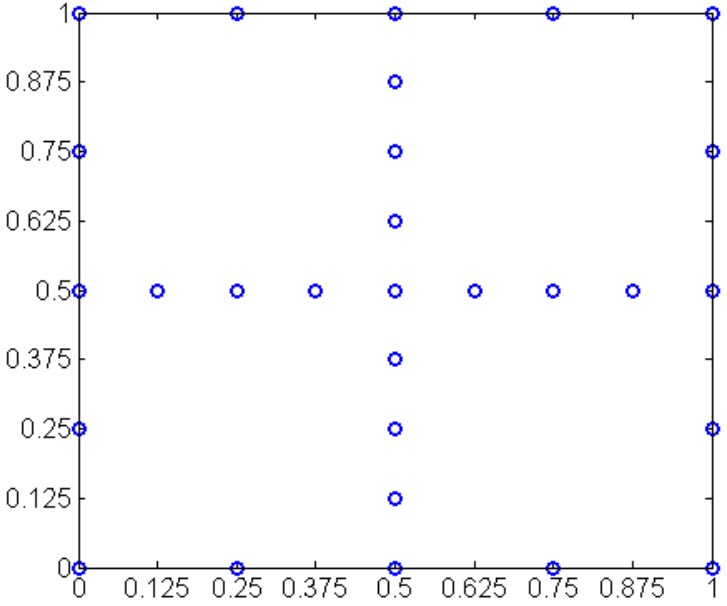
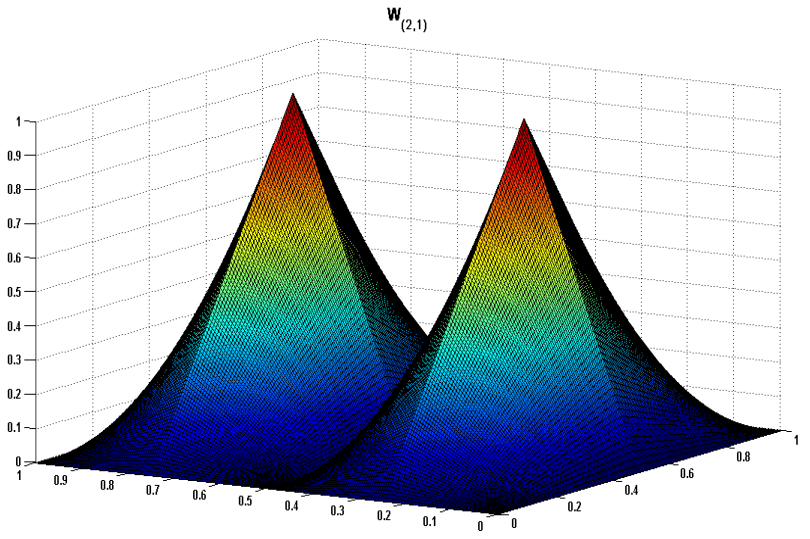
Universität Zürich[UZH]

Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models

J. Brumm & S. Scheidegger

~~`Geek'~~ - 'Nerd' weeks, Hoover Institution, Stanford





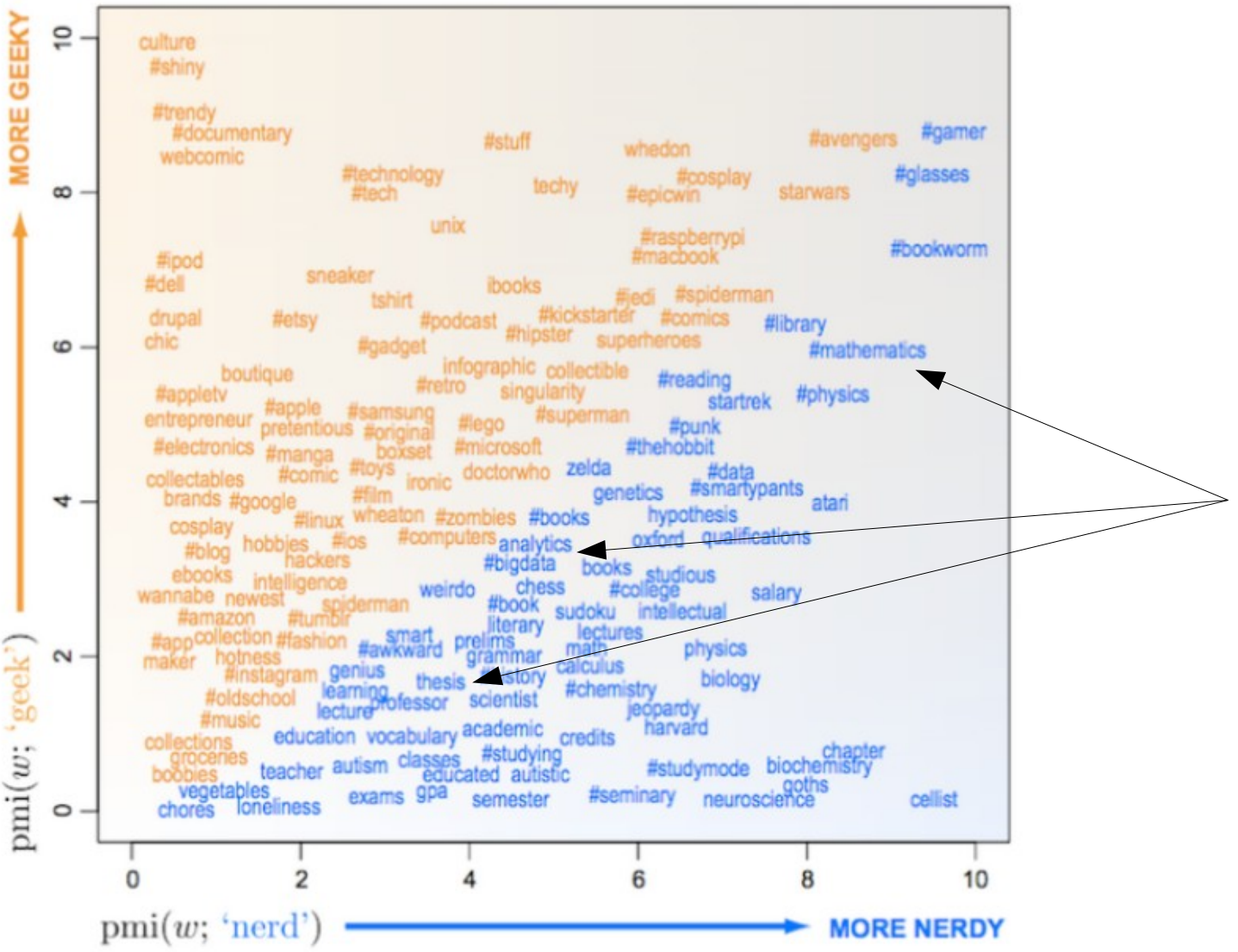
`Geek weeks' – or `nerd weeks'?

geek - An enthusiast of a particular topic or field. Geeks are "collection" oriented, gathering facts and mementos related to their subject of interest.
They are obsessed with the newest, coolest, trendiest things that their subject has to offer.

nerd - A studious intellectual, although again of a particular topic or field. Nerds are "achievement" oriented, and focus their efforts on acquiring knowledge and skill over trivia and memorabilia.

Source: http://slackprop.wordpress.com/2013/06/03/on-geek-versus-nerd/

`Geek' → `Nerd weeks'



Source: Settles, B. *On "Geek" Versus "Nerd."* June 13, 2013
http://slackprop.wordpress.com/2013/06/03/on-geek-versus-nerd/

Outline

I) From Full Grids to Sparse Grids

II) Adaptive Sparse Grids

III) Time Iteration & Adaptive Sparse Grids & HPC

→ Implementation, testing

Where are Sparse grids used?

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

Sparse grid methods date back to Smolyak(1963)

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, DSGE models in Economics...
e.g. Kubler & Kruger (2004), Winschel & Kraetzig (2010)

Our motivation

i) Want to compute high-dimensional time iteration problems e.g. Int. real business cycle model:

(see, e.g. Wouter et al. (2011), Malin et al. (2011))

$$\text{“}Tv = v\text{”} \longleftrightarrow |Tv_i - v_{i+1}| < \varepsilon$$

Problem: curse of dimensionality

⟶ N^d points in ordinary discretization schemes

ii) Want to overcome curse of dimensionality

iii) **Want locality & adaptivity of interpolation scheme** (ability to handle singularities, kinks,...)

iv) Technically: **access HPC systems** (MPI, OpenMP)

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

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

- Discrete, full grid $\Omega_{\vec{l}}$ on Ω with mesh size

$$h_{\vec{l}} := (h_{l_1}, ..., h_{l_d}) = 2^{-\vec{l}} := \left(2^{-l_1}, ..., 2^{-l_d}\right)$$

- 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 $i_t \in \{0, 1, ..., 2^{l_t}\}$

Interpolation on a Full Grid I

-Consider a **d-dimensional function** $f : \Omega \to \mathbb{R}$

-In numerical simulations:
f might be given only algorithmically – expensive to evaluate!
But: need to be able to evaluate *f* at arbitrary points using a numeircal code

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

-With suitable (e.g. linear) basis functions $\varphi_i(\vec{x})$
and coefficients α_i

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

Interpolation on a Full Grid II

-Sparse grids depend on a **hierarchical decomposition** of the underlying approximation space.

-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**

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

Interpolation on a Full Grid III

Hierarchical increment spaces:

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

with the **index set**

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

The corresponding function space:

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

The **1d-interpolant:**

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

Note: all basis functions of W_k mutually disjoint!

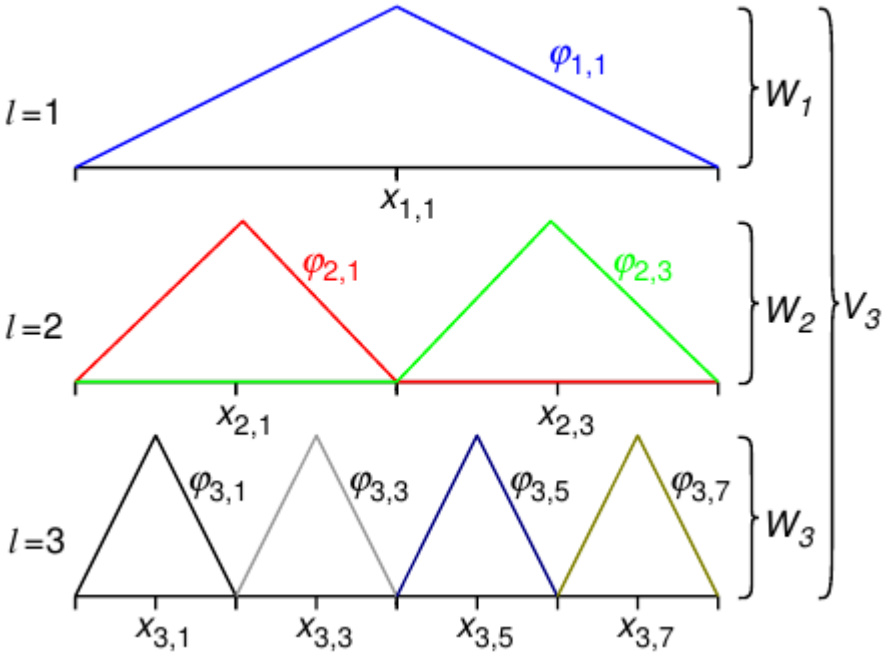


Fig.: **1-d basis functions** $\phi_{l,i}$ and the corresponding **grid points** up level ***l* = 3** in the hierarchical basis.

Interpolation on a Full Grid

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

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

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

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

Multi-d interpolant:

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

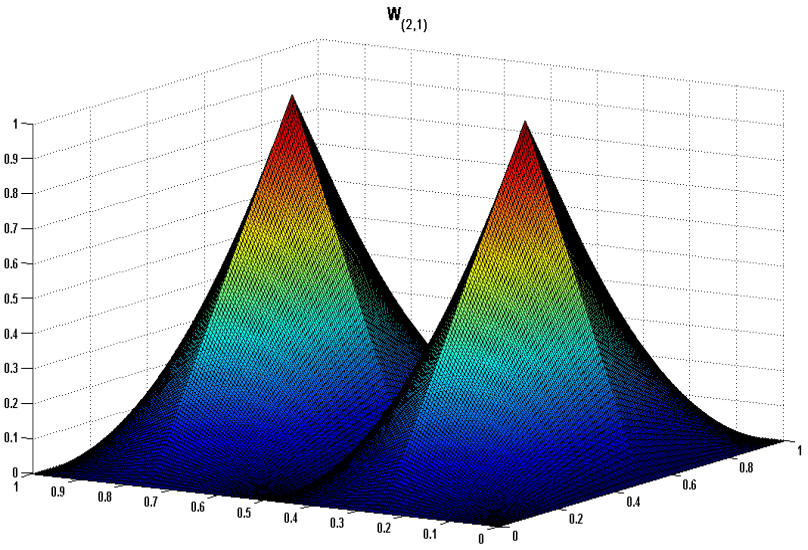


Fig.: Basis functions of the **subspace $W_{2,1}$**

The curse of dimensionality or: why reality bites...

Interpolant consists of $(2^n - 1)^d$ grid points

For **sufficiently smooth *f*** and its interpolant ***u***, we obtain an asymptotic error decay of

$$\| f(\vec{x}) - u(\vec{x}) \|_{L_2} \in \mathcal{O}\left(h_n^2\right) \quad \text{where} \quad \|f\|_{L_2} := \left(\int_\Omega |f(\vec{x})|^2 d\vec{x} \right)^{1/2}$$

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}** degrees of freedom

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 √**

Under certain smoothness conditions – the second mixed derivatives have to be bounded, this can be done a priori.

$$D^{\vec{l}} f := \frac{\partial^{|\vec{l}|_1}}{\partial x_1^{l_1} \cdots \partial x_d^{l_d}} f \quad \text{where} \quad |\vec{l}|_1 := \sum_{t=1}^{d} l_t$$

These functions belong to a so-called Sobolev space

$$H_2^{\text{mix}}(\Omega) := \{ f : \Omega \to \mathbb{R} : D^{\vec{l}} f \in L_2(\Omega), |\vec{l}|_\infty \leq 2, f|_{\partial\Omega} = 0 \} \text{ where } |\vec{l}|_\infty := \max_{1 \leq t \leq d} l_t$$

Under this prerequisite, the hierarchical coefficients rapidly decay

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

Breaking the curse of dimensionality II

-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***.

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

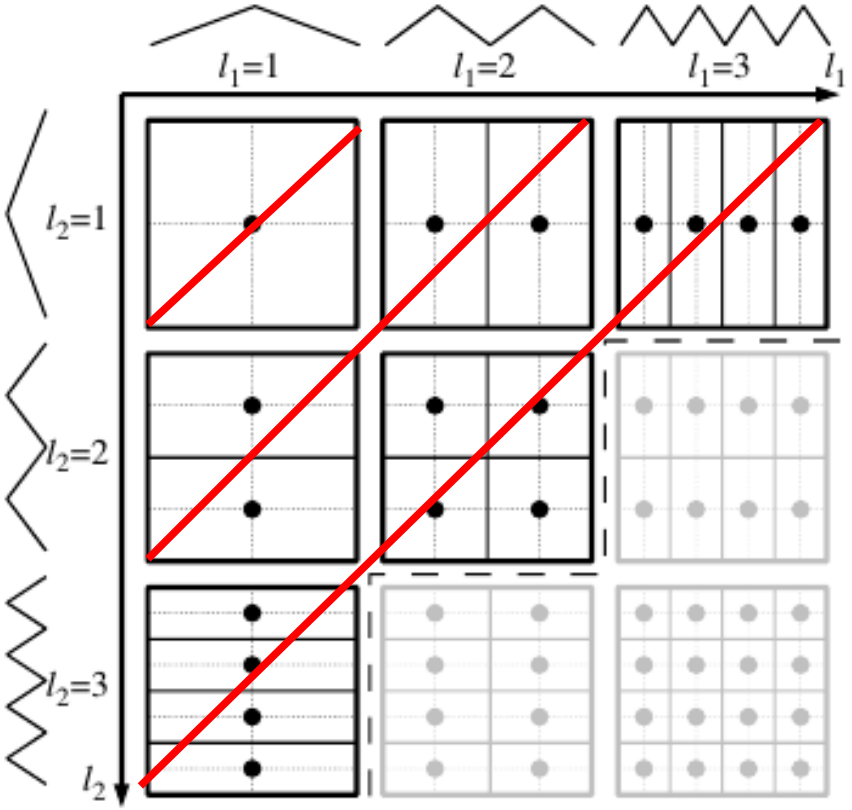
Note: This result is optimal for the L_2- Norm and the $L_\infty-$ Norm.

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

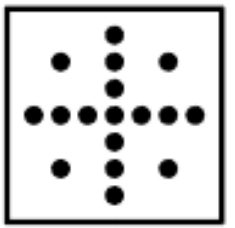
grid points: $\mathcal{O}\left(h_n^{-1} \cdot \left(\log(h_n^{-1})\right)^{d-1}\right) = \mathcal{O}\left(2^n \cdot n^{d-1}\right)$ << $\mathcal{O}\left(h_n^{-d}\right) = \mathcal{O}\left(2^{nd}\right)$

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

Sparse grid construction in 2D



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



$V_3^{(1)}$

Fig.: Two-dimensional subspaces $\mathbf{W_l}$ up to $\boldsymbol{l}$=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.

Degrees of Freedom

d	$\lvert V_n \rvert$	$\lvert V_n^S \rvert$	$\lvert V_n^{S,CC} \rvert$
1	15	15	9
2	225	49	29
3	3375	111	69
4	50'625	209	137
5	759'375	351	241
10	$5.77 \cdot 10^{11}$	2'001	1'581
15	$4.37 \cdot 10^{17}$	5'951	5'021
20	$3.33 \cdot 10^{23}$	13'201	11'561
30	$1.92 \cdot 10^{35}$	41'601	37'941
40	$1.11 \cdot 10^{47}$	95'201	88'721
50	$6.38 \cdot 10^{58}$	182'001	171'901
100	>Googol	1'394'001	1'353'801

Tab.: Number of grid points for several types of sparse grids of level $n = 4$.
Left: Full grid; **middle: classical sparse grid** with **no points at the boundaries**;
right column: Clenshaw-Curtis grid.

Hierarchical surplus – nested structure

Coefficients of interpolant termed **“hierarchical surpluses”**.

Can easily be determined due to **nested property** of the hierarchical grid.

$$\alpha_{l,i} = f(x_{l,i}) - \frac{f(x_{l,i} - h_l) + f(x_{l,i} + h_l)}{2}$$

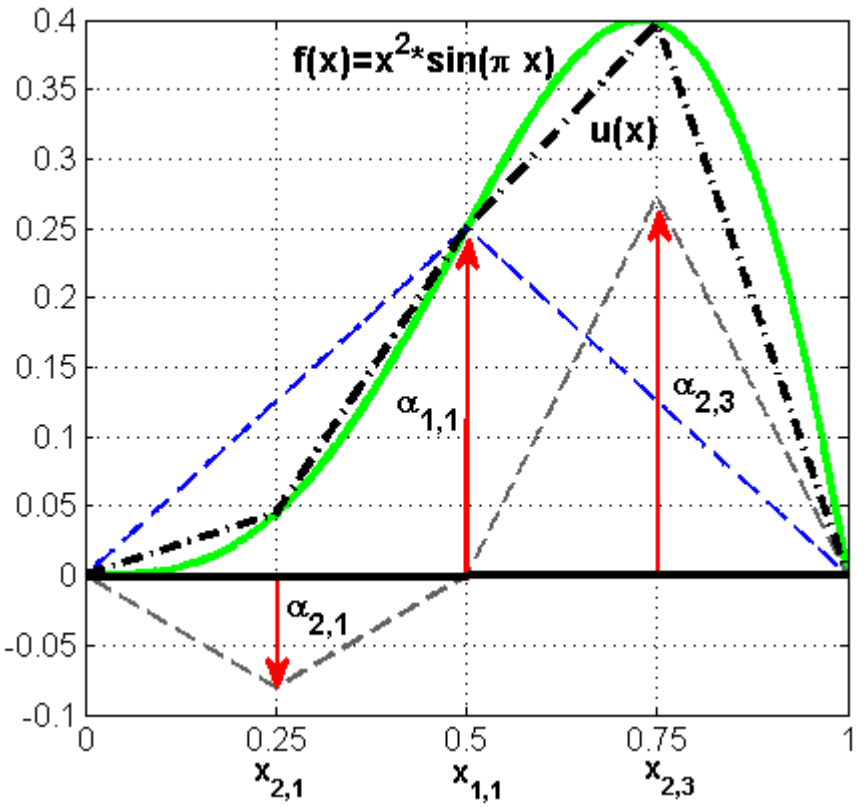


Fig.: Interpolant u of f of level 2.

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: go from one level or refinement to the next

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

Treatment of non-zero boundaries

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!

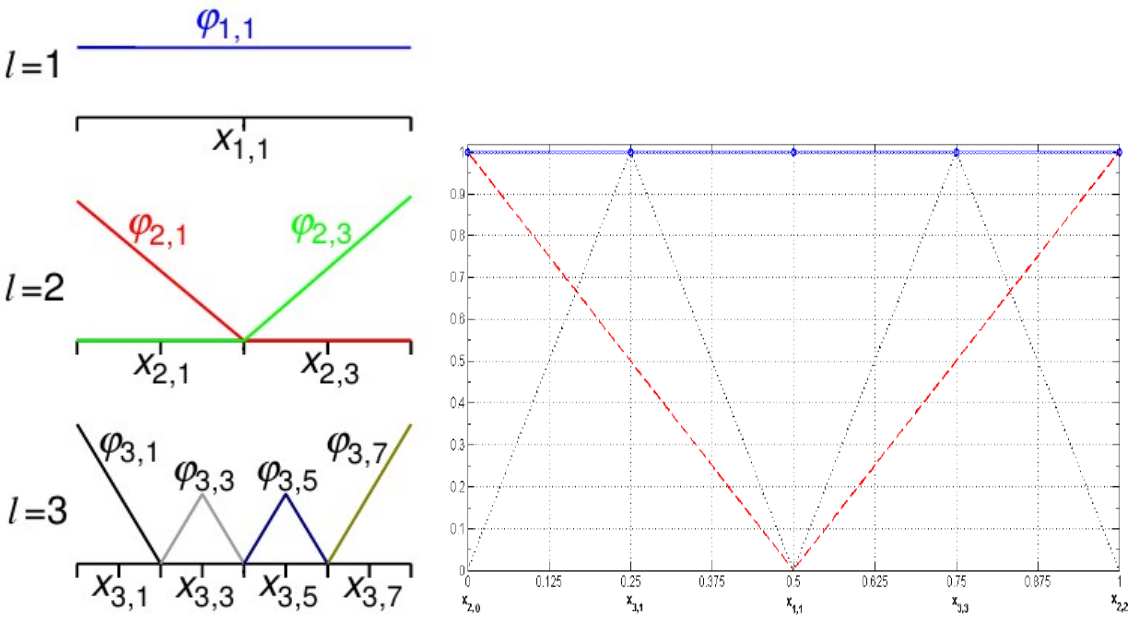


Fig.: Example of modified 1d-basis functions According to Pflüger (2010), which are extrapolating towards the boundary (**left**). They are constant on level 1 and **"folded-up"** if adjacent to the boundary on all other levels. **Right**: "Clenshaw-Curtis" basis.

Adaptive grids in general

-Ordinary sparse grids: a priori selection of grid points, optimal under certain smoothness conditions.

-Real-world applications: often do not fulfill these prerequisits (functions of interest often show kinks, finite discontinuities, steep gradients...)

→ Sparse grid methods outlined so far may fail to converge (can capture local behaviour only to some limited extend)

-Effective strategy to achieve this: **ADAPTIVITY**

→ **refine sparse grid at points around discontinuities.**

→ **spend less points in the region of smooth variation.**

Sketch of adaptive refinement

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

-Surpluses quickly decay to zero as the level of interpolation increases assuming a smooth fct.

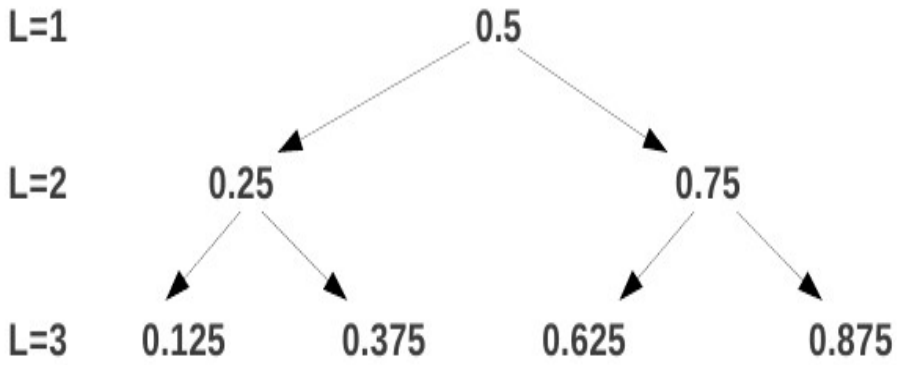
-**Use hierarchical surplus as error indicator**.

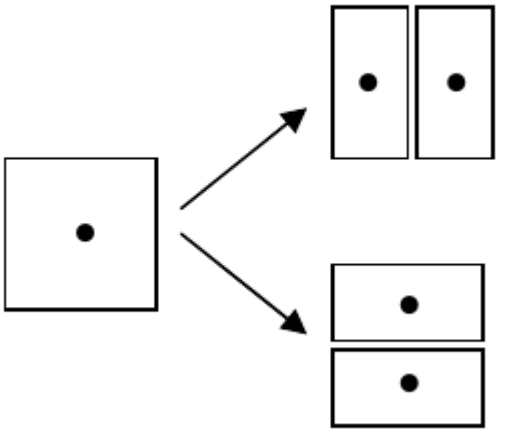
-**Automatically detect "discontinuity regions"** and adaptively refine the points in this region.

-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.

Adaptive Sparse Grid Algorithm

a) construction of interpolant level by level

b) First calculate hierarchical surplus for each point.

c) Check wether refinement criterion is satisfied $|\alpha_{\vec{l},\vec{i}}| \geq \epsilon$

d) if so, generate **2d** neighbouring points.

e) evaluate the **surplus for each new point in parallel**.

f) stop refinement
either nothing to refine, or max. level reached (if discontinuity is strong)

Test in 1d (See Genz (1984) for test functions)

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

Error measure:

→ 1000 random points from $[0,1]^d$

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

Test function:

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

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

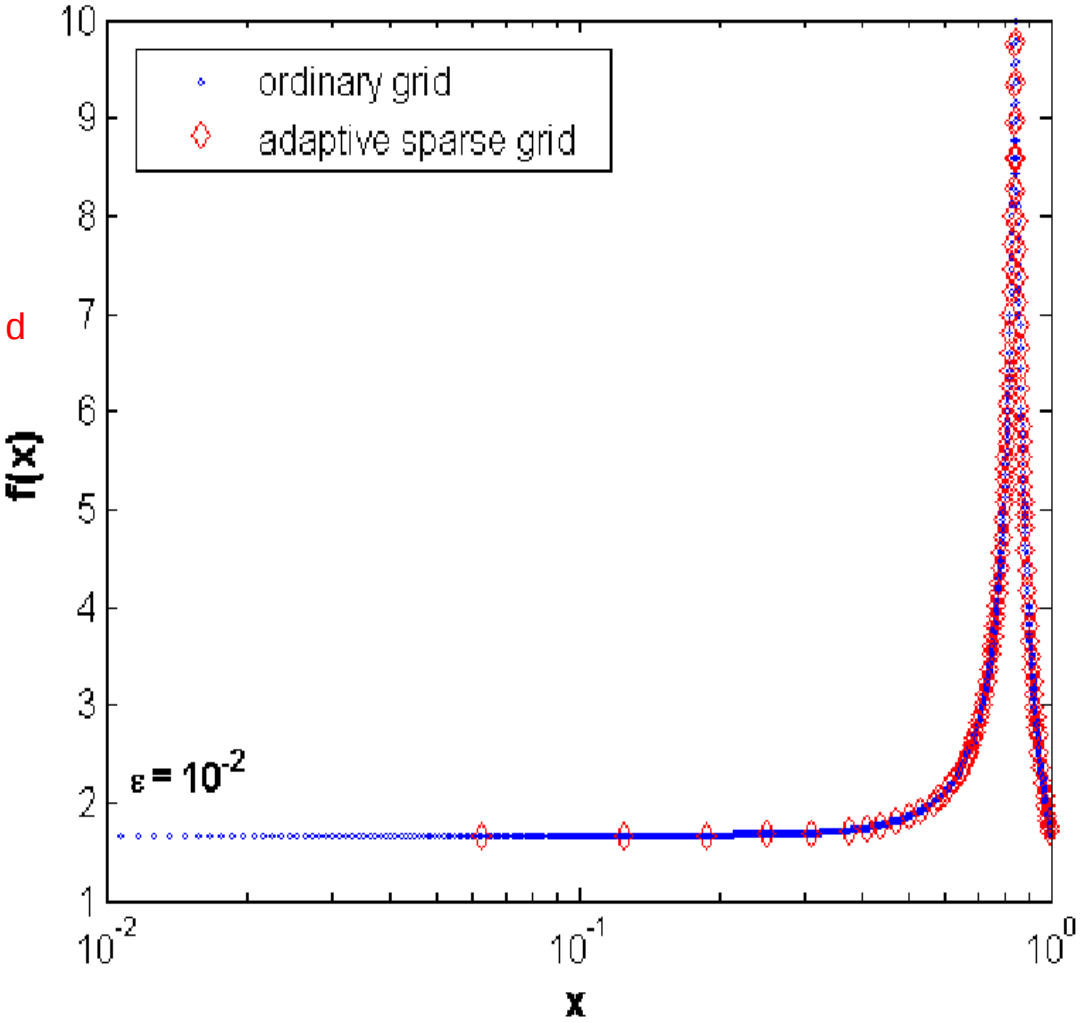


Fig.: Blue: Full grid;red: adaptive sparse grid.

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**

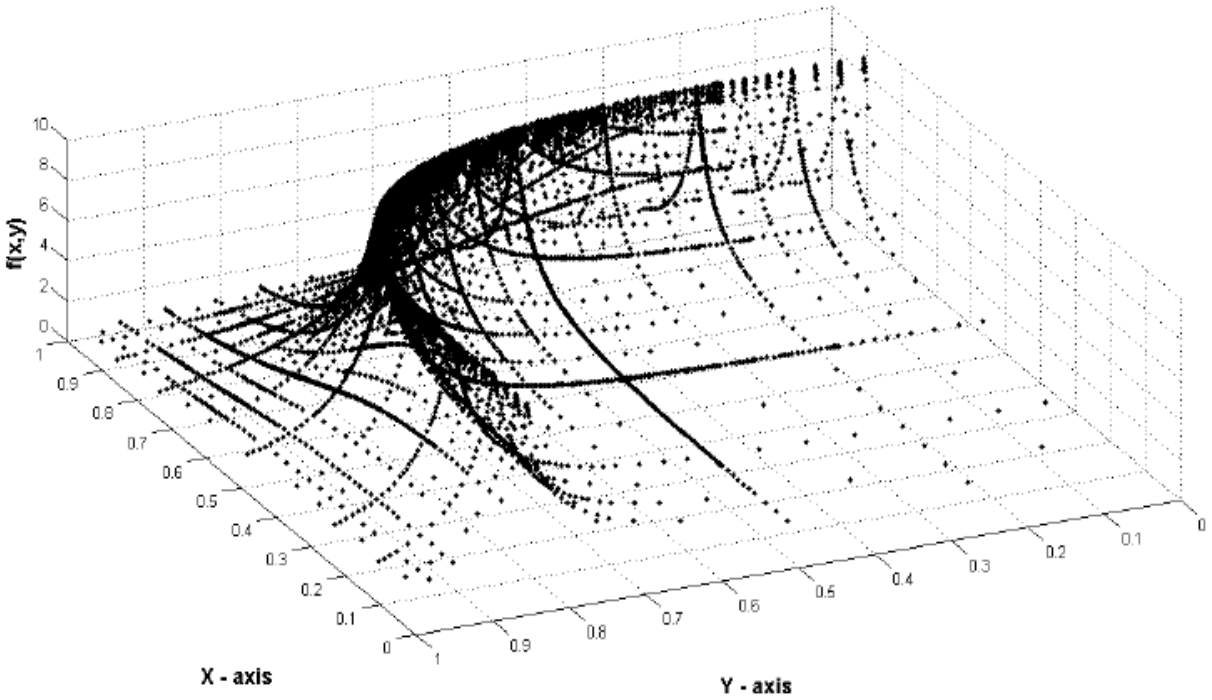


Fig.: 2d test function and its corresponding grid points after 15 refinement steps.

Movie

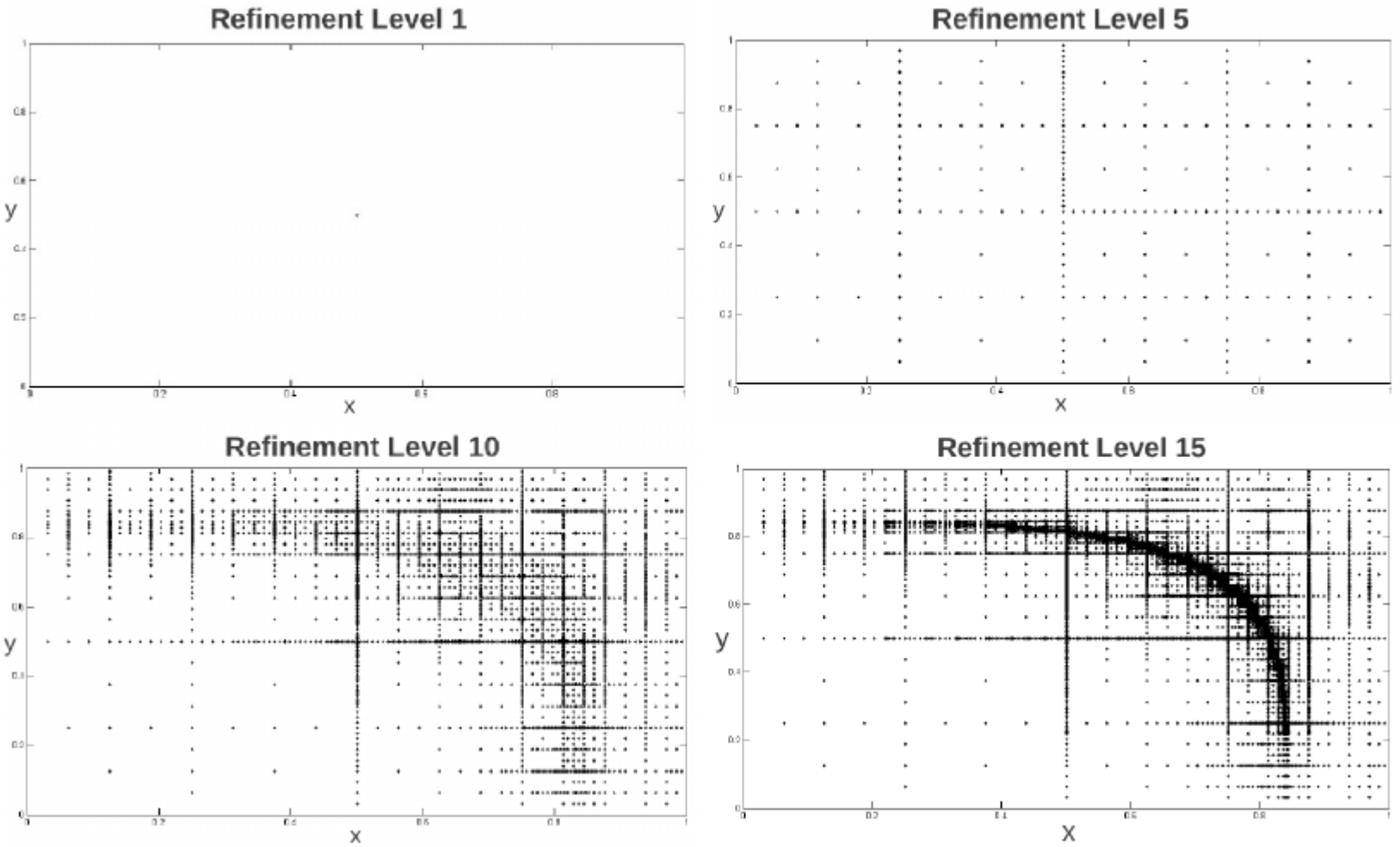


Fig.: Evolution of the adaptive sparse grid with a **threshold for refinement of 10^{-2}** . The refinement levels displayed are L = 1, 5, 10, 15.

Convergence

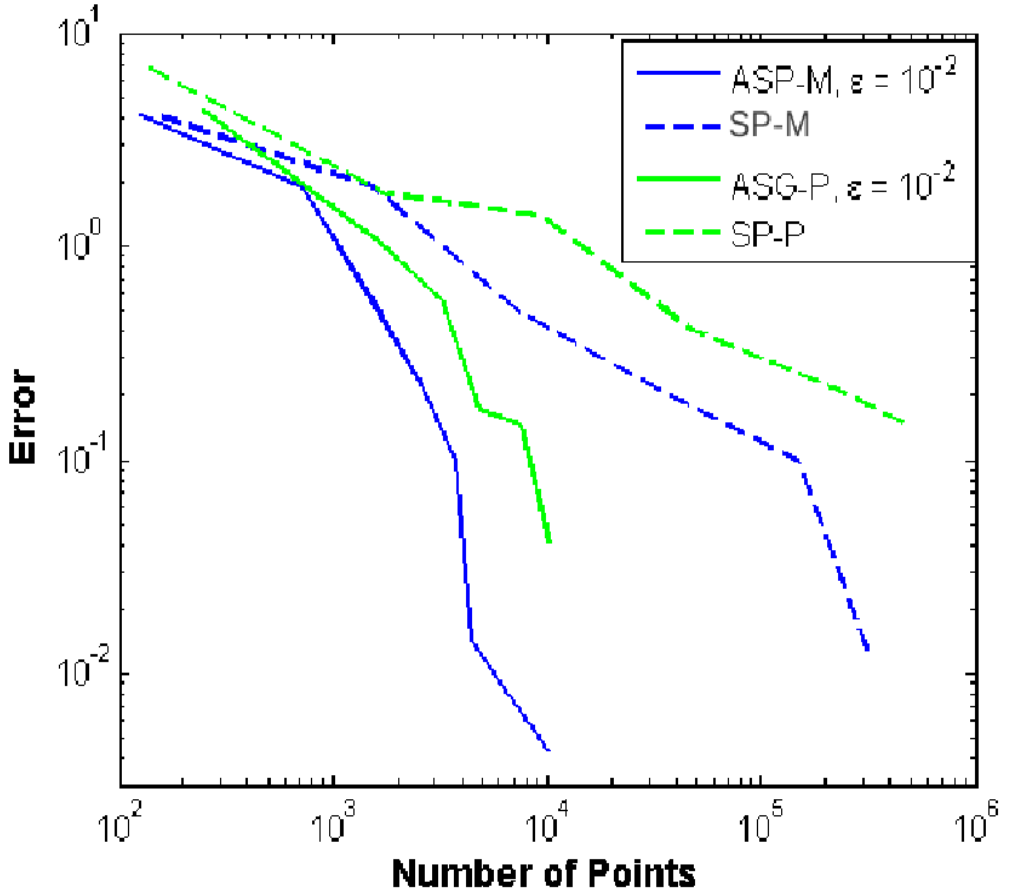


Fig.: Comparison of the interpolation error for **conventional and adaptive Sparse grid interpolation** (two different adaptive sparse grid choices - blue: Clenshaw-curtis vs. green: classical sparse grid choice).

Time iteration algorithm

We want to solve "time iteratation" problems, e.g. the International Real Business Cycle model (e.g. Den Haan et al. (2011), Malin et al. (2011))

Time iteration algorithm

1. Make an initial guess for next period's policy function:

$$p_{init} = \left(k_{t+2}^1, \ldots, k_{t+2}^N, \lambda_{t+1}\right).$$

Set $p_{next} = p_{init}$.

2. Make one time iteration step:

   (a) Using an adaptive sparse grid procedure, construct a set G of grid points

$$g = \left(a_t^1, \ldots, a_t^N, k_t^1, \ldots, k_t^N\right) \in G \subset S$$

and optimal policies at these points

$$p(g) = \left(k_{t+1}^1(g), \ldots, k_{t+1}^N(g), \lambda_t(g)\right)$$

by solving at each grid point g the system of equilibrium conditions given next period's policy

$$p_{next} = \left(k_{t+2}^1, \ldots, k_{t+2}^N, \lambda_{t+1}\right).$$

   (b) Define the policy function p by interpolating between $\{p(g)\}_{g \in G}$.

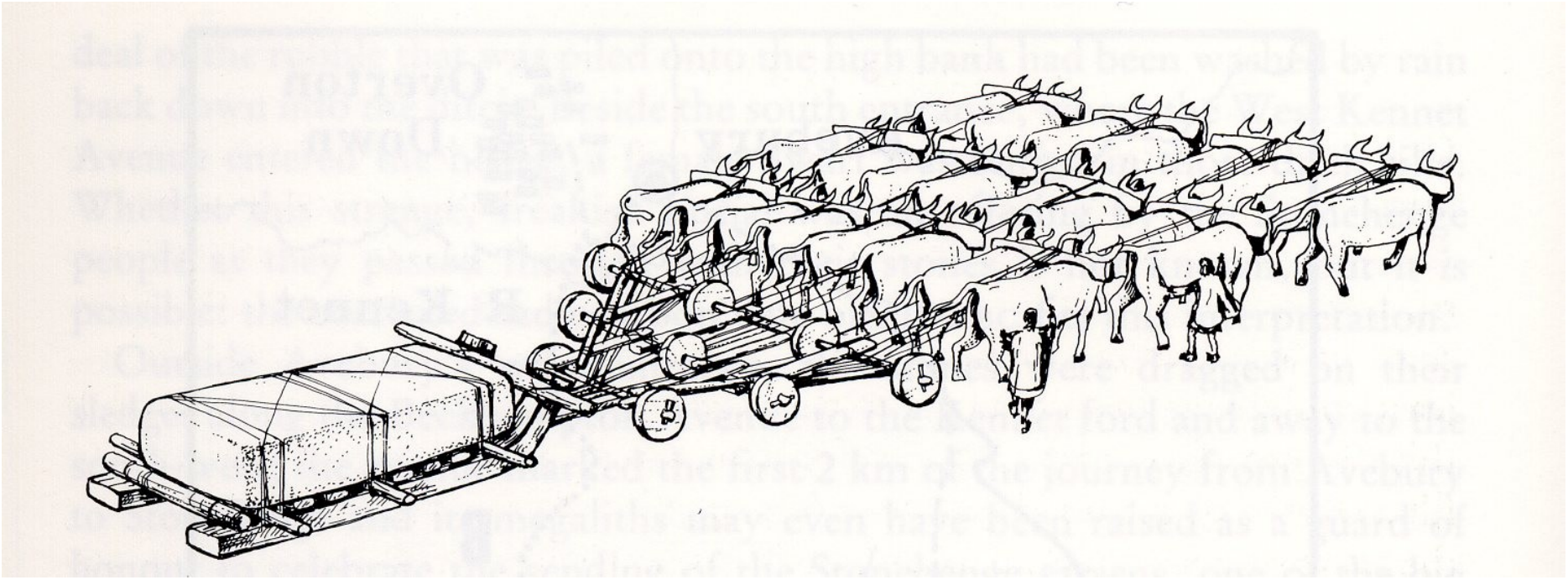
   (c) Calculate (an approximation for) the error, e.g.

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

If $\eta > \epsilon$, set $p_{next} = p$ and go to step 2, else go to step 3.

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

“To pull a bigger wagon, it is easier to add more oxen than to grow a gigantic ox” (Skjellum et al. 1999)



“To pull a bigger wagon, it is easier to add more oxen than to grow a gigantic ox” (Skjellum et al. 1999)

Fig.: **Left:** Piz Daint (CSCS)~40k Cores, **right:** UZH Schrödinger Cluster , ~5k Cores available
→ **ACCESS ON BOTH MACHINES GRANTED**

Parallel time iteration algorithm

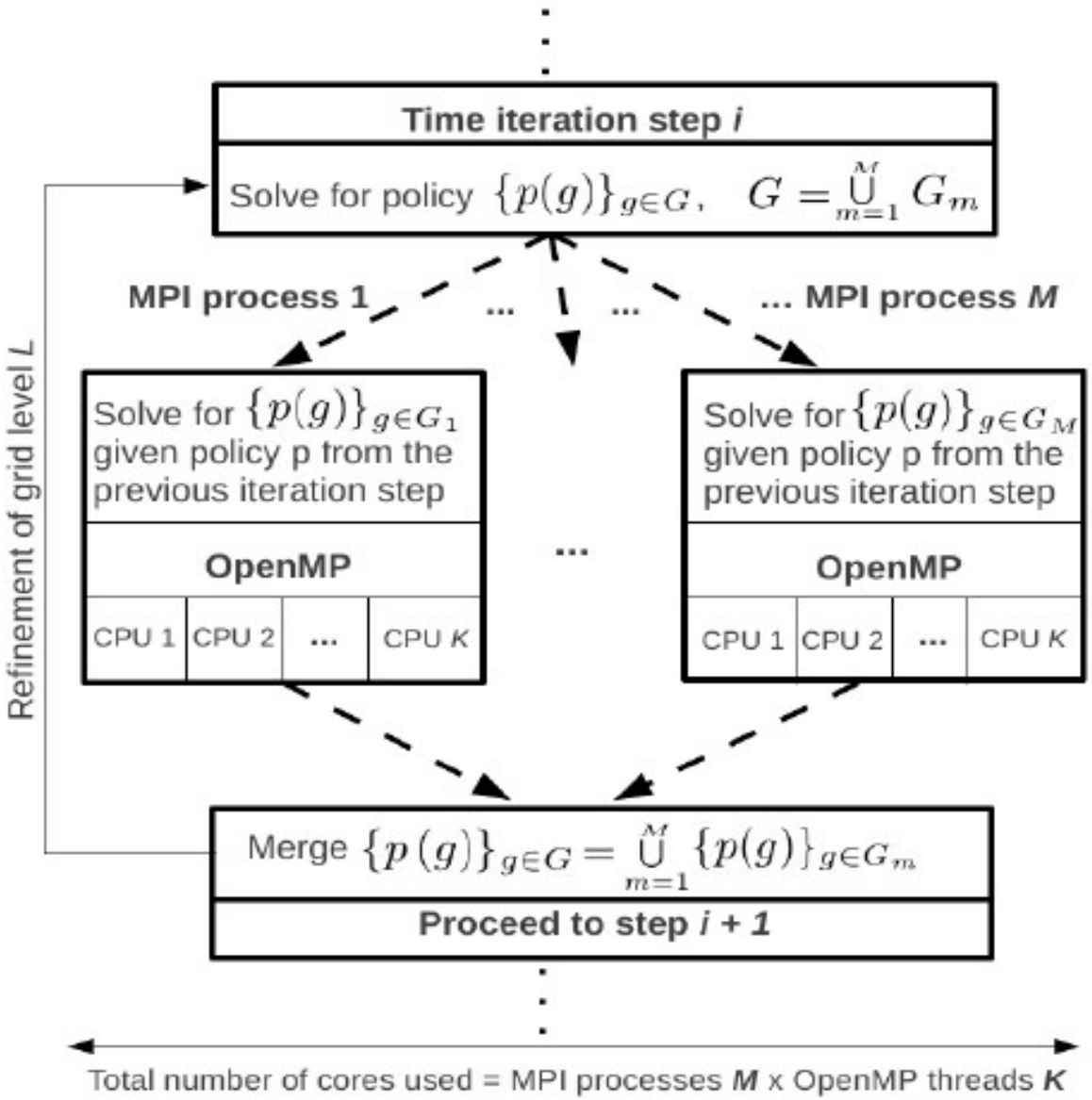
-Our implementation:
Hybrid parallel (MPI & OpenMP).

-newly generated points are distributed via MPI

Nonlinear equations locally solved on each node/MPI process by a combination of **IPOPT & PARDISO**
(Waechter & Biegler (2006), Schenk et al. (2008)).

In parallel: `messy' !

→ grid 1 needs to be visible on all MPI processes.
→ we have to ensure some sort of `load balancing'.



Scaling & Performance

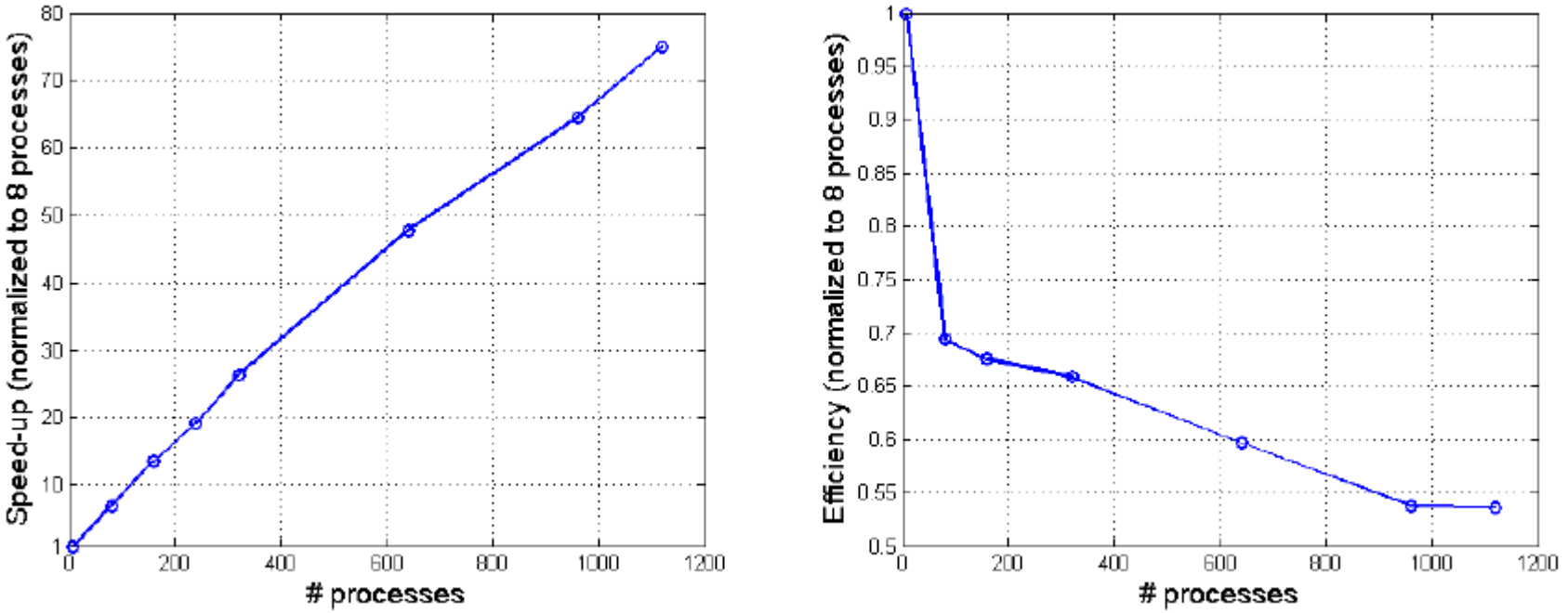


Fig.: Left panel: strong scaling of the code. Right: Efficiency.
Problem: **one timestep of a 10d** IRBC model with fixed sparse grid (level 3).

Scaling & Performance II

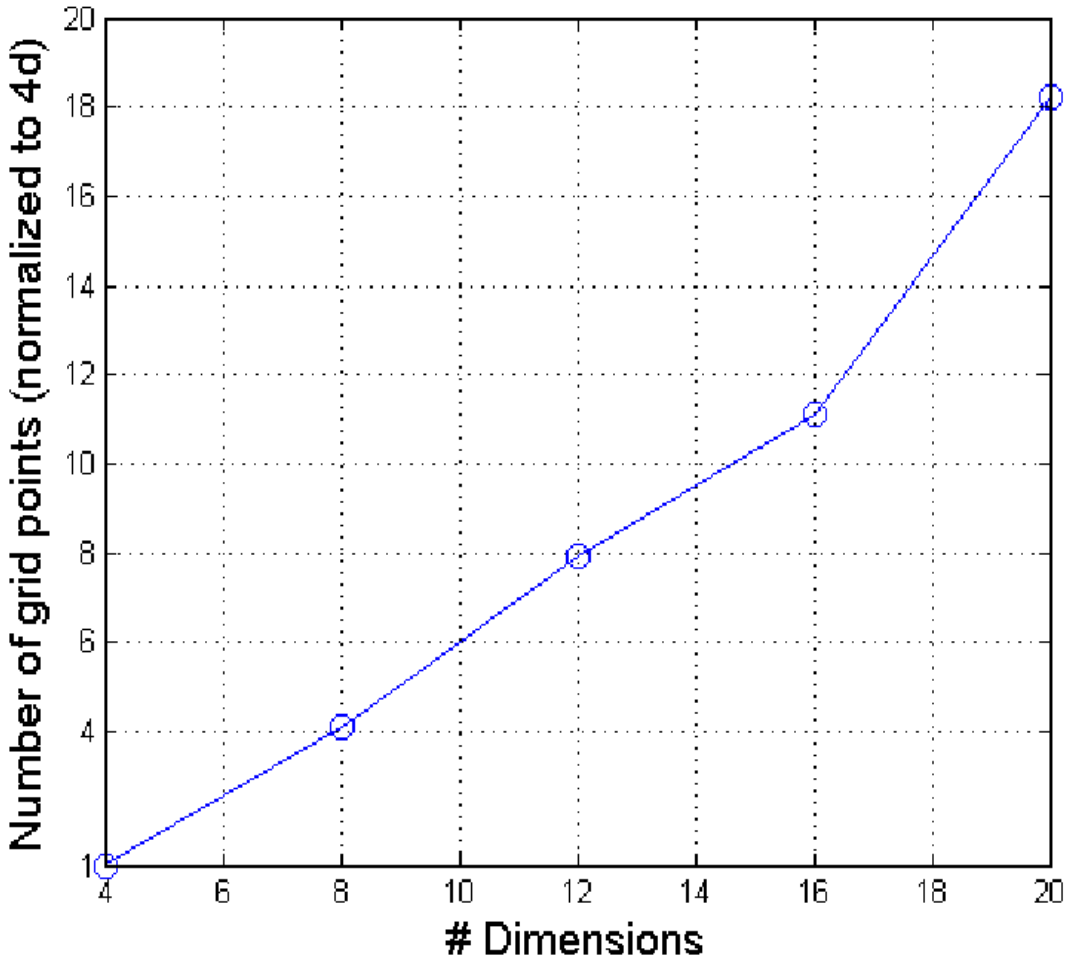


Fig.: Number of grid points grow ~O(d) with increasing dimensionality.
Test: **one timestep**, **adaptive sparse grid** with **refinement criterion *O(10 $^{-3}$)***.

Conclusion & Outlook

- **Method perfectly suited to solve high dimensional dynamic models!**
- First time adaptive sparse grids are applied to economical problems/dynamic programming
- Finalize IRBC project in order to introduce method.
- **Pull bigger wagon** (solve models that contain features which favour adaptivity).

A I: Hierarchical integration

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

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

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

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

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

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

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

A II: Model ingredients

-Want to demonstrate in a first step capabilities of code.

-**I**nternational **R**eal **B**usiness **C**ycle model
(e.g. Den Haan et al. (2011), Malin et al. (2011))

Model:
→ *N* countries (differ in **productivity** '*a*' & **capital stock** '*k*')

→ dimensionality of simulation (dim = *2N*)

→ Model: N+1 equilibrium conditions at each grid point; solved by DP

N equations:

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

1 equation:

$$\sum_{j=1}^{N} \left(a^j_t \cdot A \cdot (k^j_t)^{\alpha} + k^j_t \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g^j_{t+1})^2 \right) - k^j_{t+1} - \left(\frac{\lambda_t}{\tau_j} \right)^{-\gamma^j} \right) = 0$$

→ **We solve the nonlinear equations locally** on each node/MPI process
by a combination of **IPOPT & PARDISO** (Waechter & Biegler (2006), Schenk et al. (2008)).

AIII: Model ingredients II

$$g^j_{t+1} := k^j_{t+1}/k^j_t - 1, \; g^j_{t+2} := k^j_{t+2}/k^j_{t+1} - 1,$$

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

$$\sum_{j=1}^{N} \left(a^j_t \cdot A \cdot (k^j_t)^{\alpha} + k^j_t \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g^j_{t+1})^2 \right) - k^j_{t+1} - \left(\frac{\lambda_t}{\tau_j} \right)^{-\gamma^j} \right) = 0.$$

We solve the model by iterating on these FOCs, i.e. at each grid point

$$(a^1_t, \ldots, a^N_t, k^1_t, \ldots, k^N_t) \longrightarrow \text{Current grid points}$$

solve for the unknown policy variables

$$(k^1_{t+1}, \ldots, k^N_{t+1}, \lambda_t) \longrightarrow \text{Solution of system of Eqs. At the current iteration}$$

given the known policy functions from last iteration

$$(k^1_{t+2}(a_{t+1}, k_{t+1}), \ldots, k^N_{t+2}(a_{t+1}, k_{t+1}), \lambda_{t+1}(a_{t+1}, k_{t+1})) \longrightarrow \text{Interpolant}$$

evaluated at next periods state

$$(a^1_t, \ldots, a^N_t, k^1_t, \ldots, k^N_t), \text{where } a^j_{t+1} = (a^j_t)^{\rho} \cdot e^{\sigma(e_t + e^j_t)}.$$