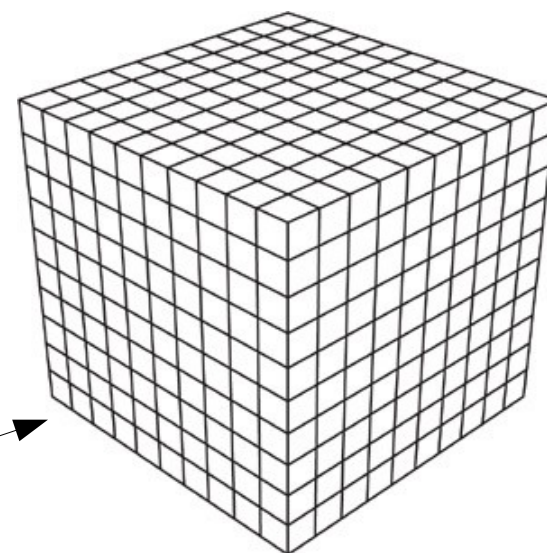
$$\tilde{x} = g(x, u)$$

$\mathbf{x}$ : 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 dimensions is high dimensions?



# How many is dimensions is high dimensions?

| <b>Number of parameters</b><br>(the dimension) | <b>Number of model runs</b><br>(at 10 points per dimension) | <b>Time for parameter study</b><br>(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<br>(240x age of the universe)           |



# How many is dimensions is high dimensions?

| Number of parameters<br>(the dimension) | Number of model runs<br>(at 10 points per dimension) | Time for parameter study<br>(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<br>(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<br>(the dimension) | Number of model runs<br>(at 10 points per dimension) | Time for parameter study<br>(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<br>(240x age of the universe)    |

## Dimension reduction

*Exploit symmetries,...*

## Deal with #Points

*Adaptive Sparse Grids*

## High-performance computing

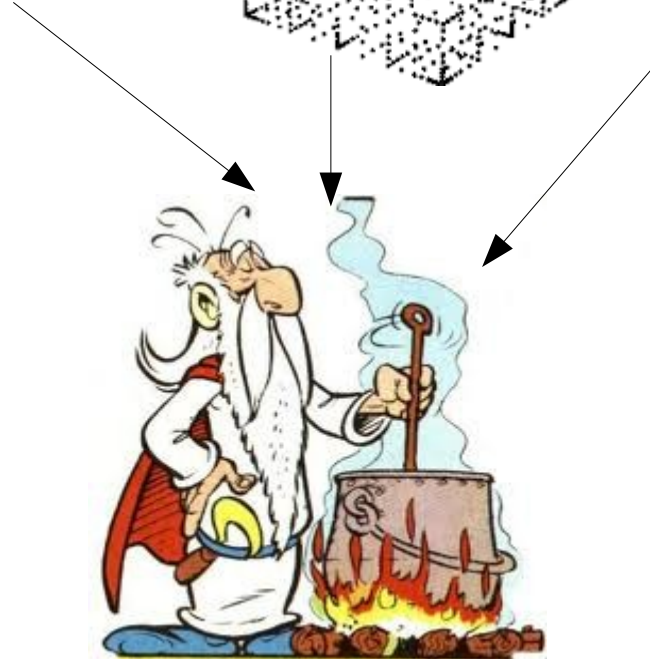
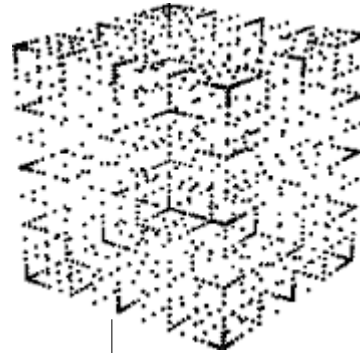
*Reduces time to solution, but not the problem size*

# Computational modelling

$$\lambda_t \cdot \left[ 1 + \phi \cdot g_{t+1}^j \right] - \mu_t^j$$

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

$$0 \leq \mu_t^j \perp (k_{t+1}^j - k_t^j(1 - \delta)) \geq 0.$$



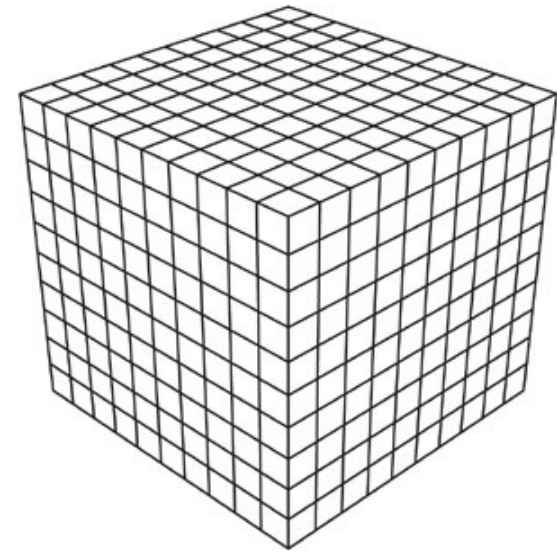
# Abstract Problem Formulation

- 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)

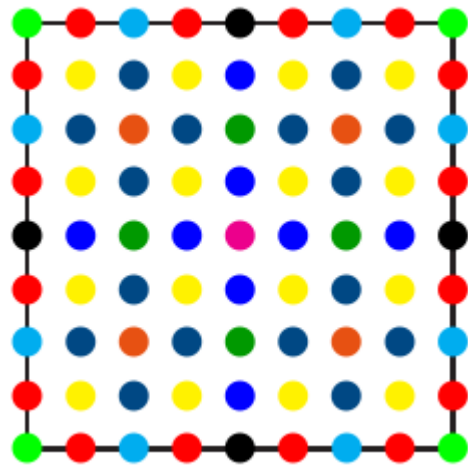
# Models where high-dim. state spaces show up

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

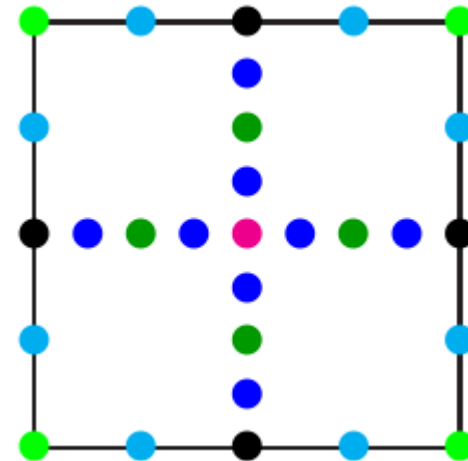
- International Real Business Cycle (IRBC) Models:  
**Exchange Rates, Global Trade Imbalances**
- Dynamic Stochastic General Equilibrium (DSGE) Models:  
**Monetary Policy, Business Cycle Fluctuations**
- Overlapping Generations (OLG) Models:  
**Demographic Change, Social Security**
- Mathematical 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:  $\alpha_i$
- 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:**

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

with the **index set**

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

The corresponding function space:

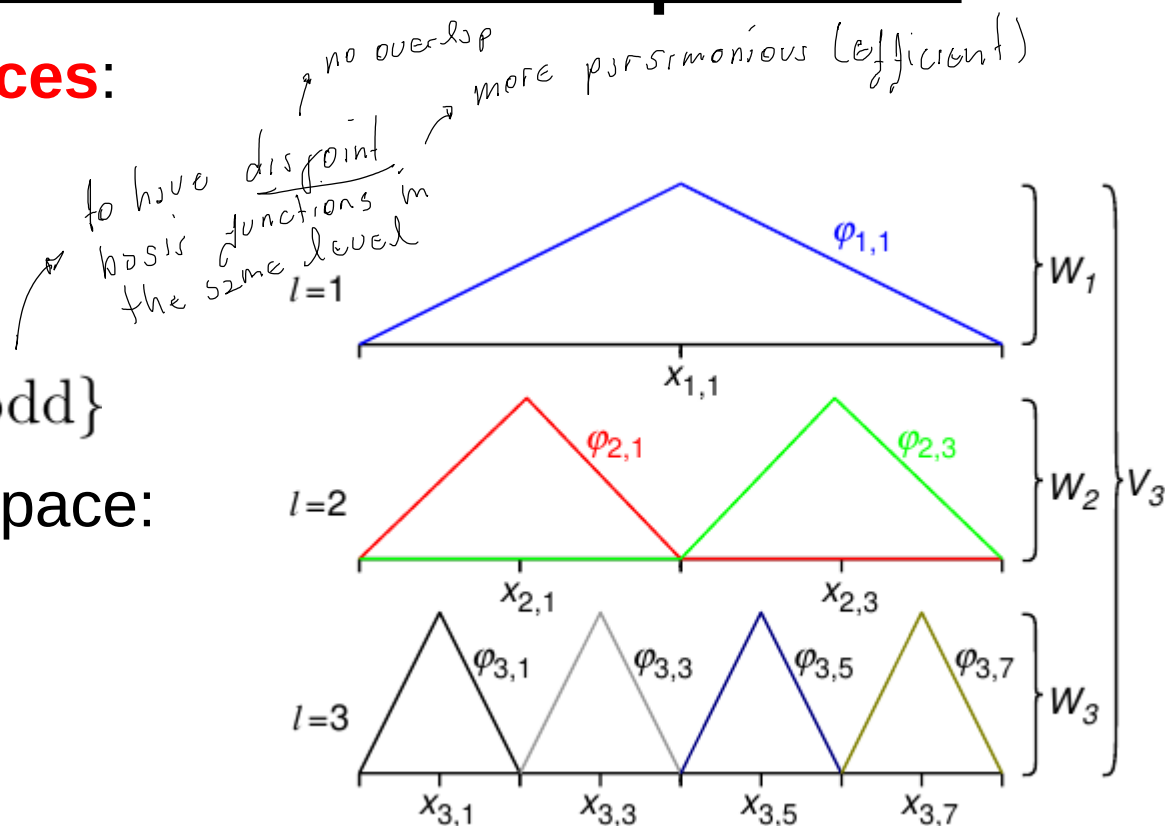
$$V_l = \bigoplus_{k \leq l} W_k$$

*sum of functions*  
↳ combination

The **1d-interpolant:**

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

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



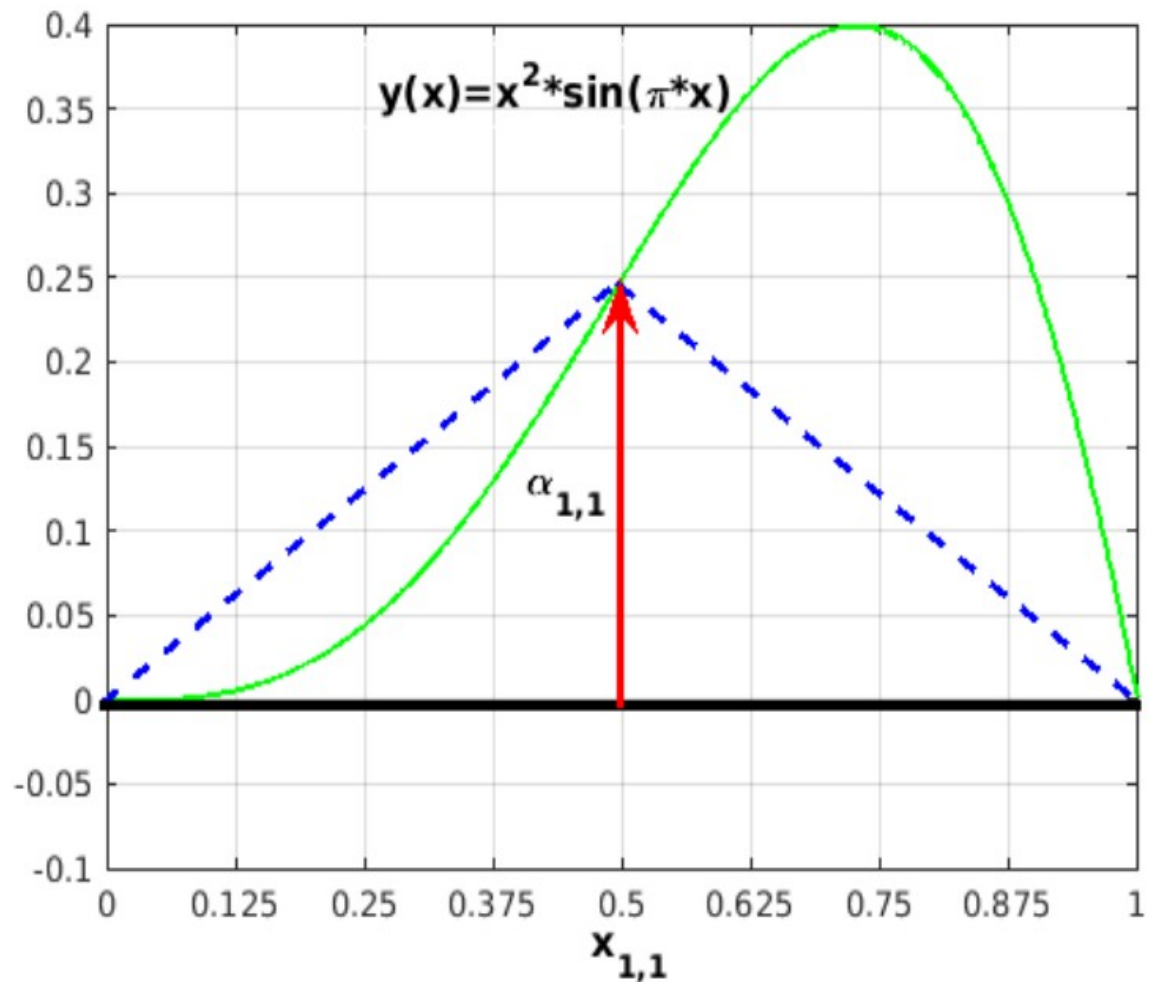
**Fig.: 1-d basis functions  $\phi_{l,i}$  and the corresponding grid points up level  $l=3$  in the hierarchical basis.**

# 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.**

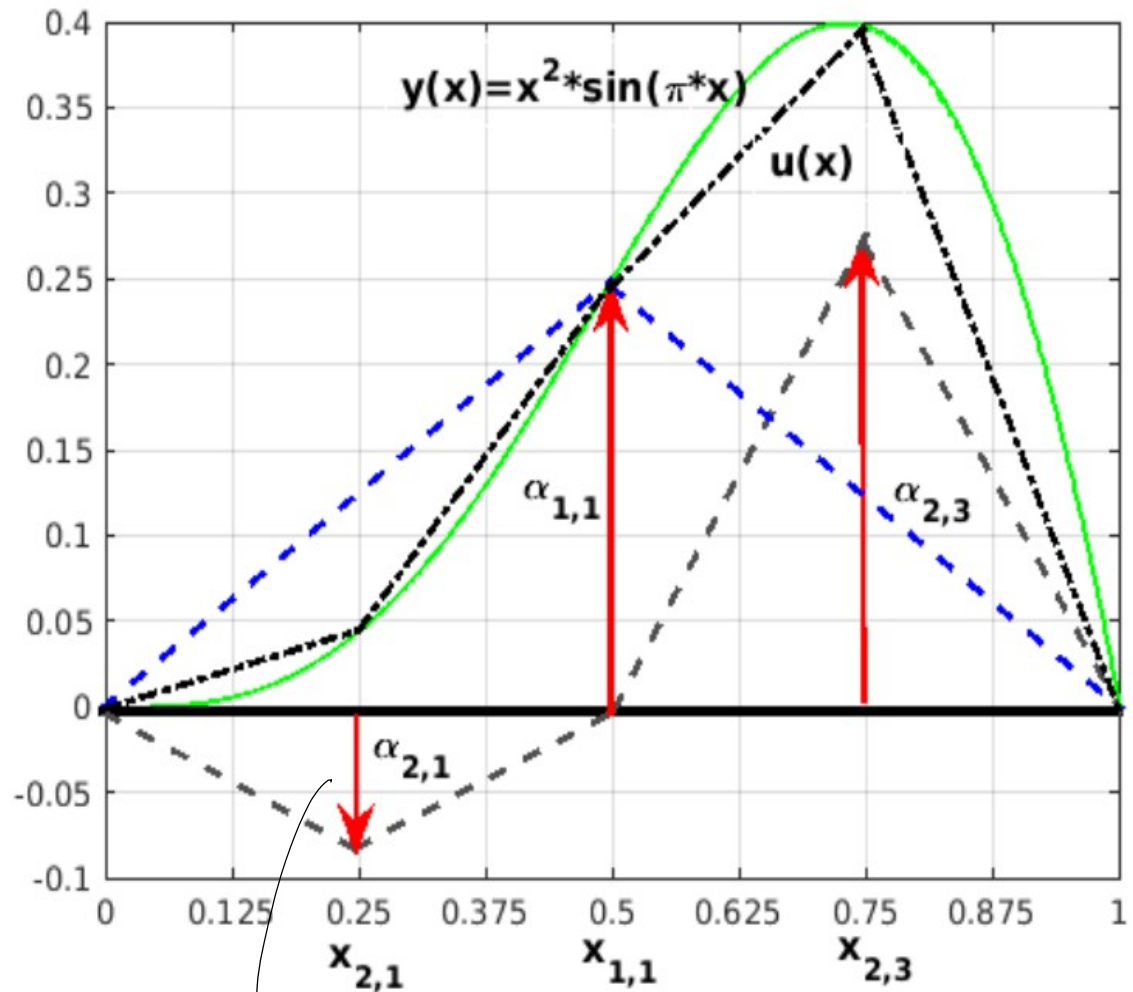


# Piecewise Linear Interpolation: Level II

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.**



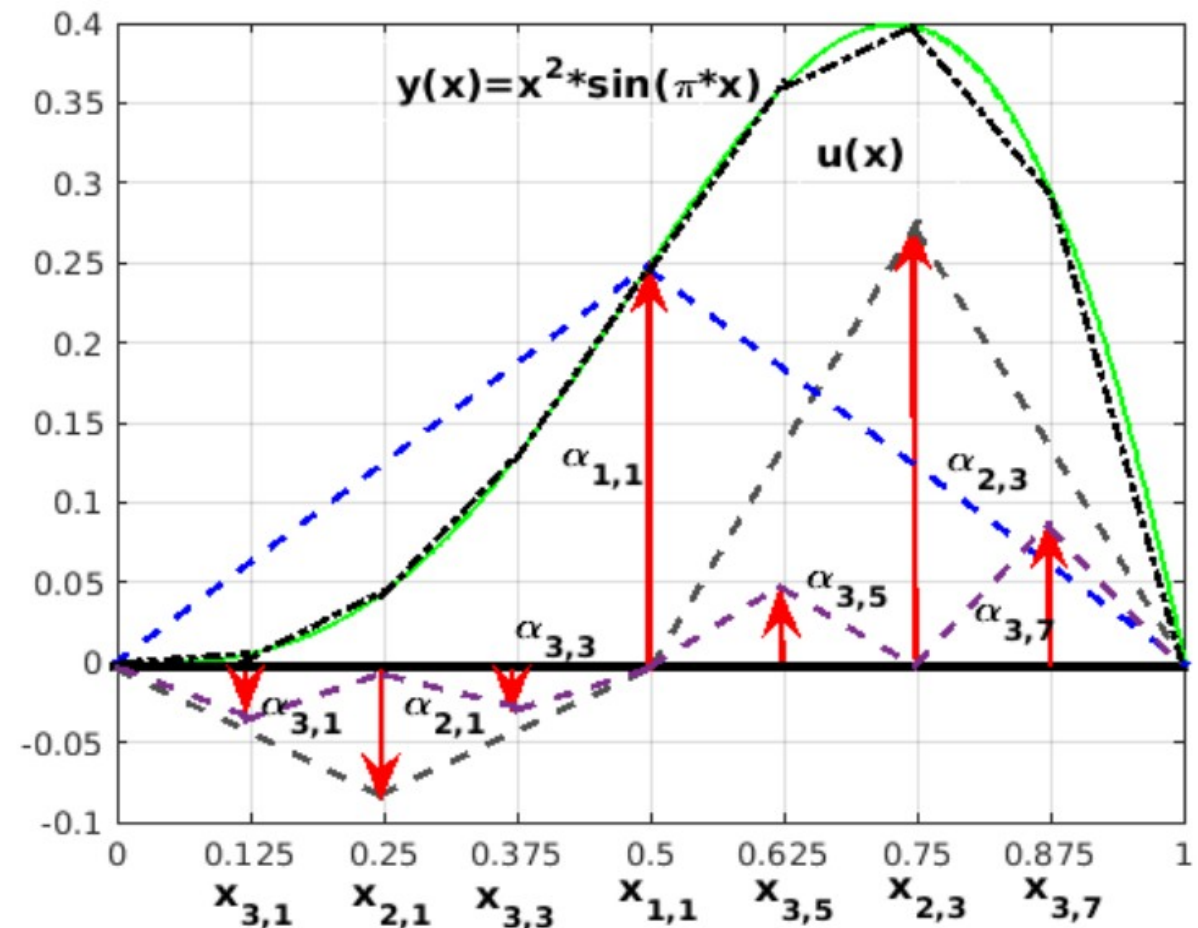
the  $\neq$  between  
green and blue lines

# Piecewise Linear Interpolation: Level III

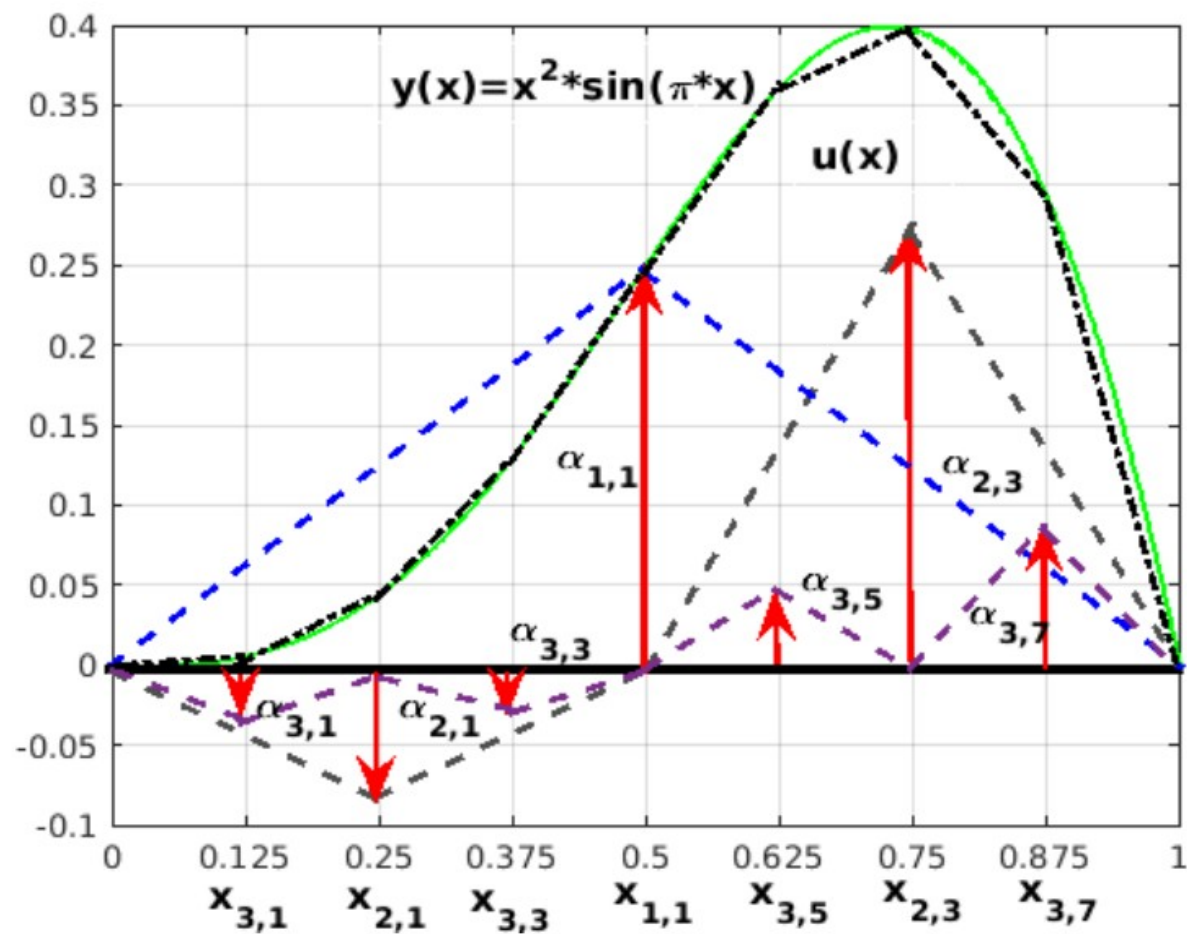
Coefficients:  
**hierarchical surpluses**

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

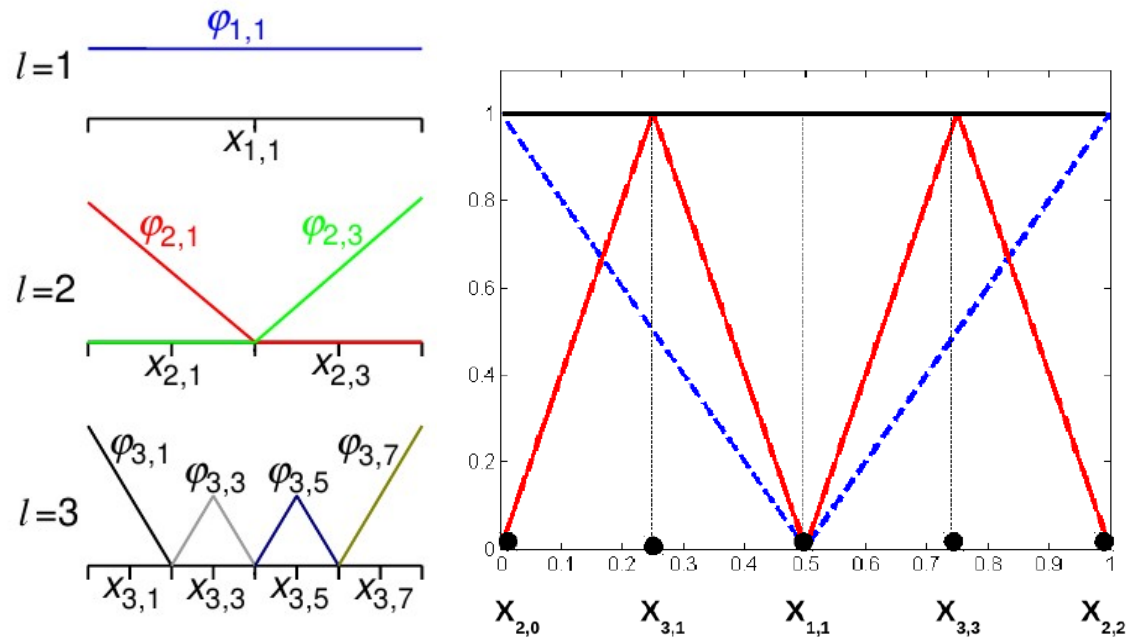
Want to be able to handle non-zero boundaries:

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

If we add naively points at boundaries, **3<sup>d</sup>** 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

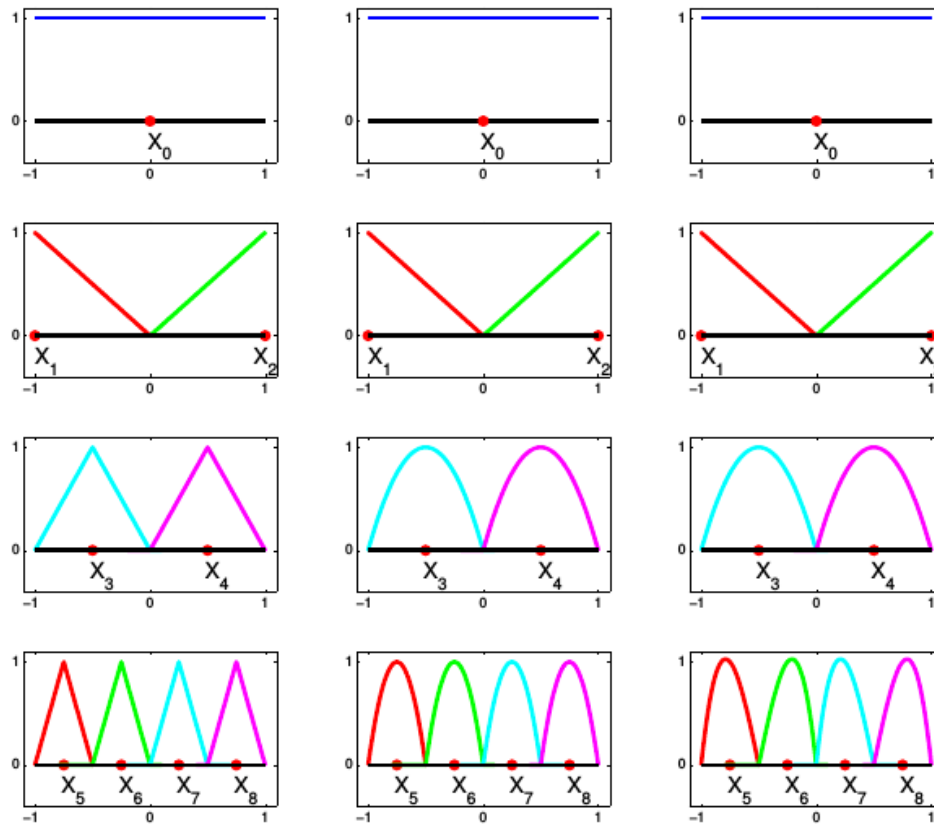


Figure 1: Local polynomial points (*rule\_localp*) and functions, left to right: linear, quadratic, and cubic functions.

# 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:**  $\vec{l} = (l_1, \dots, l_d) \in \mathbb{N}^d$

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

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

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

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



# Multi-Dimensional Interpolant

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

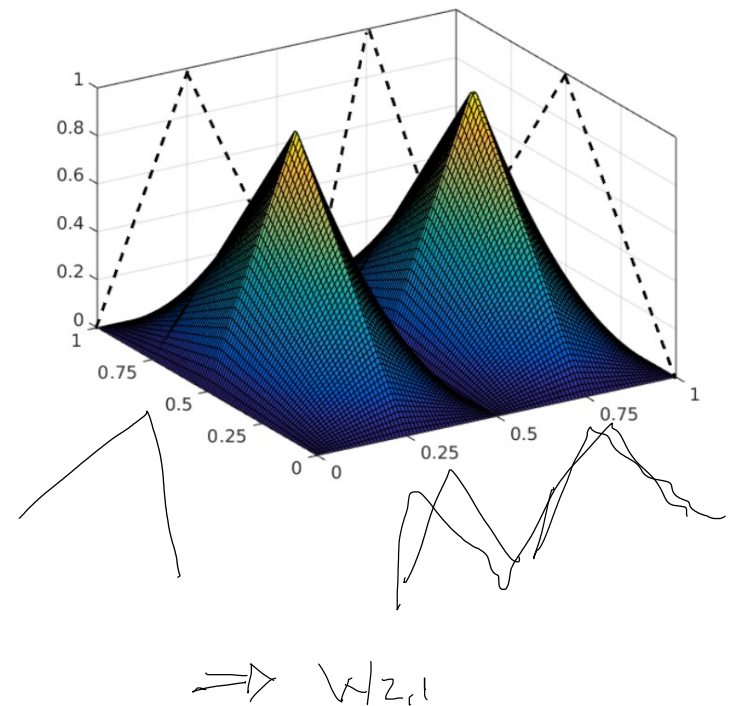
Multi-d basis:  $\phi_{\vec{l}, \vec{i}}(\vec{x}) := \prod_{t=1}^d \phi_{l_t, i_t}(x_t)$

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

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

Multi-d interpolant:

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



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

## Why reality bites...

Interpolant consists of  $\mathcal{O}(2^{nd})$  grid points

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

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

function evaluations  $\rightarrow$  **“curse of dimensionality”**

Hard to handle more than 4 dimensions numerically

$\rightarrow$  e.g.  $d=10$ ,  $n = 4$ , 15 points/d,  **$5.8 \times 10^{11}$**  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  $n$ .

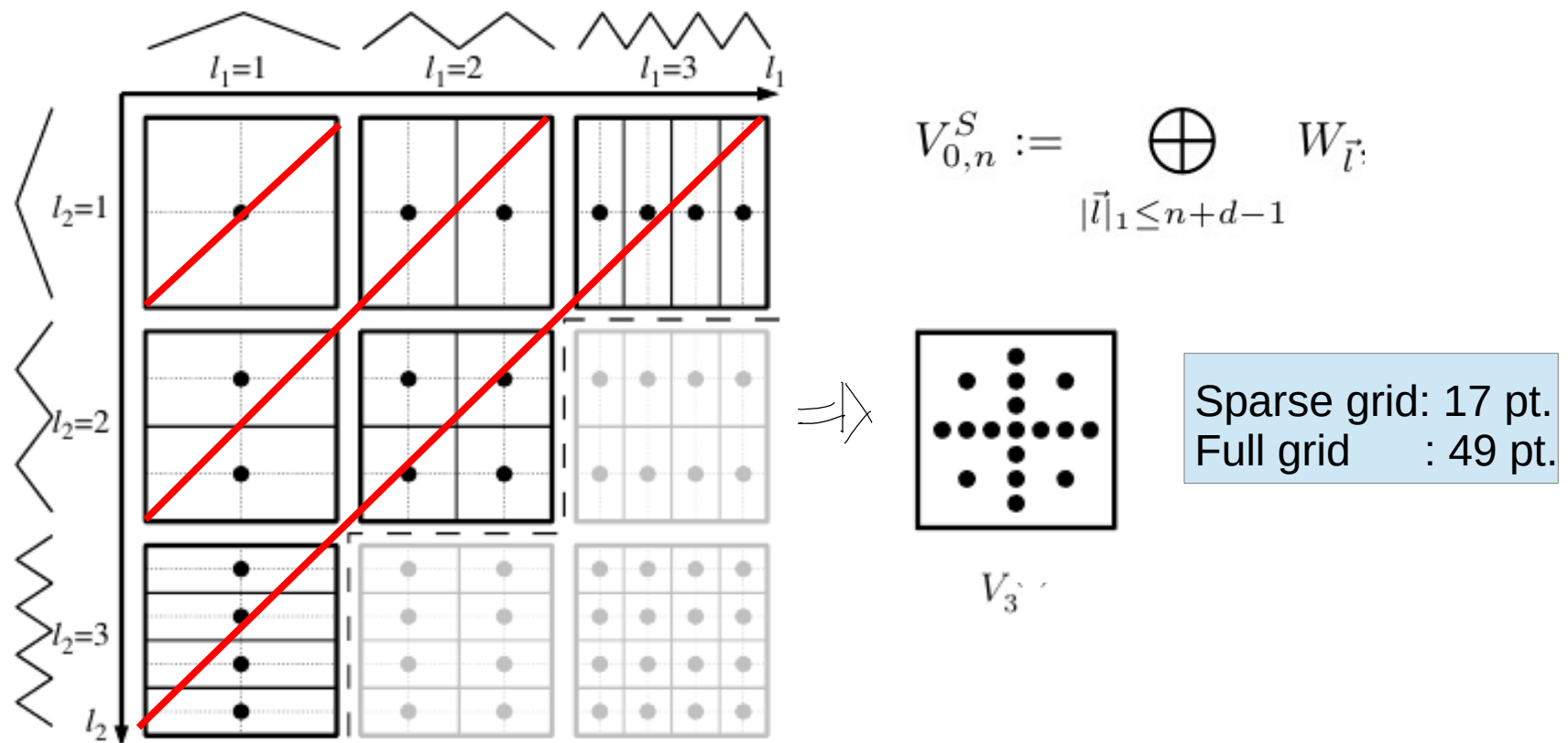
$$V_{0,n}^S := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}$$

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

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

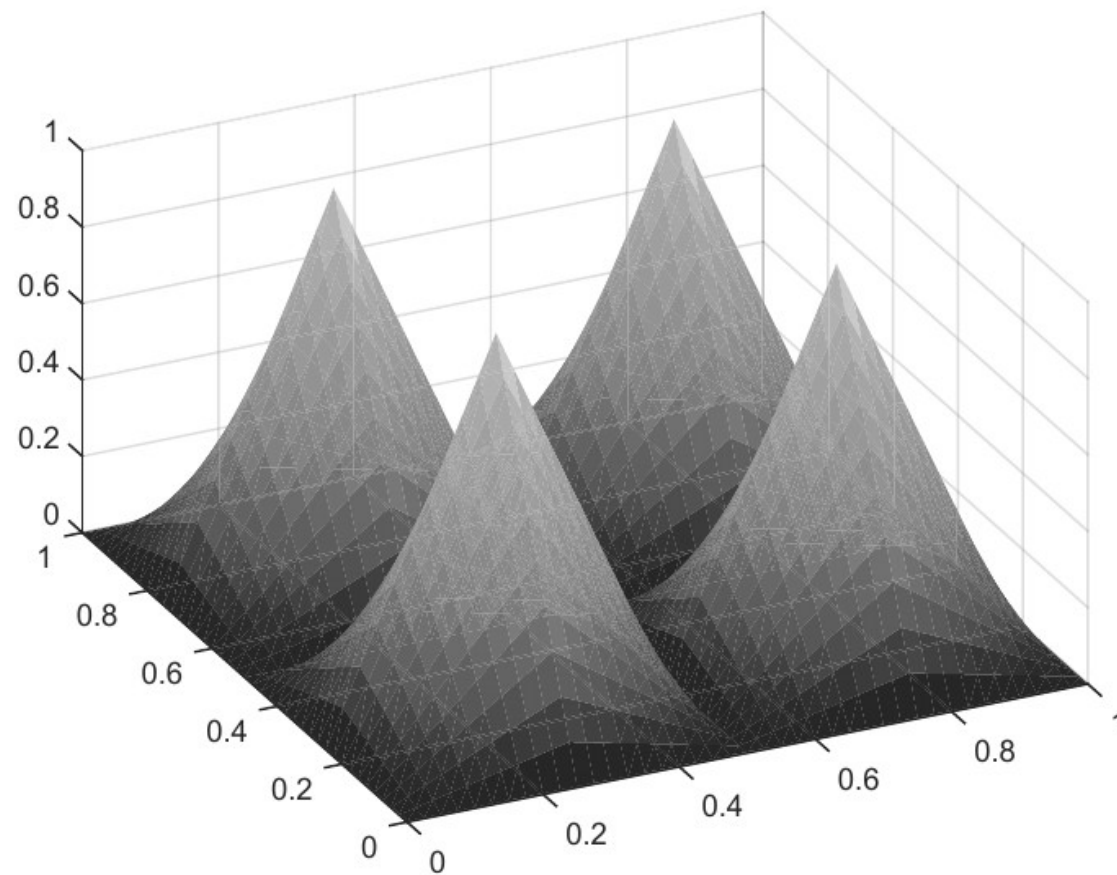
Accuracy of the interpolant:  $\mathcal{O}(h_n^2 \cdot \log(h_n^{-1})^{d-1})$  vs.  $\mathcal{O}(h_n^2)$

# Sparse grid construction in 2D

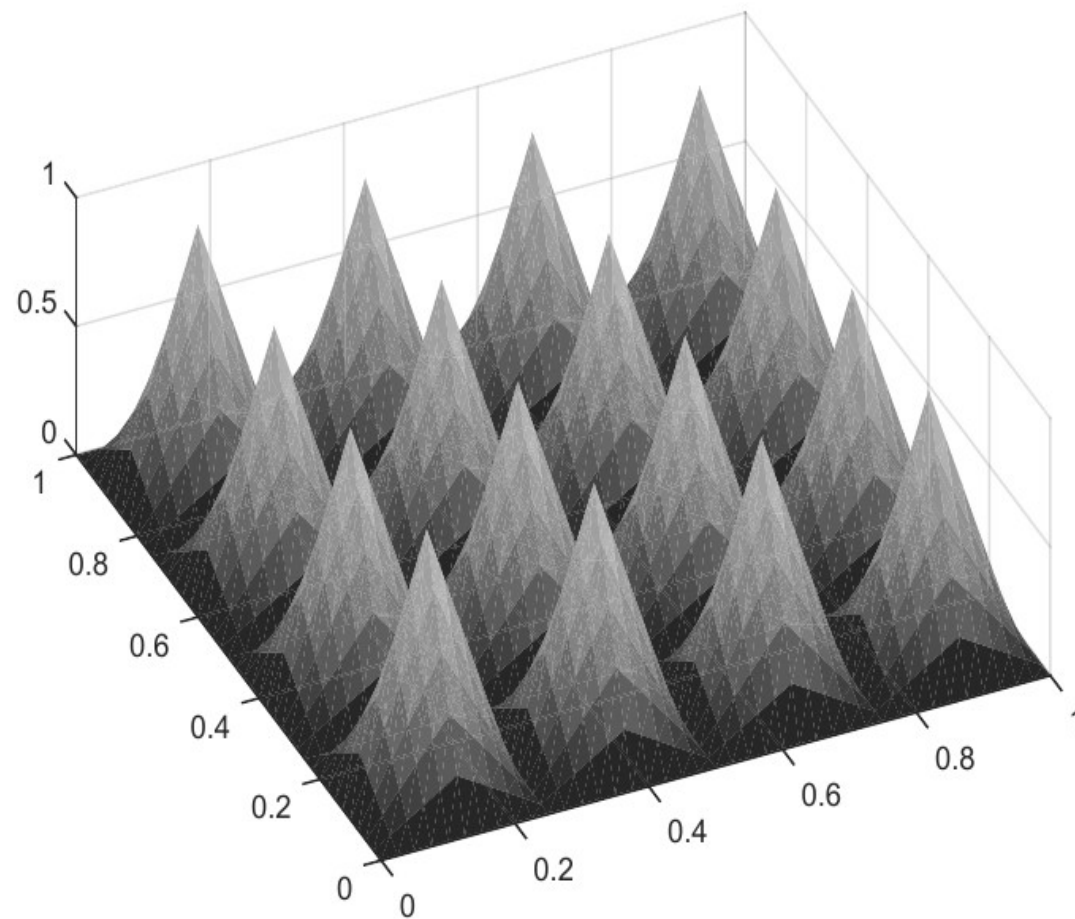


**Fig.:** Two-dimensional subspaces  $W_{\vec{l}}$  up to  $l=3$  ( $h_3 = 1/8$ ) in each dimension. The **optimal a priori selection of subspaces** is shown in black (**left**) and the corresponding sparse grid of level  $n = 3$  (**right**). For the **full grid**, the gray subspaces have to be used as well.

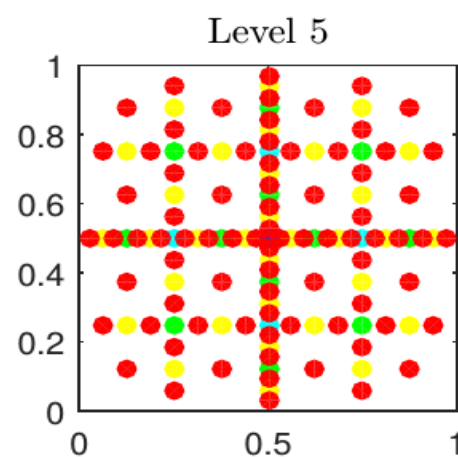
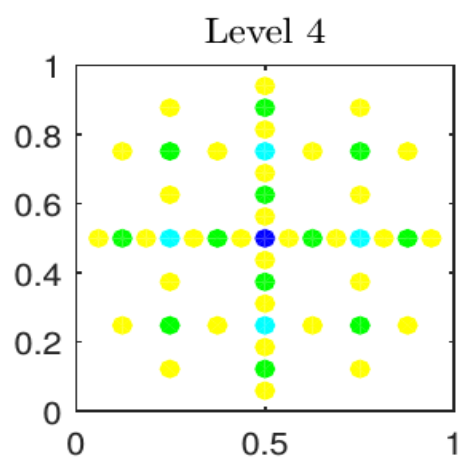
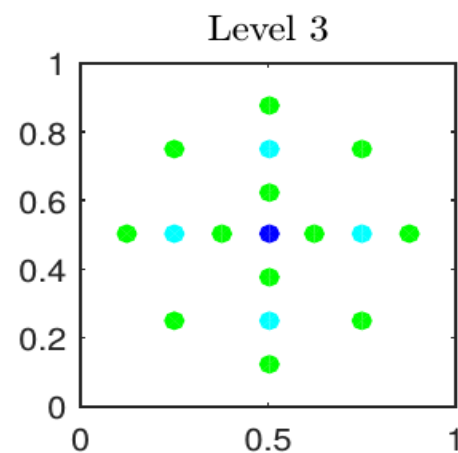
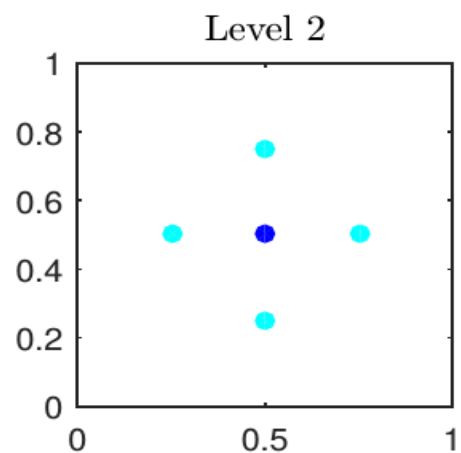
# Basis Functions of $W_{2,2}$ — Included in $V_3$



# Basis Functions of $W_{3,3}$ — not Included in $V_3$



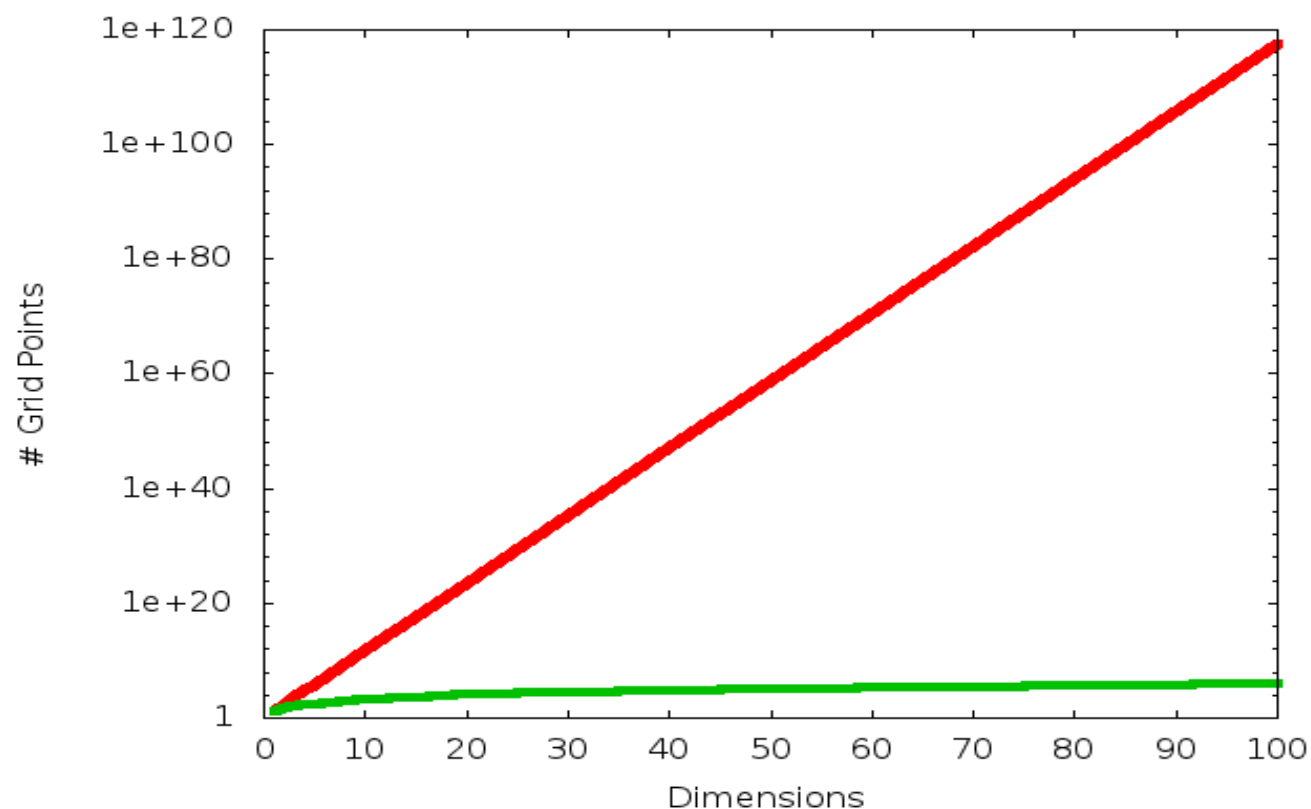
# Sparse Grid of Increasing level





# 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     |



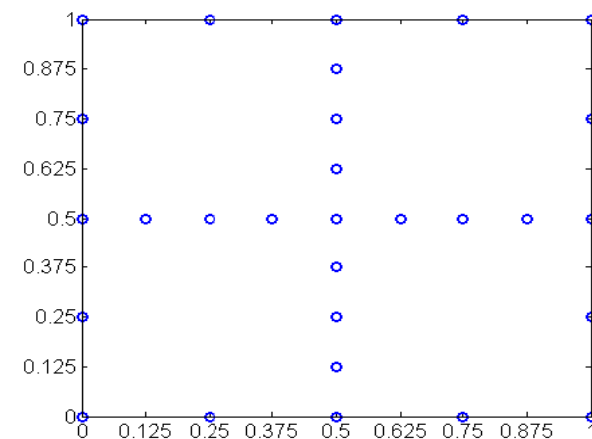
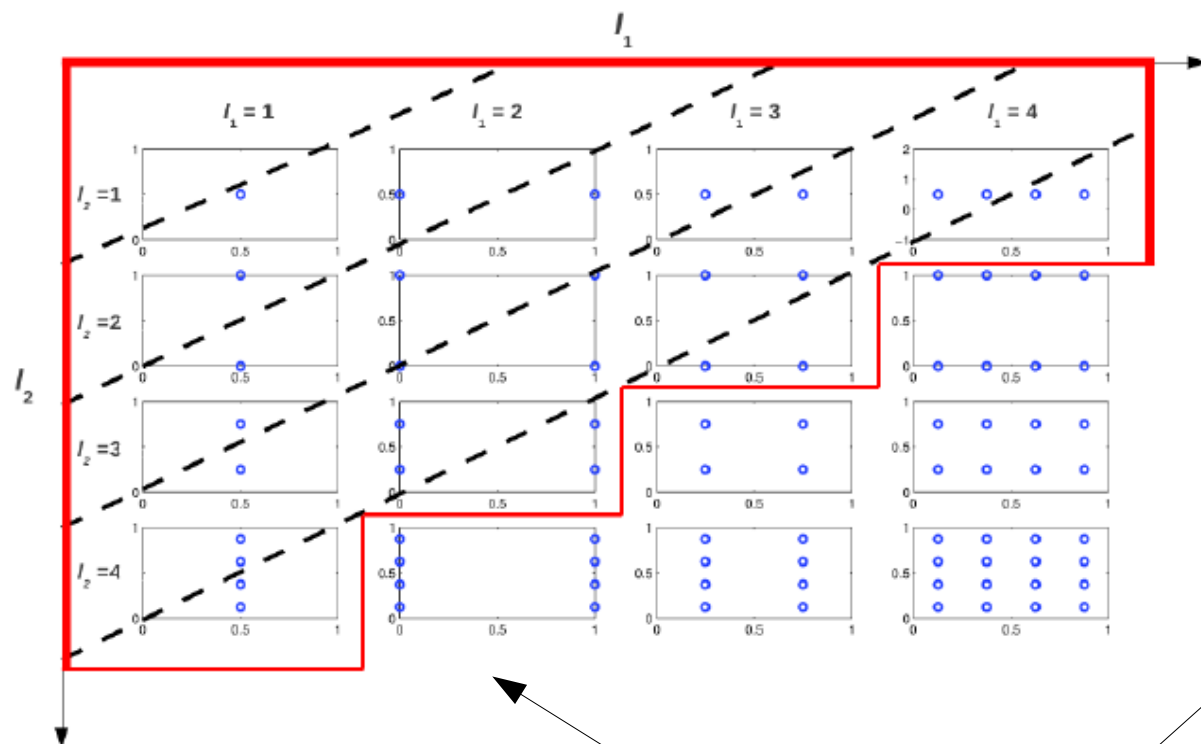
**Tab.:** Number of grid points for several types of sparse grids of level  $n = 4$ .

**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))



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

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

# Hierarchical Integration

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

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

The one-dimensional integral can now be computed analytically (Ma & Zabaras (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  $\int_{\Omega} \phi_{l,i}(\vec{x}) d\vec{x} = J_{\vec{l}, \vec{i}}$

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

# 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 & Wohlmuth (2005),...

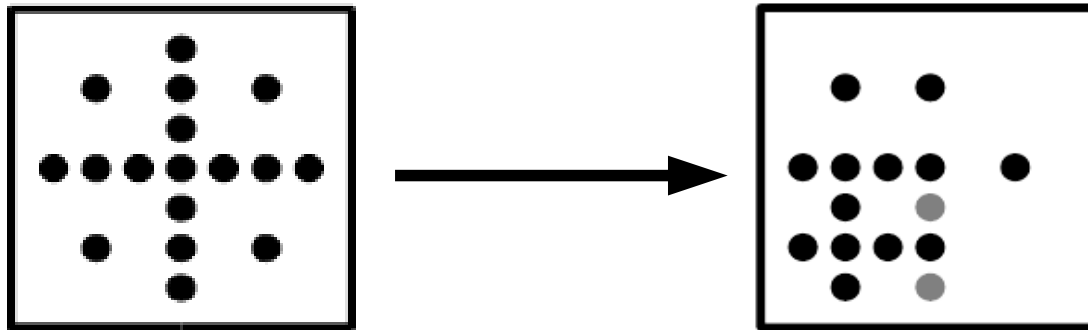
- Solution of PDEs

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 & Zabarar (2008), Pflüger (2010), Bungartz (2003),...

-Surpluses should quickly decay to zero

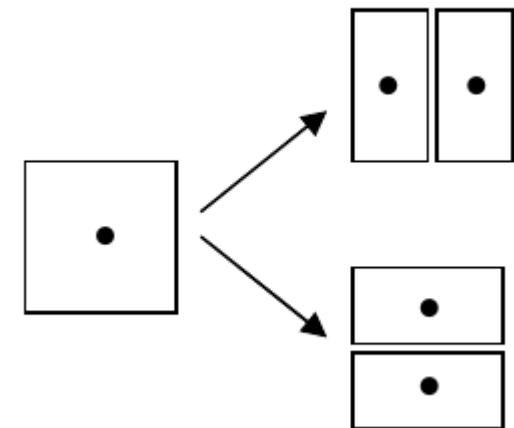
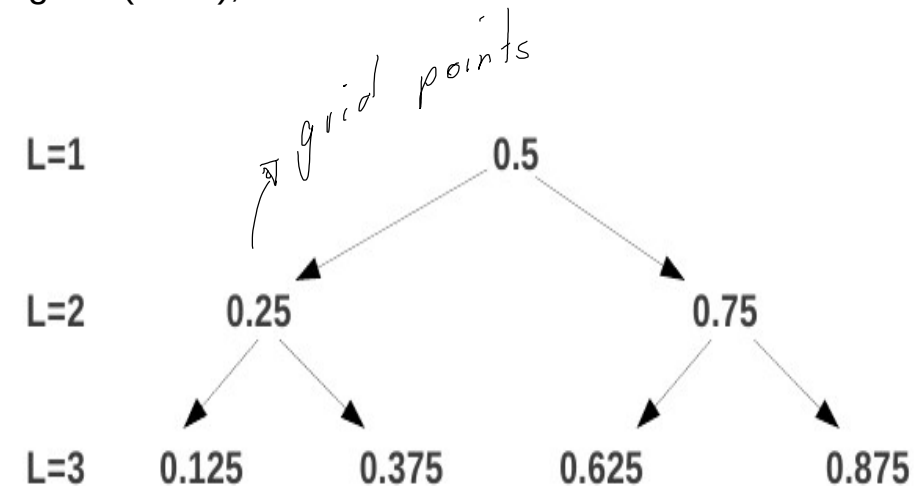
**-Use hierarchical surplus as error indicator.**

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

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

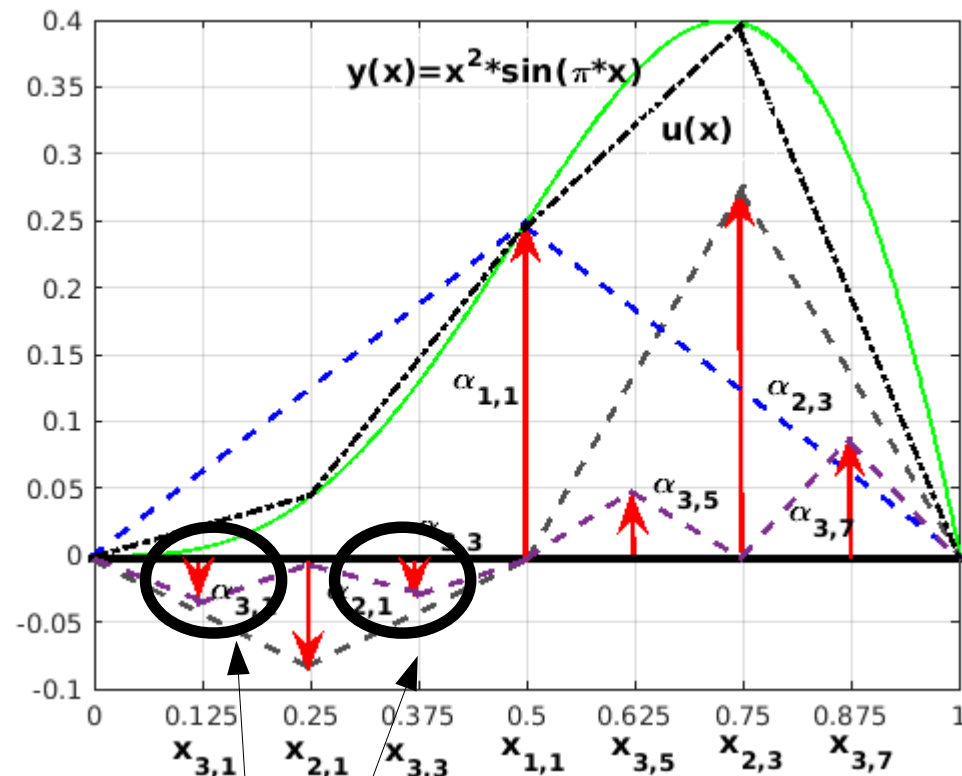
**-Add neighbour points, i.e. locally refine interpolation level from  $l$  to  $l+1$**

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



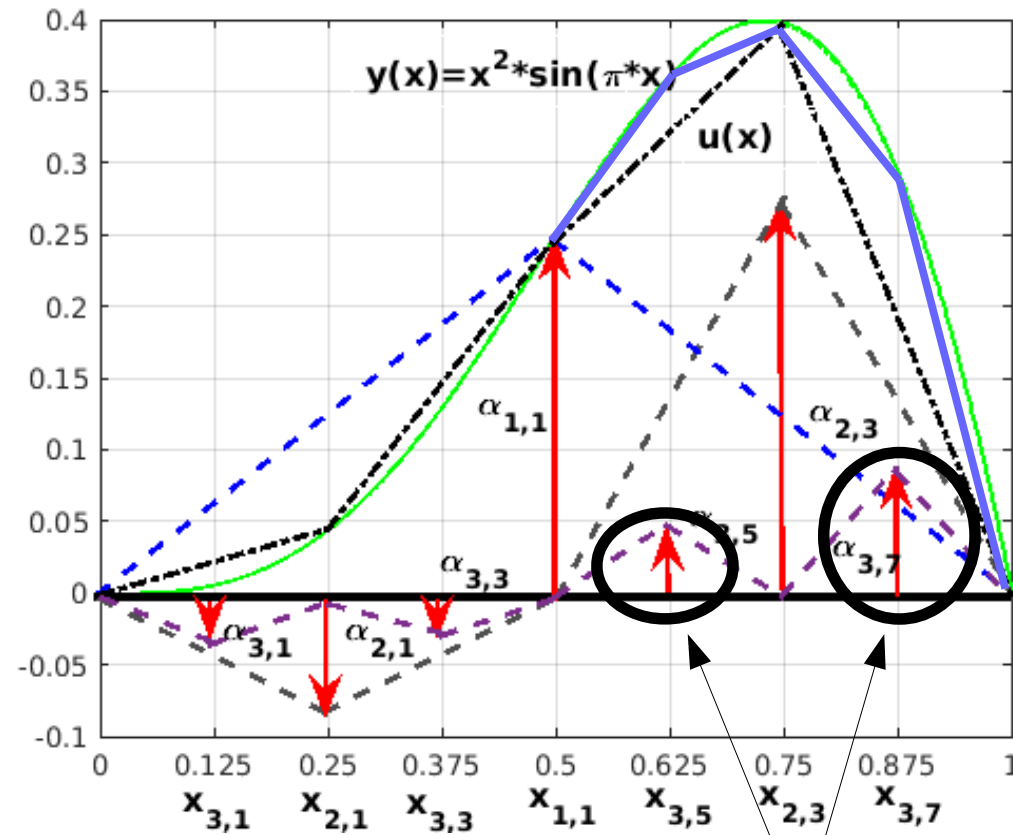
**top panel:** tree-like structure of sparse grid.  
**lower panel:** locally refined sparse grid in 2D.

# 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  $[0,1]$

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

Full grid: **1023** points

Adaptive sparse grid: **109** points.

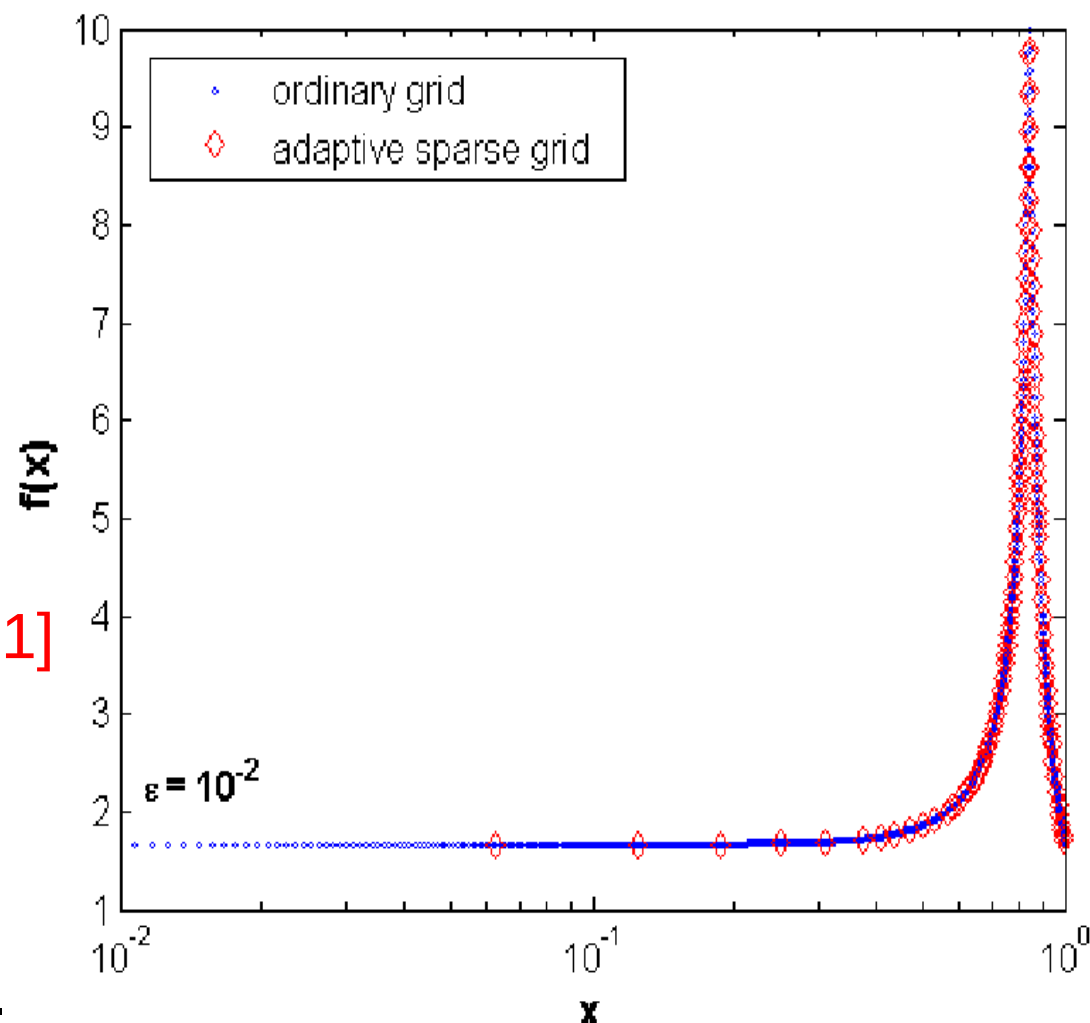


Fig.: Blue: Full grid; red: adaptive sparse grid.

# Test in 2d

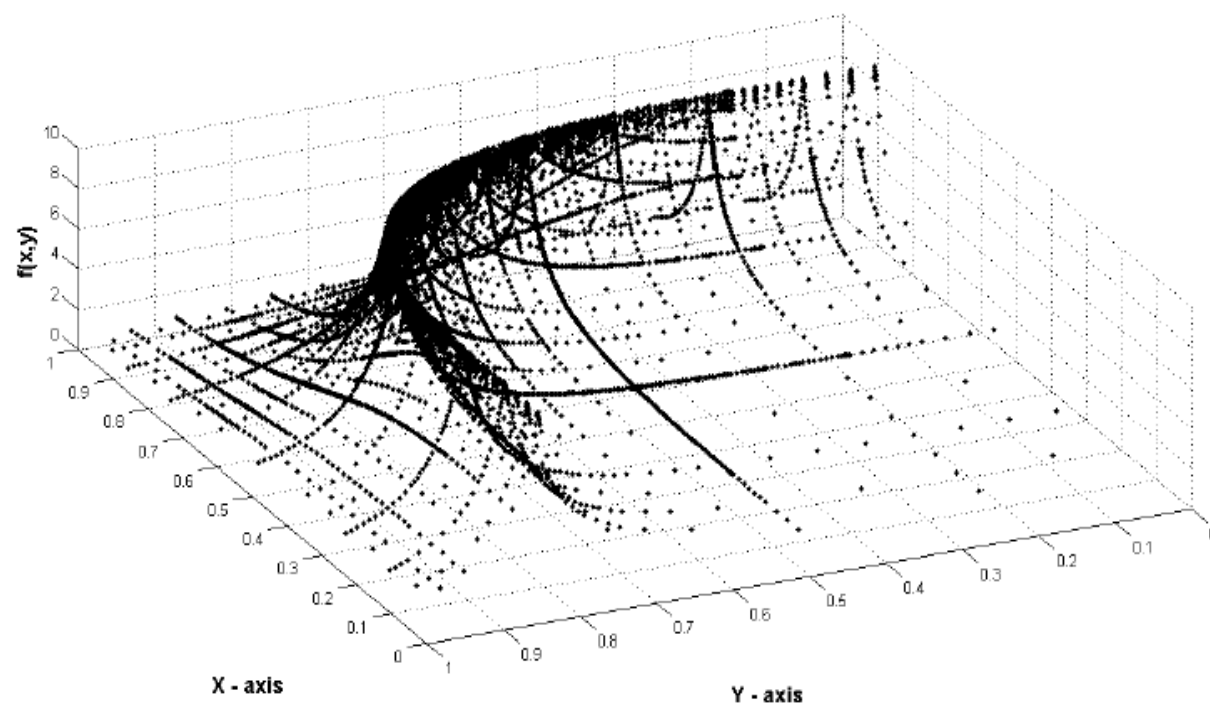
Test function:  $\frac{1}{|0.5 - x^4 - y^4| + 0.1}$

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

Full grid:  
→  $O(10^9)$  points

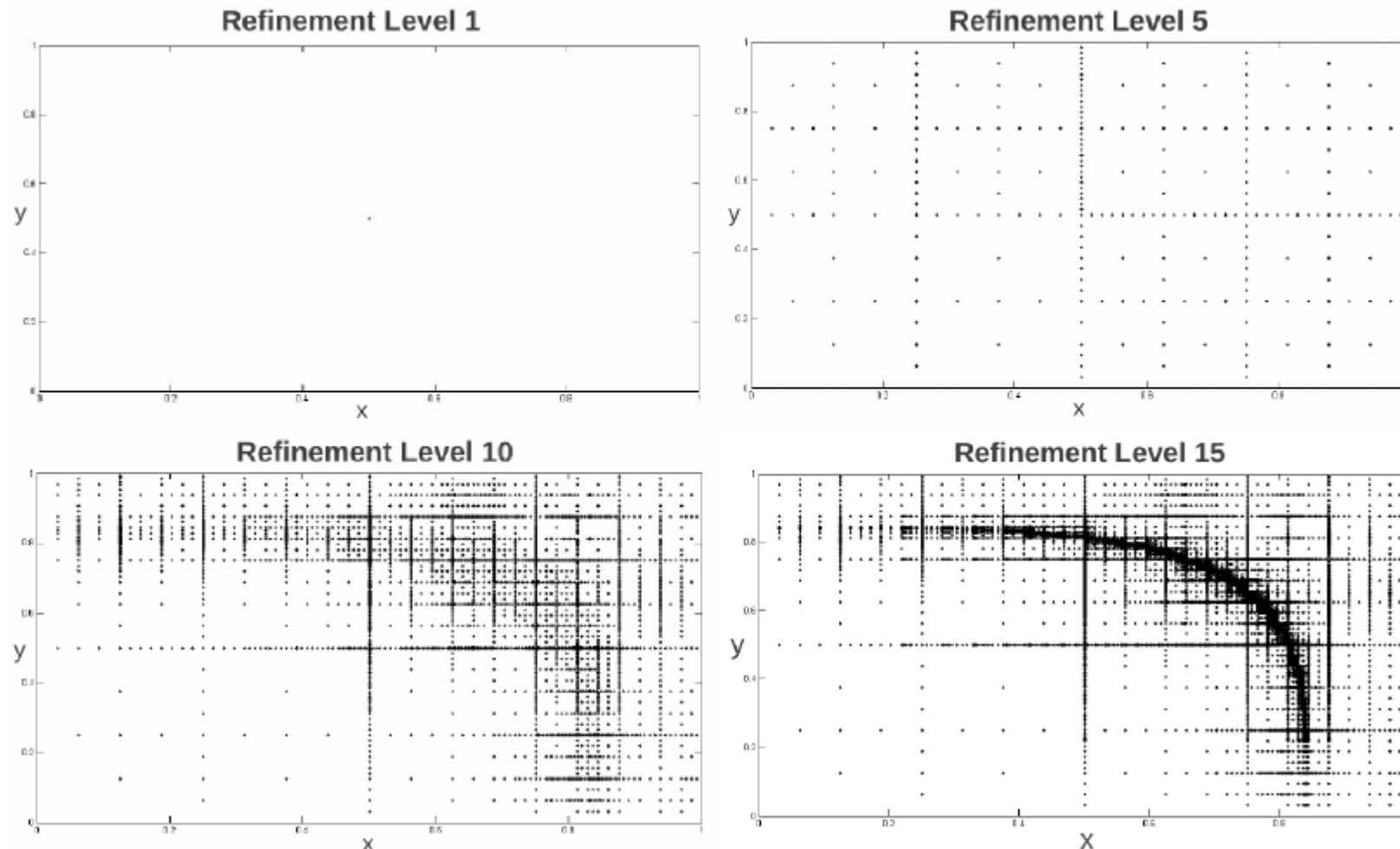
Sparse grid:  
→ **311,297 points**

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



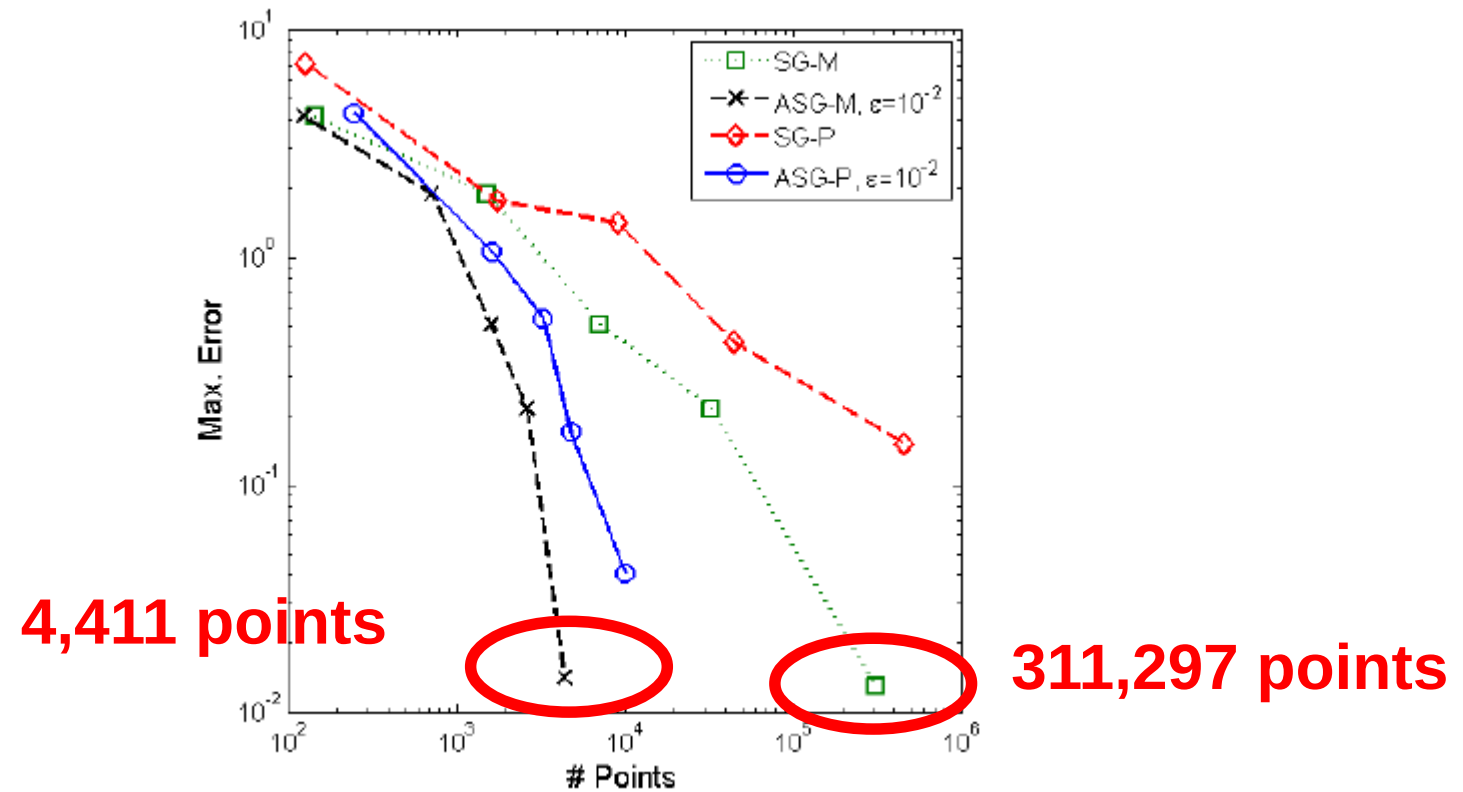
**Fig.:** 2d test function and its corresponding grid points after 15 refinement steps.

# Movie



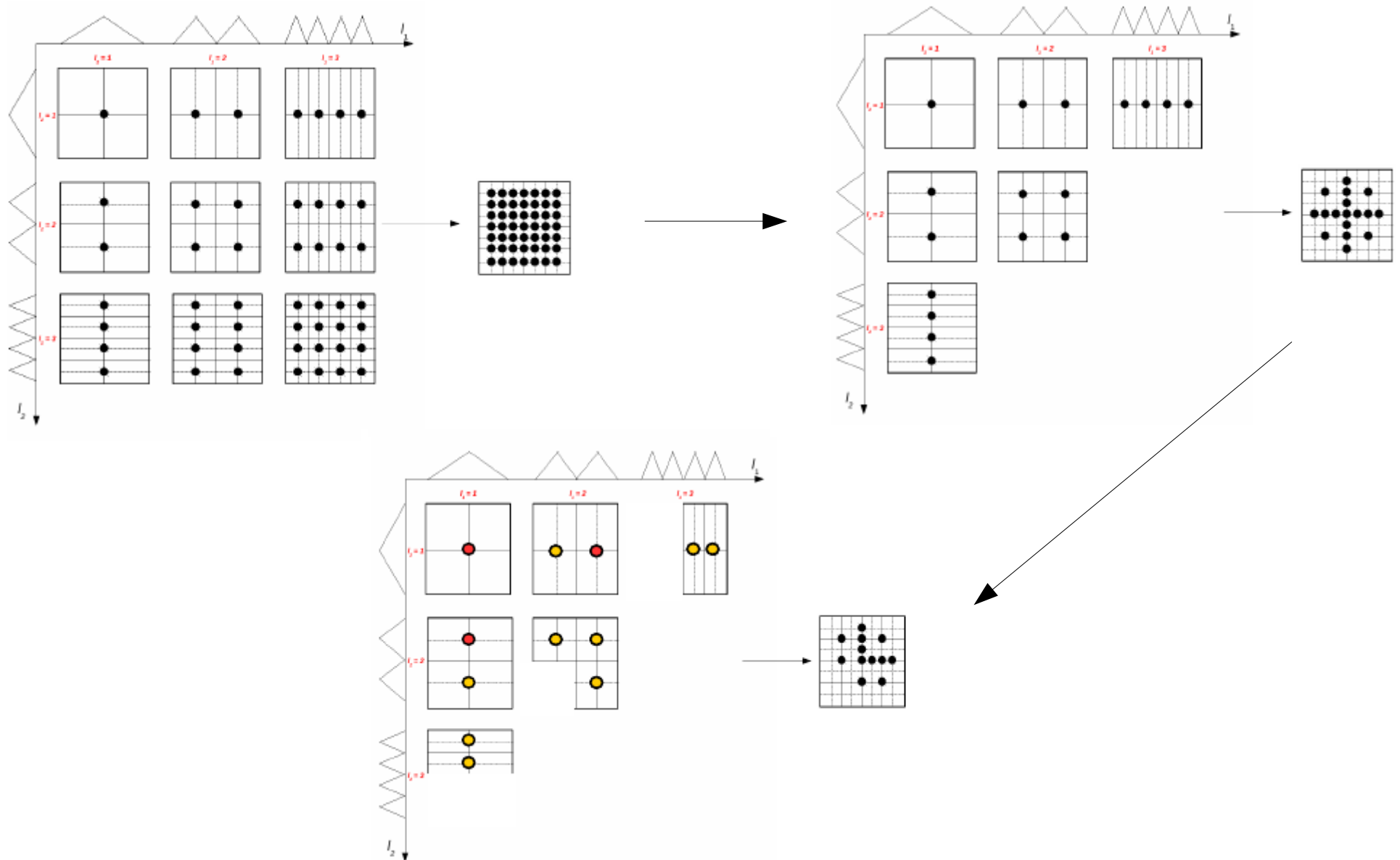
**Fig.:** Evolution of the adaptive sparse grid with a **threshold for refinement of  $10^{-2}$** . The refinement levels displayed are  $L = 1, 5, 10, 15$ .

# Convergence



**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



# 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

# 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

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

$$x_{t+1} \sim F(\cdot | g, p(g)),$$

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

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

N Euler equations

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

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

1 aggregate resource constraint

$(a_t^1, \dots, a_t^N, k_t^1, \dots, k_t^N)$   $\longrightarrow$  **Current grid points**

solve for the unknown policy variables

$(k_{t+1}^1, \dots, k_{t+1}^N, \lambda_t)$   $\longrightarrow$  **Solution of system of Eqs.  
At the current iteration**

given the known policy functions from last iteration

$(k_{t+2}^1(a_{t+1}, k_{t+1}), \dots, k_{t+2}^N(a_{t+1}, k_{t+1}), \lambda_{t+1}(a_{t+1}, k_{t+1}))$   $\longrightarrow$  **Interpolant**

evaluated at next periods state

$(a_{t+1}^1, \dots, a_{t+1}^N, k_{t+1}^1, \dots, k_{t+1}^N)$ , 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:

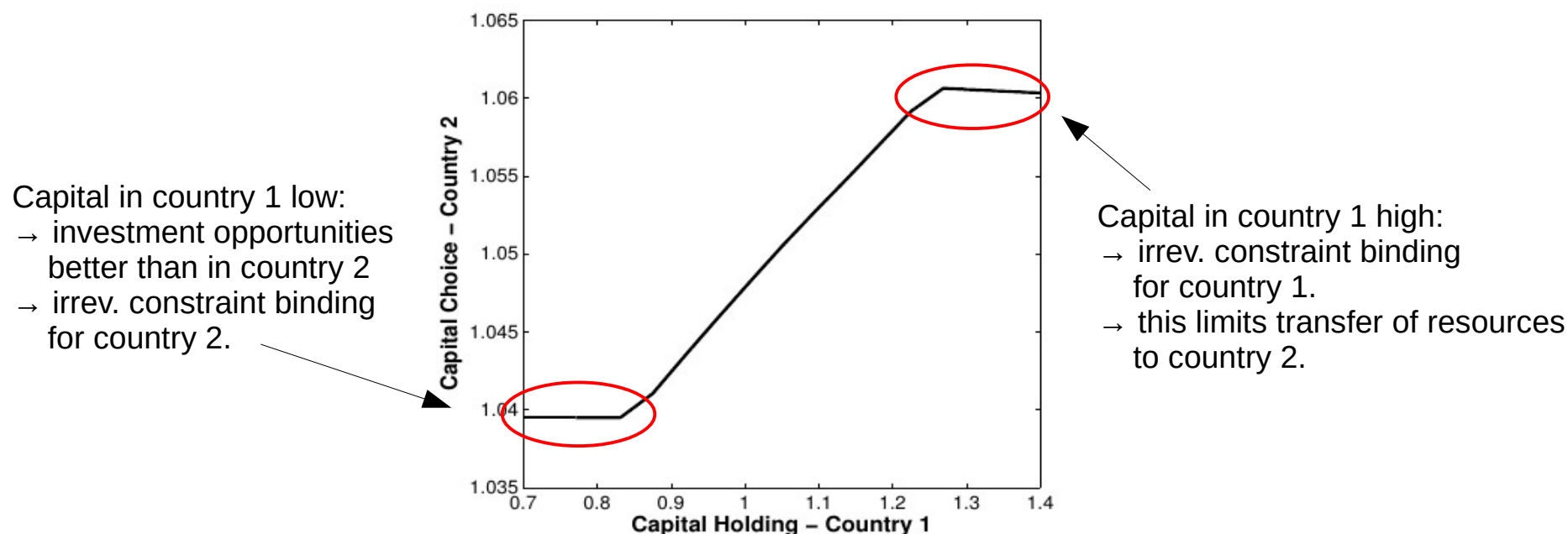
| Dimension | Level | Points | Max. Error | Avg. Error |
|-----------|-------|--------|------------|------------|
| 4         | 5     | 401    | -2.11      | -2.93      |
| 4         | 7     | 2,929  | -2.32      | -3.12      |
| 4         | 9     | 18,945 | -2.45      | -3.32      |

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

| $\epsilon$ | 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.

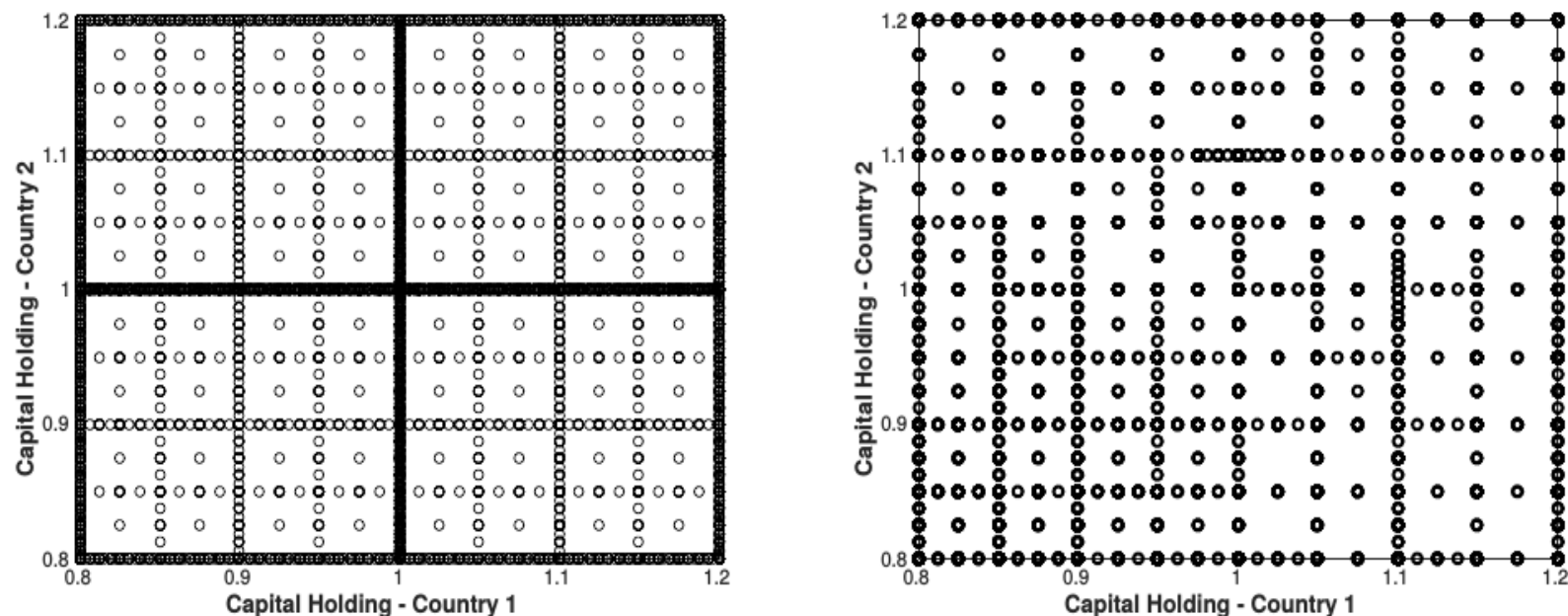
# IRBC with irreversible investment



**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 ( $\varepsilon = 0.0033$ ).

**Note: kink is  $(2N - 1)$  - dimensional hypersurface in  $2N$  - dim state space.**

# IRBC with binding constraints



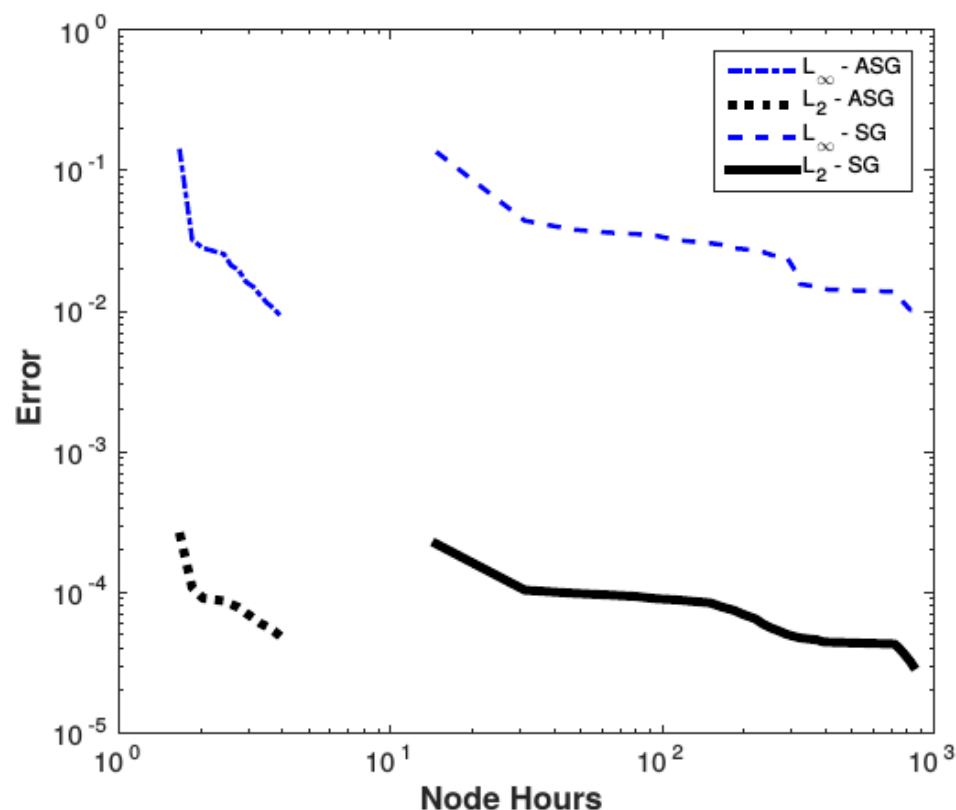
**Fig.:** 2-d projections of two different grids.

**Left:** 'classical' sparse grid of level 9 (18,945 points),

**Right:** adaptive grid with refinement threshold  $\varepsilon = 0.001$  (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 constraints: massive speedup due to adaptivity



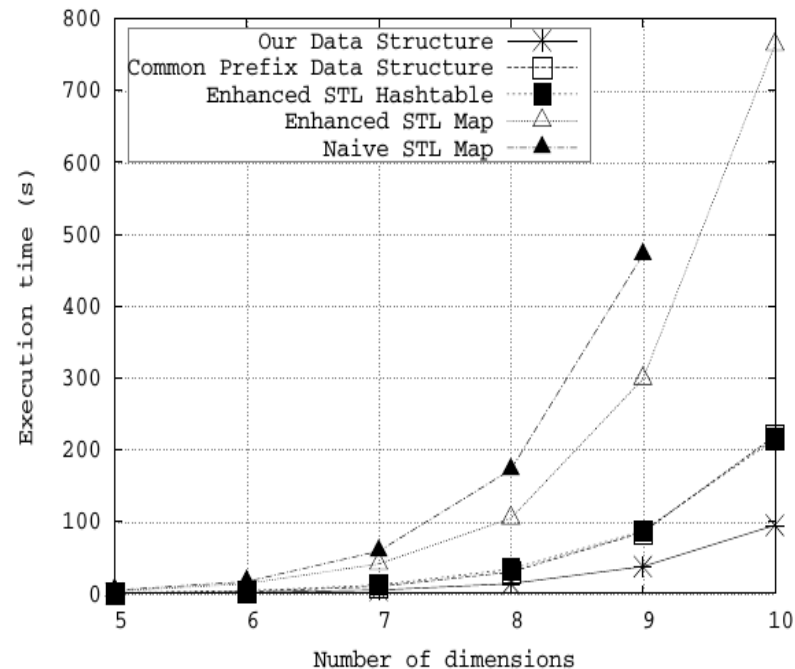
**Fig.:** 8d model with binding constraints.  
model run with/without adaptive sparse grids.  
Relative error among two consecutive time-steps.  
10k points drawn from uniform distribution.

| 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      | $\gg 10^8$     |

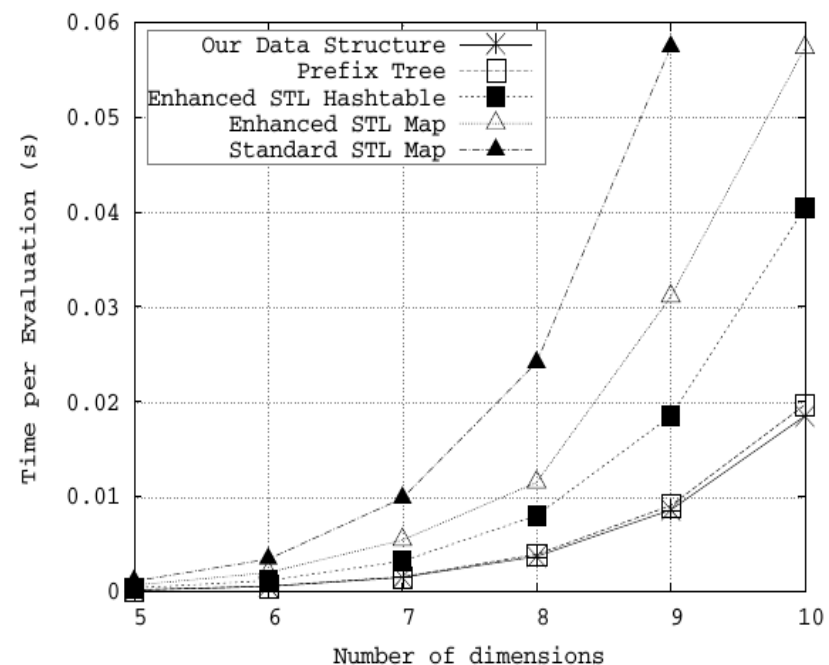
**Tab.:** Comparison of a sparse and adapt. sparse grid of comparable accuracy.

# Limitation of sparse grids:

## Execution times in higher dimension



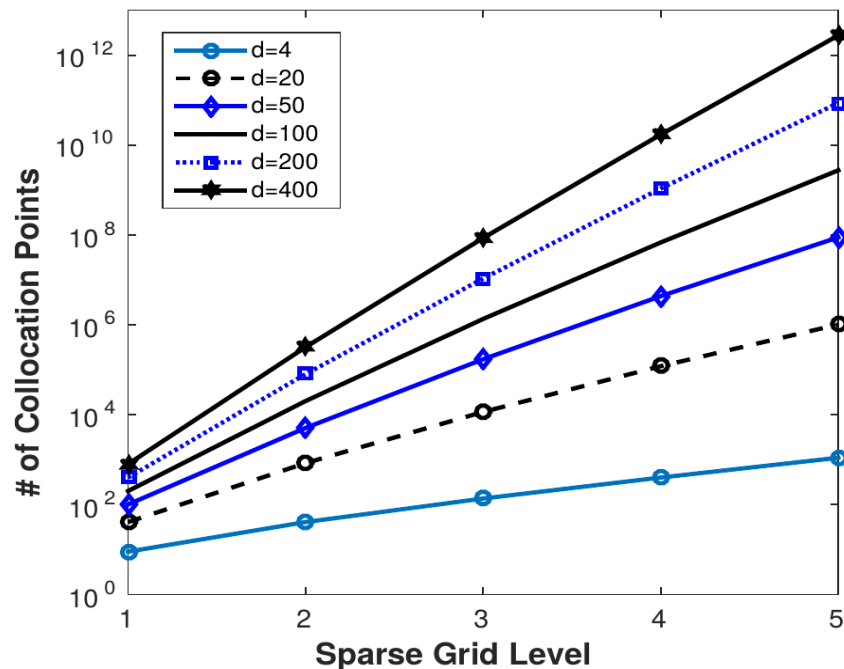
(a) Runtime for sequential hierarchization.



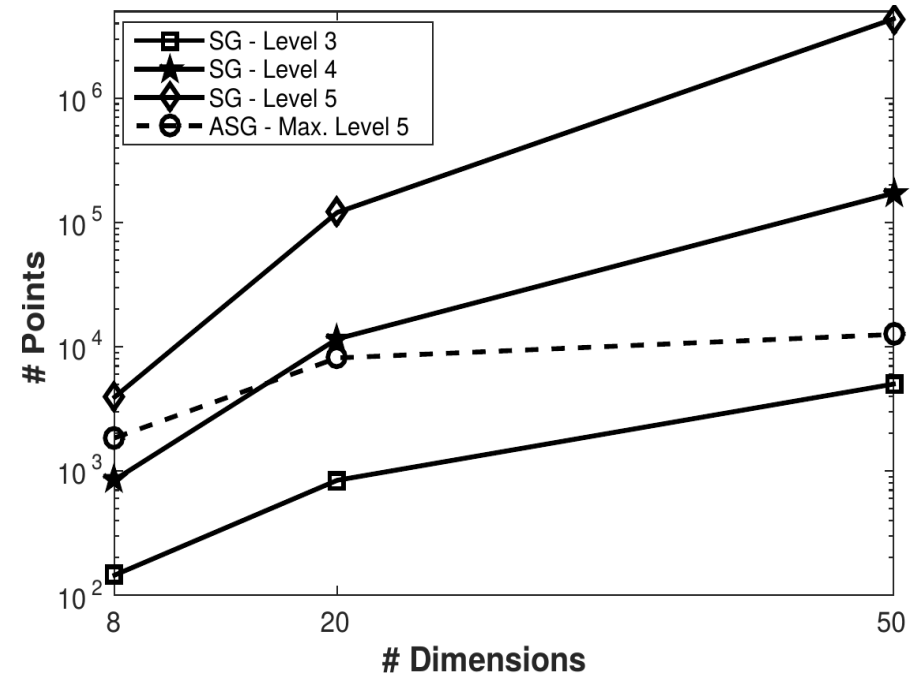
(b) Runtime for sequential evaluation.

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

# 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 extent **(right panel)**.

# How to build a star killer for economic problems?





