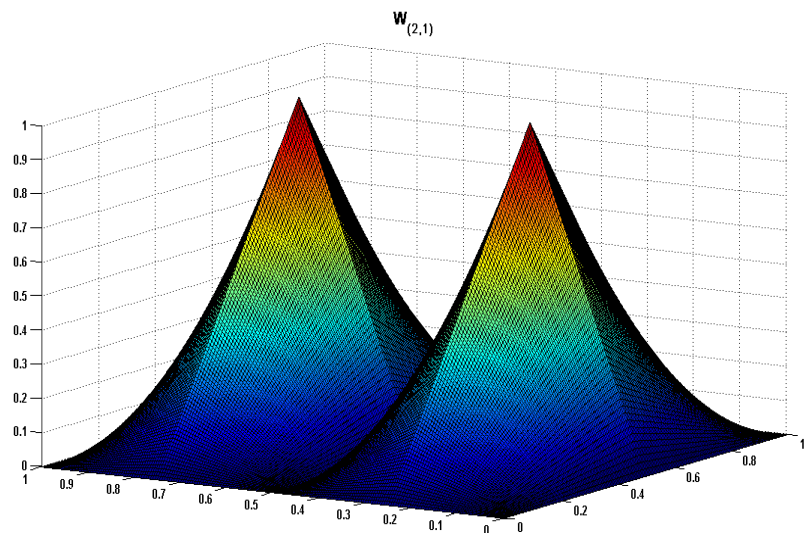
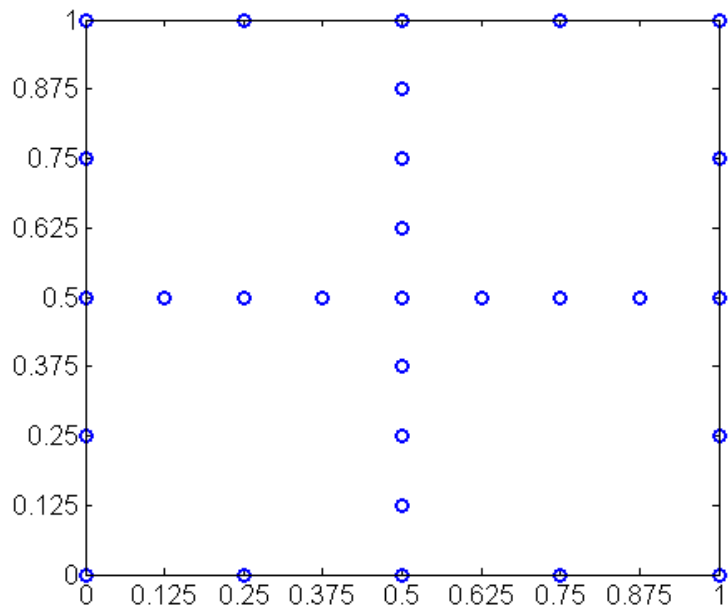




Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models

J. Brumm & S. Scheidegger

BFI, University of Chicago, Nov. 1st 2013



Outline

- I.) From Full (Cartesian) Grids to Sparse Grids
- II.) Adaptive Sparse Grids
- III.) Time Iteration, Adaptive Sparse Grids & HPC
 - Implementation, Testing, Results

Our motivation

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

$$“\Theta v = v” \longleftrightarrow |\Theta v_i - v_{i+1}| < \varepsilon$$

Problem: curse of dimensionality

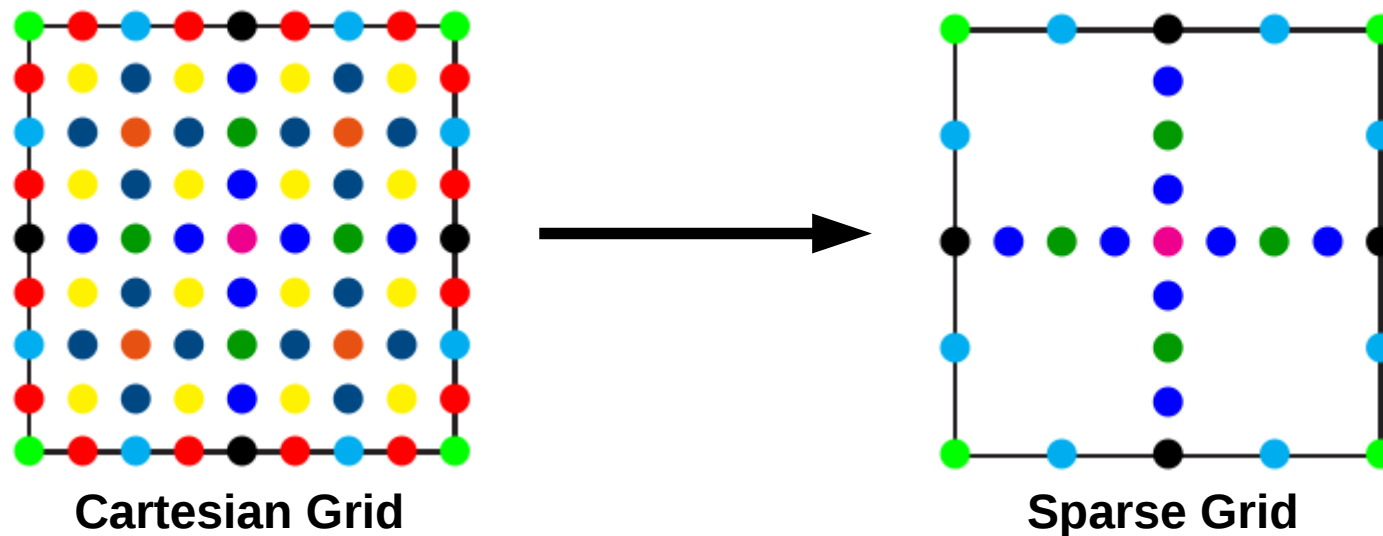
—► N^d points in ordinary discretization schemes

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)

I. From Full Grids to Sparse Grids



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

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

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

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

Interpolation on a Full Grid

- Consider a **d-dimensional function** $f : \Omega \rightarrow \mathbb{R}$
- In numerical simulations:
 f might be expensive to evaluate!
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 first on case where $f|_{\partial\Omega} = 0$

Basis Functions

-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**

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

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:

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

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: support of all basis functions of W_k mutually disjoint!

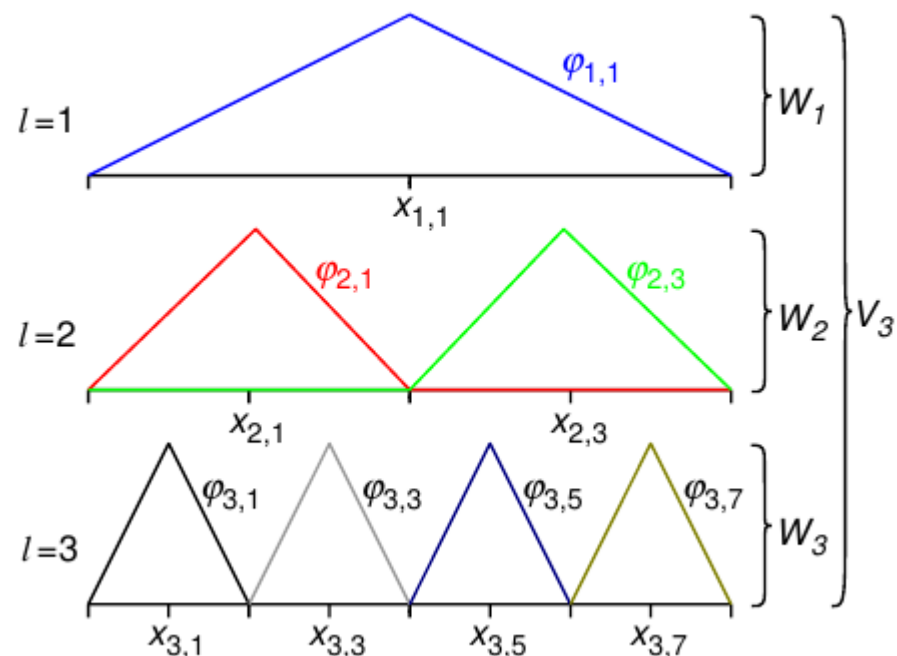


Fig.: 1-d basis functions $\phi_{l,i}$ and the corresponding grid points up level $l=3$ in the hierarchical basis.

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:

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

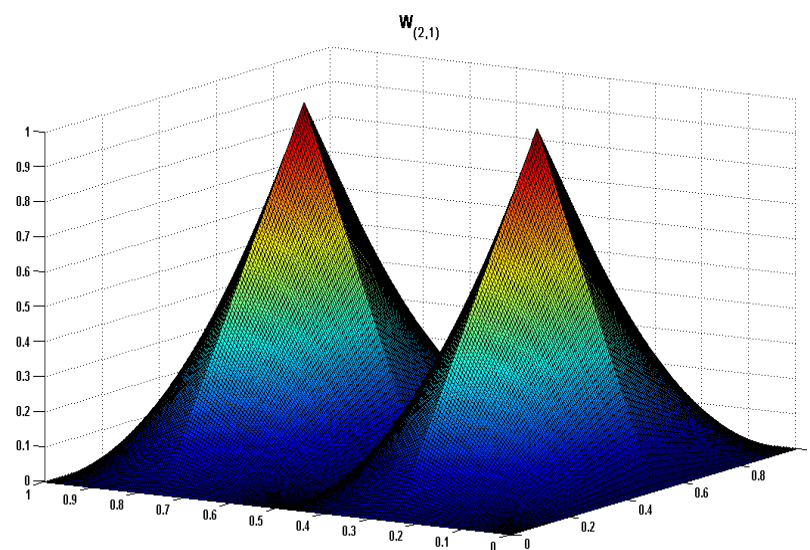


Fig.: Basis functions of the **subspace $W_{2,1}$**

Why reality bites...

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

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

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

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

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

Hard to handle more than 4 dimensions numerically

—► e.g. $d=10$, $n = 4$, 15 points/d, **5.8×10^{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 ✓

→ The second mixed derivatives have to be bounded

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

These functions belong to a Sobolev space

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

Under this prerequisite, the coefficients decay rapidly

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

'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_∞ – Norm.

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

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

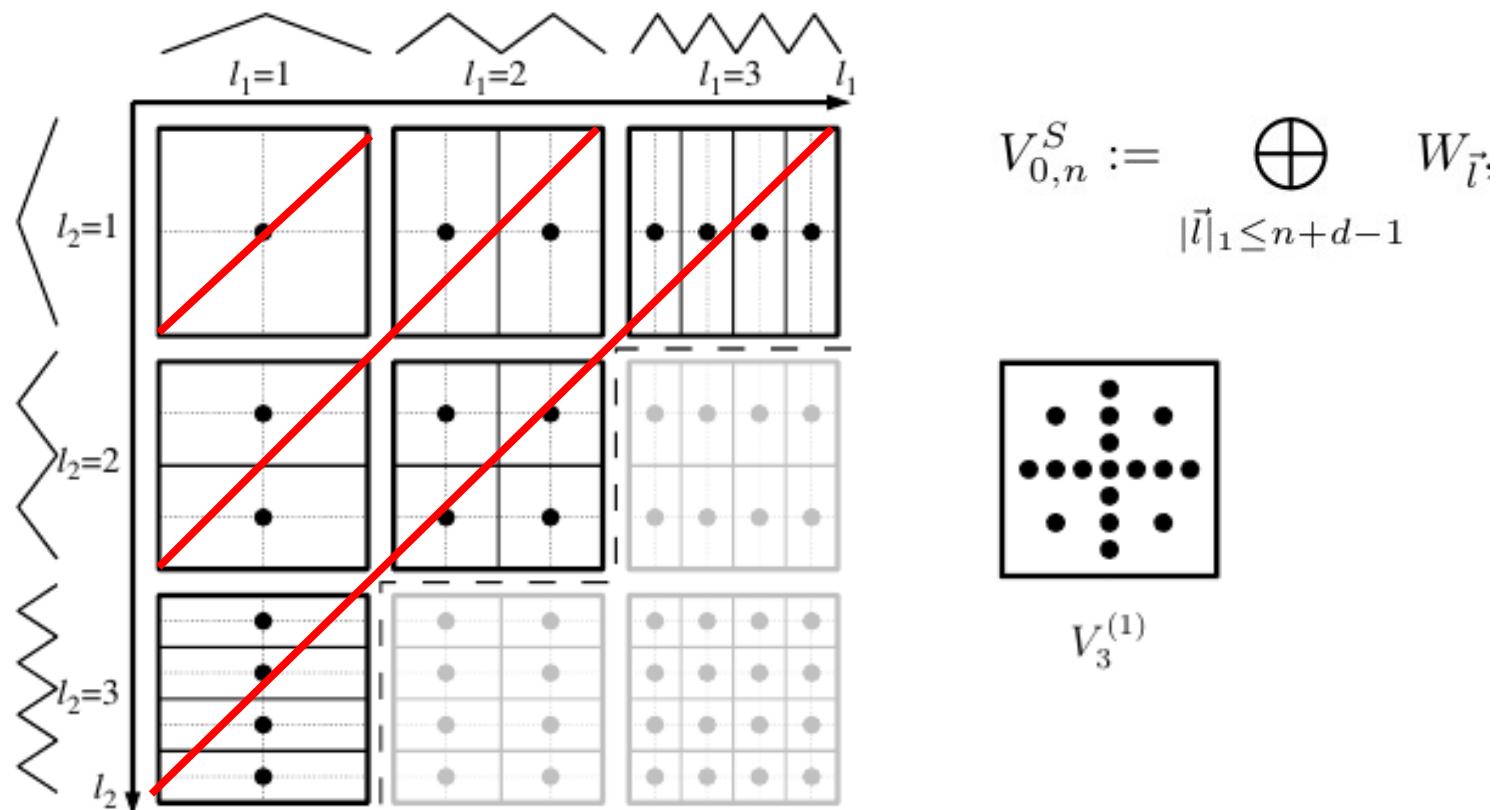
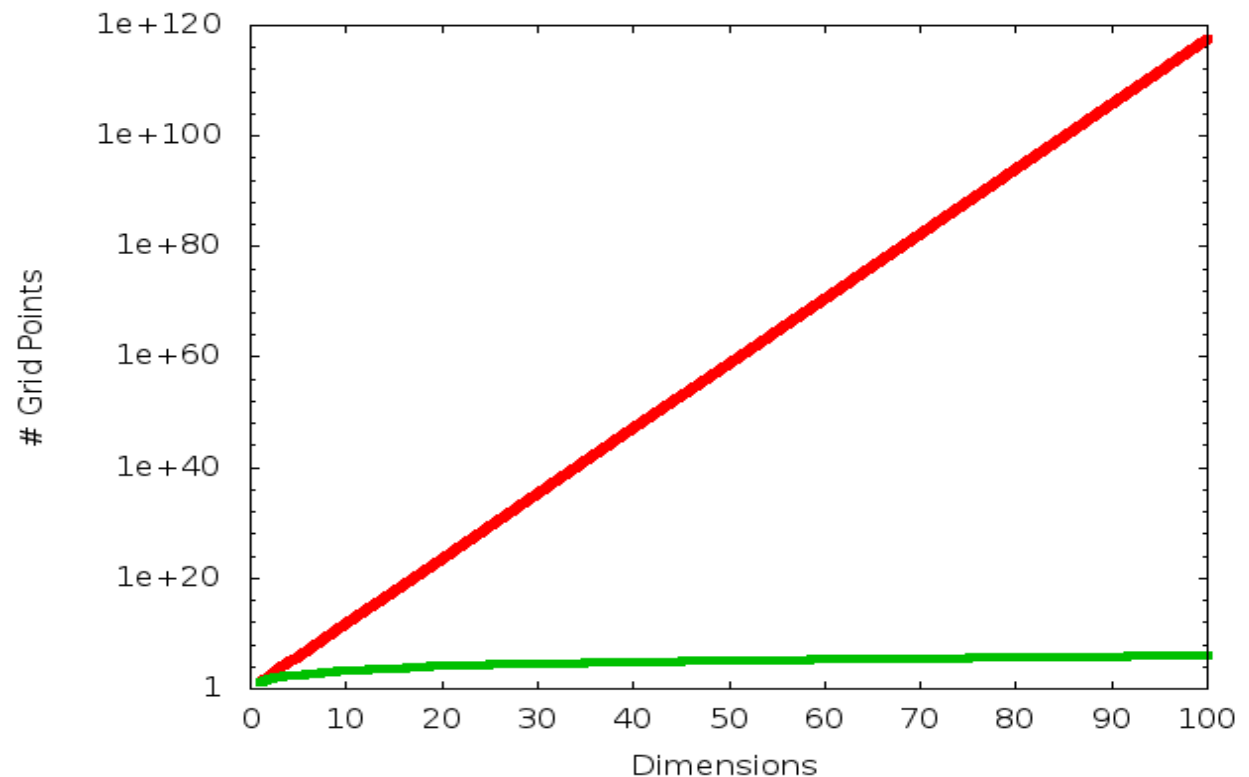


Fig.: Two-dimensional subspaces $W_{\vec{l}}$ up to $l=3$ ($h_3 = 1/8$) in each dimension. The **optimal a priori selection of subspaces** is shown in black (**left**) and the Corresponding sparse grid of level $n = 3$ (**right**). For the **full grid**, the gray subspaces have to be used as well.

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).

Hierarchical surplus – nested structure

Coefficients of interpolant termed
“hierarchical surpluses”.

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

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

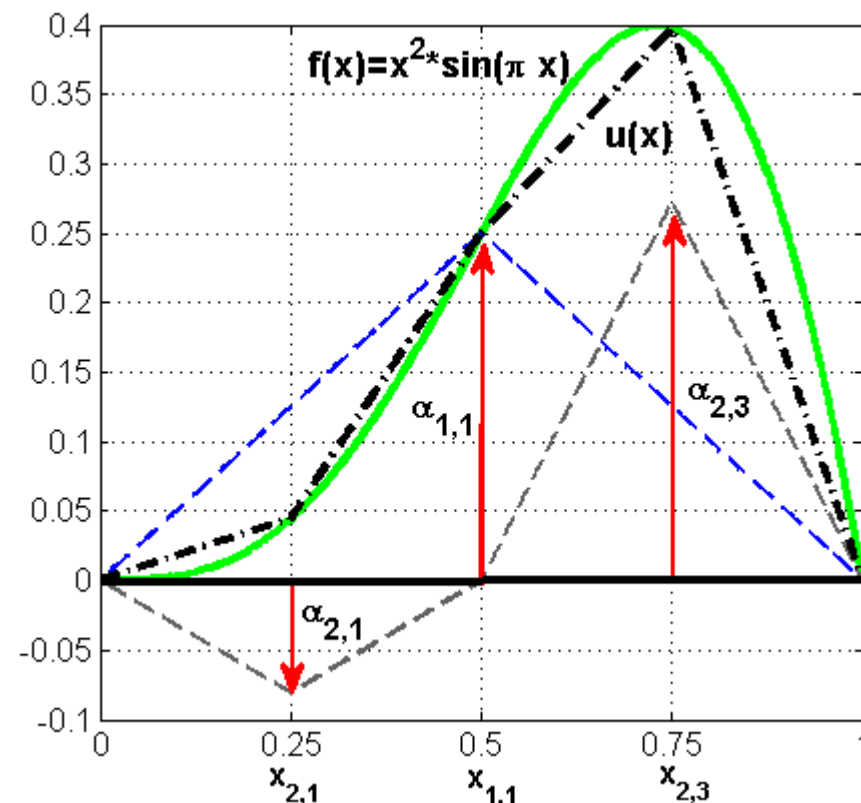


Fig.: Interpolant u of f of level 2.

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.**

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 & Wohlmuth (2005),...

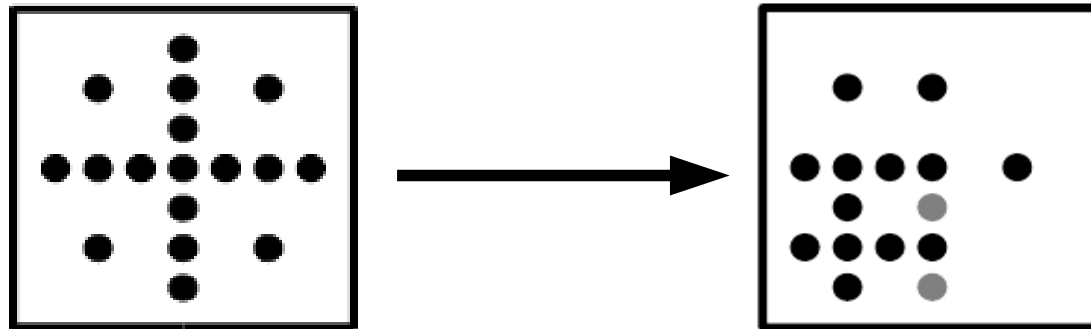
- Solution of PDEs

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), Judd et al. (2013)

II. Adaptive Sparse Grids



Adaptive grids in general

- Ordinary sparse grids: a priori selection of grid points, optimal under **certain smoothness conditions**.
- Real-world applications: **often do not fulfil these prerequisites** (functions of interest often show kinks, finite discontinuities, steep gradients...)
 - **Sparse grid methods outlined so far may converge slowly** (can capture local behaviour only to some limited extend)
- Effective strategy to achieve this: **ADAPTIVITY**
 - **refine sparse grid in regions of high curvature.**
 - **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.

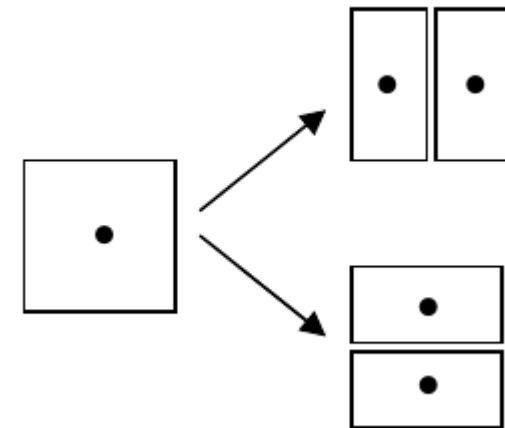
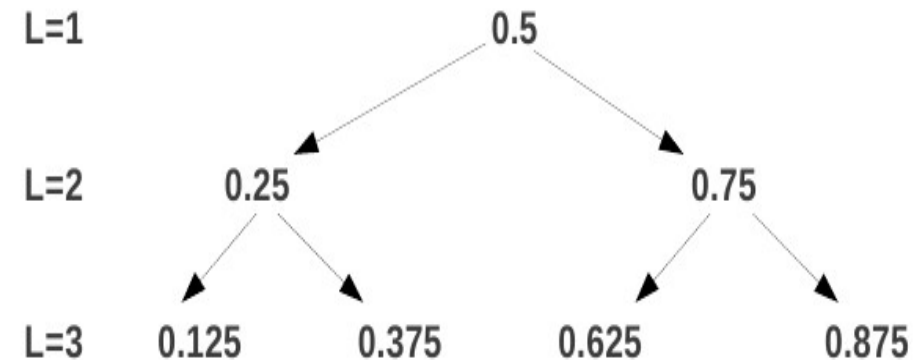
-Use hierarchical surplus as error indicator.

-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.

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

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

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

c) stop refinement

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

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 $[0,1]^d$

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

Full grid: **1023** points

Adaptive sparse grid: **109** points.

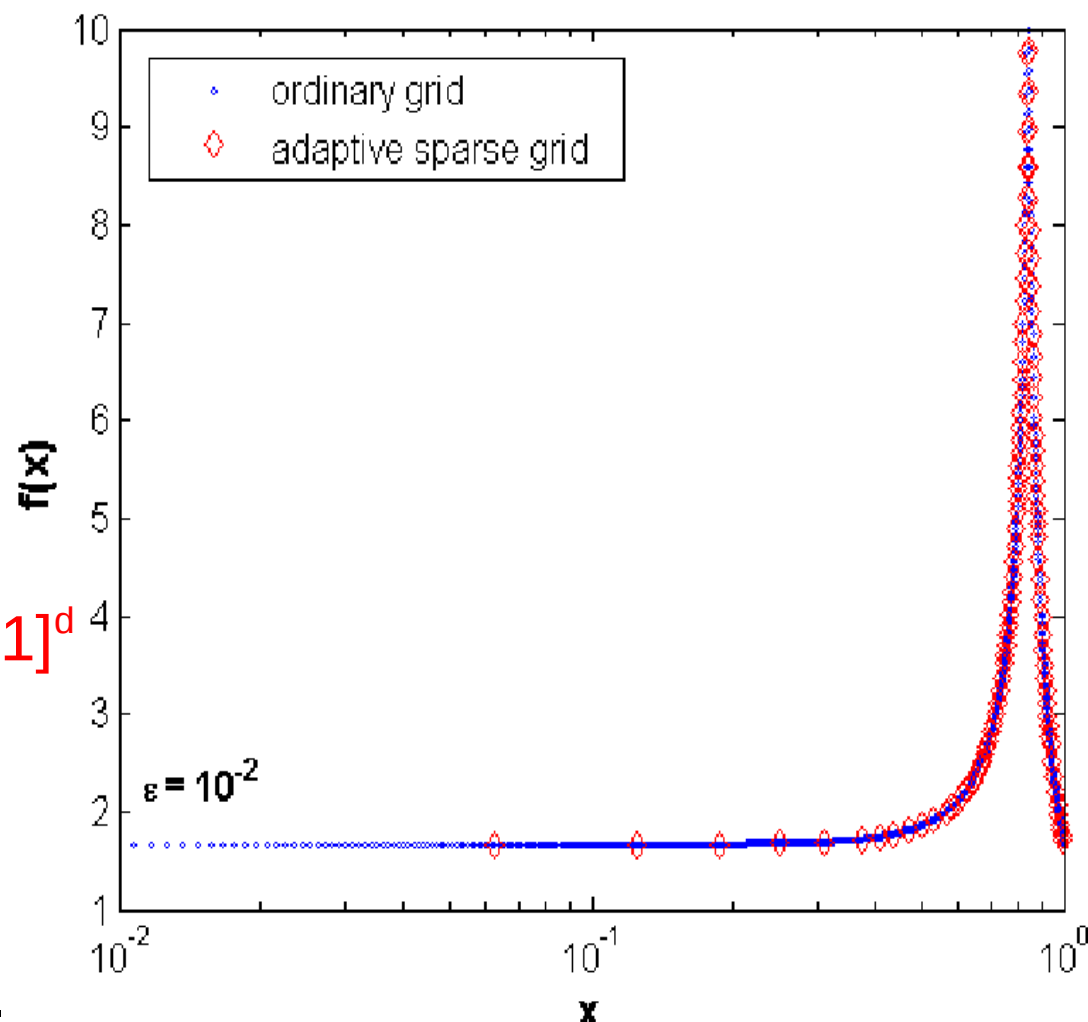


Fig.: Blue: Full grid; red: adaptive sparse grid.

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**

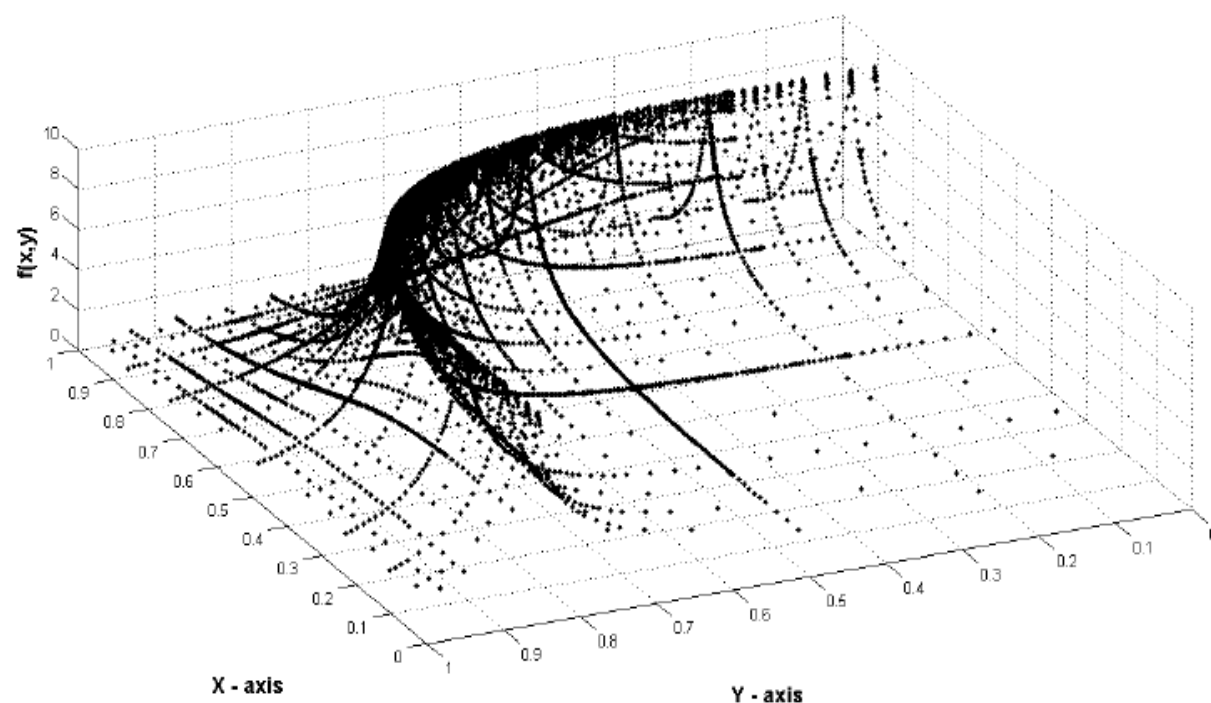


Fig.: 2d test function and its corresponding grid points after 15 refinement steps.

Movie

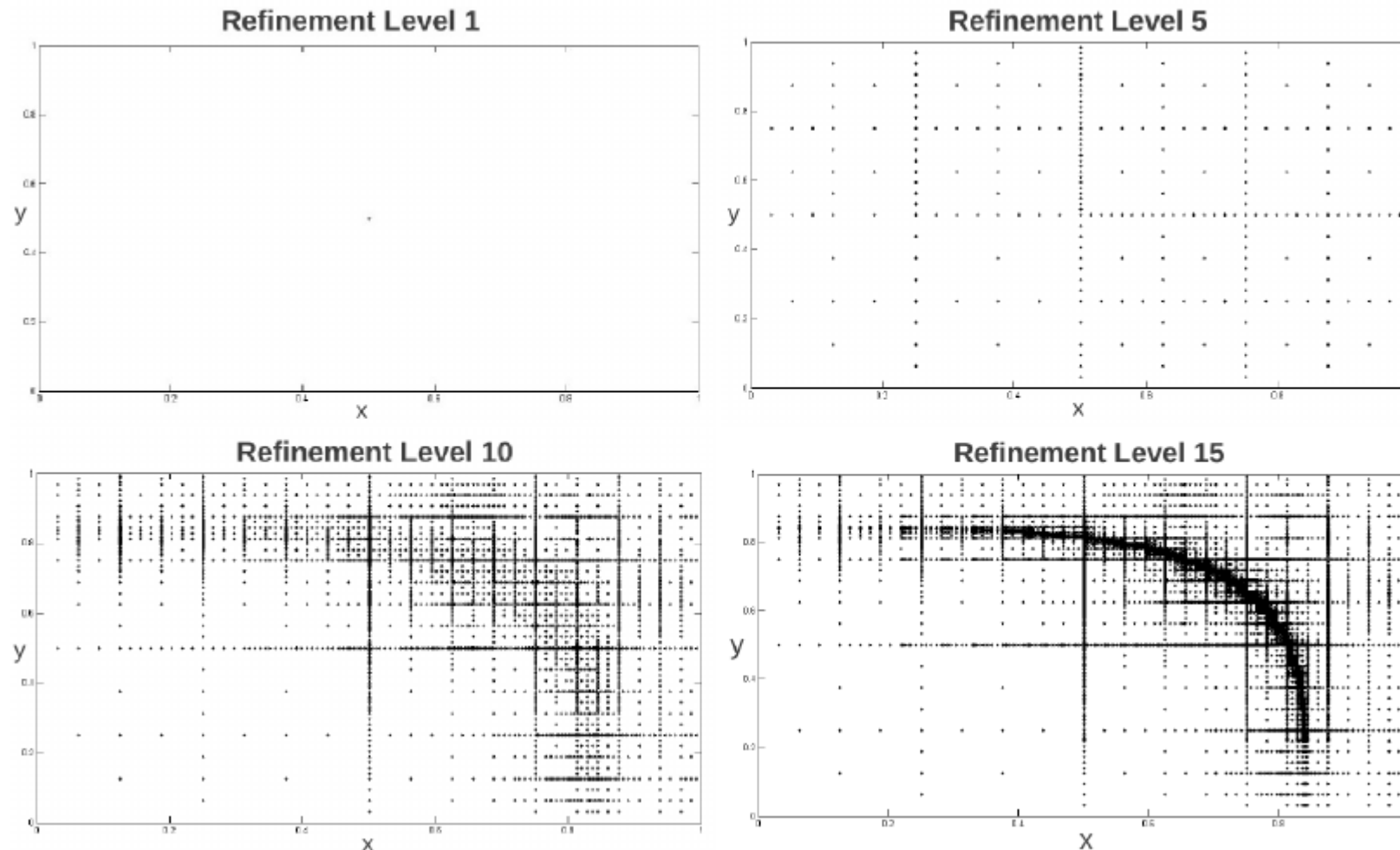


Fig.: Evolution of the adaptive sparse grid with a **threshold for refinement of 10^{-2}** . The refinement levels displayed are $L = 1, 5, 10, 15$.

Convergence

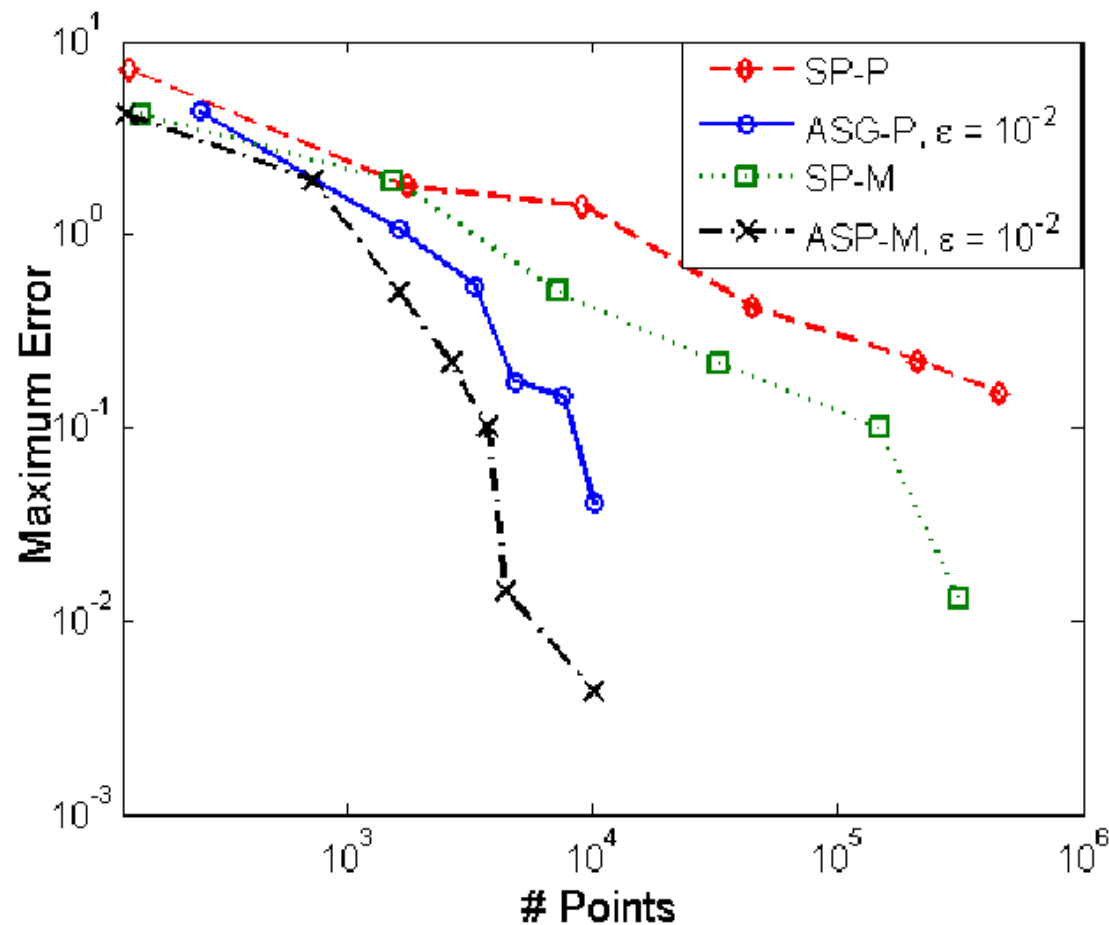


Fig.: Comparison of the interpolation error for **conventional and adaptive Sparse grid interpolation** (two different adaptive sparse grid choices - blue: Clenshaw-curtis vs. green: classical sparse grid choice).

III. Putting things together: Time Iteration, Adaptive Sparse Grids & HPC



IRBC: 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** (exogen.) ' a ' & **capital stock** (endogen.) ' k ')

→ **dimensionality of state space, i.e. grid (dim = $2N$)**

→ Model: **$N+1$** equilibrium conditions at each grid point

N equations:

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

1 equation:

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

Time iteration algorithm

Time iteration algorithm

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

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

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

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

and optimal policies at these points

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

by solving at each grid point g the system of equilibrium conditions (cf., Eqs. 51 and 52) given next period's policy

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

- (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 .

Time Iteration Algorithm II

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

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

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

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

$$(a_t^1, \dots, a_t^N, k_t^1, \dots, k_t^N) \longrightarrow \text{Current grid points}$$

solve for the unknown policy variables

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

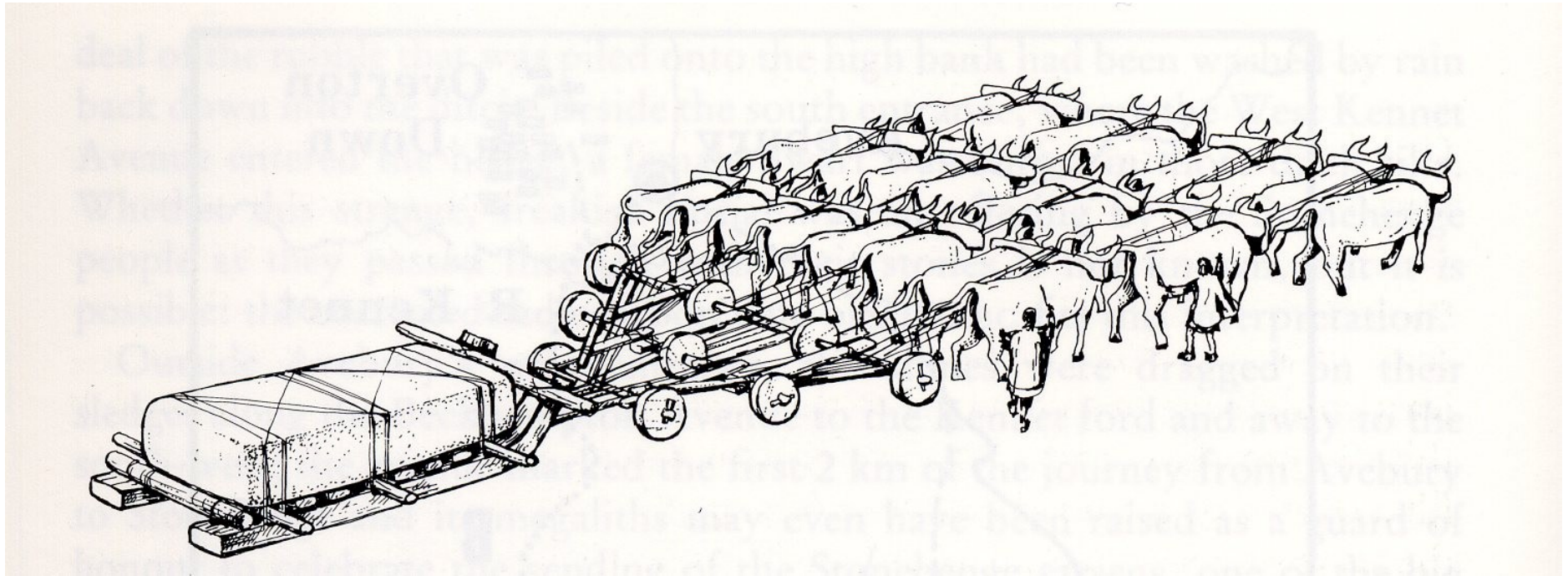
given the known policy functions from last iteration

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

evaluated at next periods state

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

“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)



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

Parallel time iteration algorithm

-Our implementation:

Hybrid parallel (MPI & OpenMP).

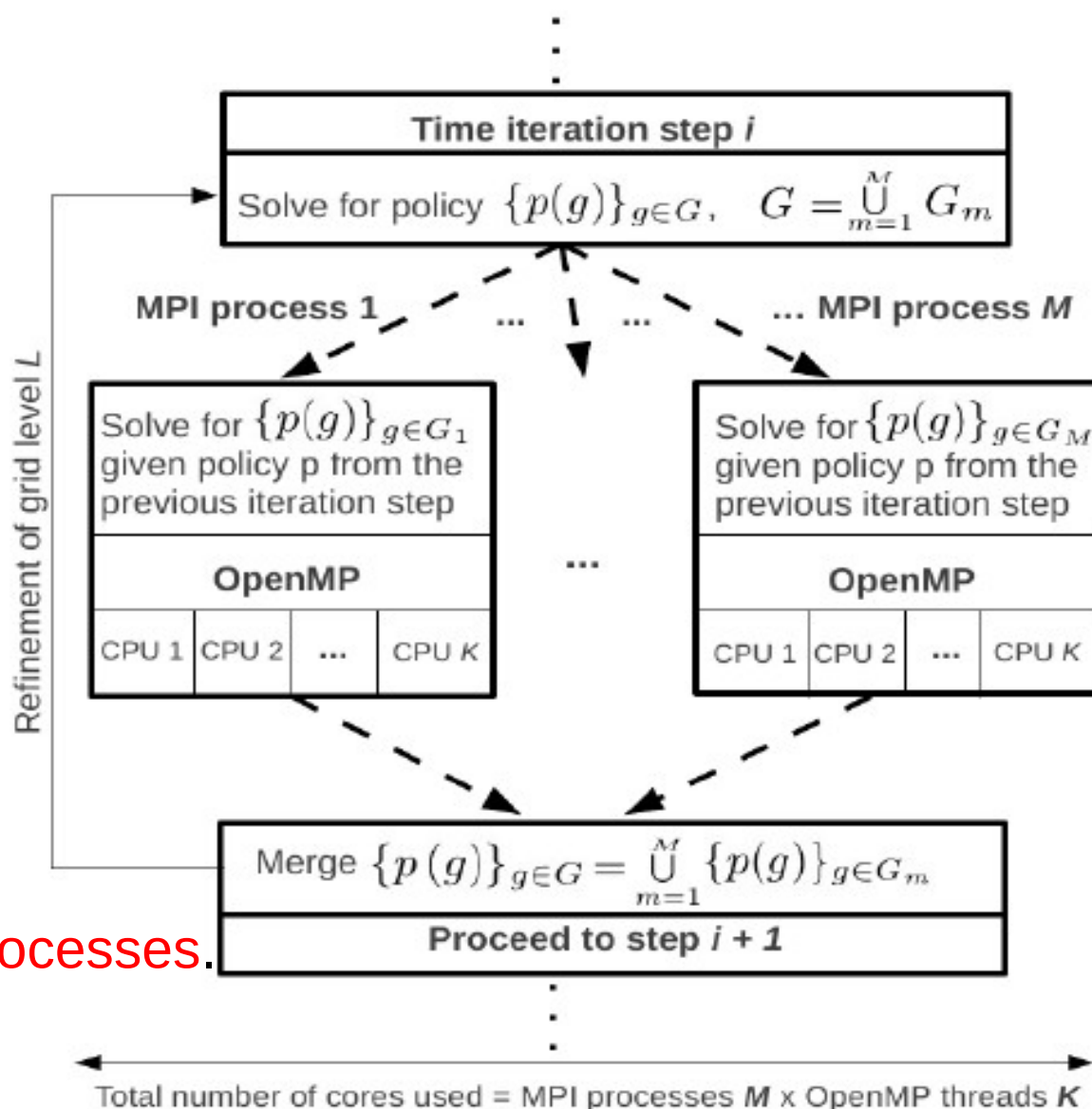
-newly generated points are distributed via MPI

Nonlinear equations locally solved on each node/MPI process by a combination of **IPOPT & PARDISO**

(Wächter & Biegler (2006), Schenk et al. (2008)).

In parallel: 'messy' !

- policy from previous iteration has to be **visible on all MPI processes**.
- we have to ensure some sort of **'load balancing'**.



Scaling & Performance

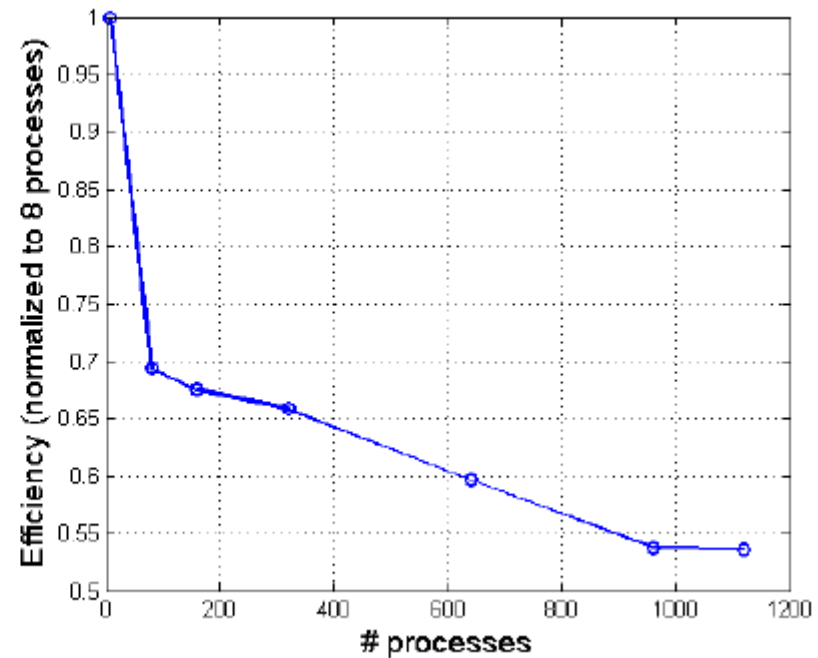
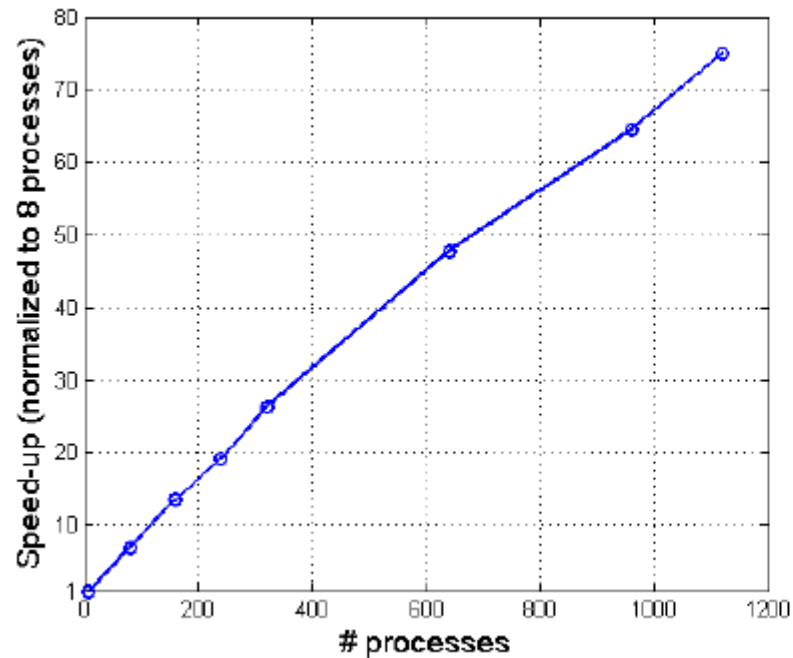


Fig.: Left panel: strong scaling of the code. Right: Efficiency.
 Problem: **one timestep of a 10d IRBC model** with fixed sparse grid (level 3).

Scaling & Performance II

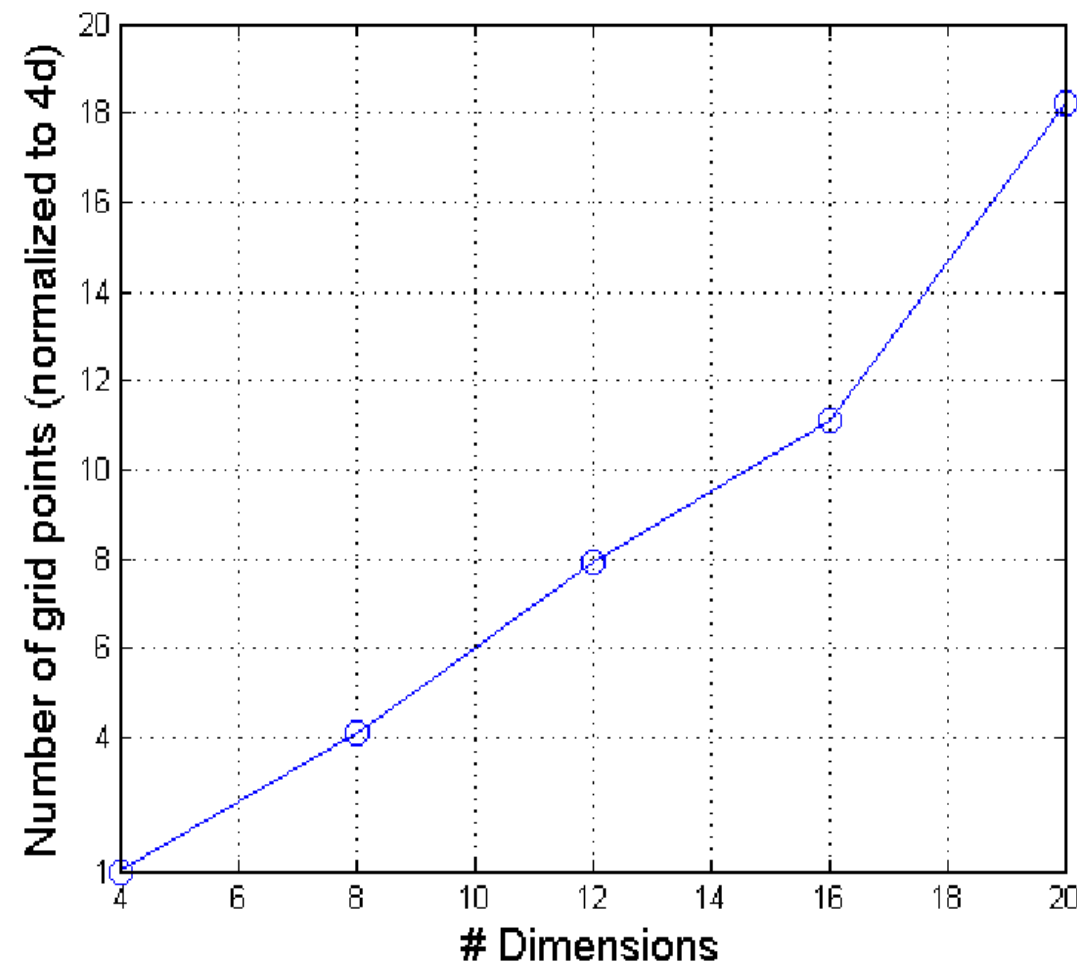


Fig.: Number of grid points grow $\sim O(d)$ with increasing dimensionality.
Test: **one timestep, adaptive sparse grid** with **refinement criterion $O(10^{-3})$** .

Errors: Results – IRBC

Increase Level
→ Errors get better



Dimension	Level	Points	Max. Error	Avg. Error
4	3	137	-2.82	-3.68
4	4	401	-2.98	-4.19
4	5	1105	-3.19	-4.24
4	6	2929	-3.28	-4.55

Increase Dimension
→ errors remain
of same quality



Dimension	Level	Points	Max. Error	Avg. Error
4	3	137	-2.97	-3.55
8	3	849	-3.04	-3.83
12	3	2649	-2.71	-3.78
16	3	6049	-2.72	-3.80
20*	2	841	-2.58	-3.29
22*	2	1013	-2.60	-3.29

Augment IRBC with 'kinks'

-Want to demonstrate that we are able to handle non-smooth behaviour.

-We include irreversible investment in the IRBC:

installed capital cannot be consumed or moved to other country.

Model:

→ N countries (differ in **productivity** (exogen.) ' a ' & **capital stock** (endogen) ' k ')

→ **dimensionality of simulation - grid (dim = $2N$)**

→ for each country, a resource constraint has to be satisfied

N equations: $k_{t+1}^j - k_t^j(1 - \delta) \geq 0, \mu_t^j \geq 0, \left(k_{t+1}^j - k_t^j(1 - \delta)\right) \cdot \mu_t^j = 0 \quad (\rightarrow \text{'kinks'})$

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

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

IRBC with irreversible investment

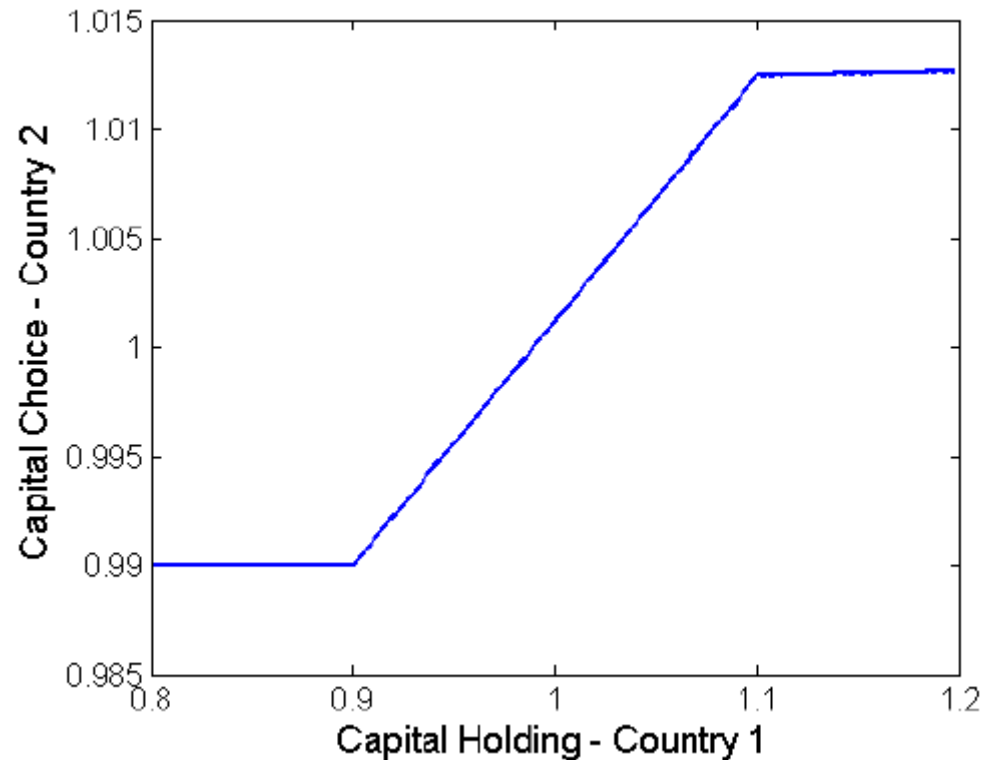


Fig.: Capital choice of country 2 as a function of capital holding of country 1. All other state variables of this model are kept fixed at steady state.

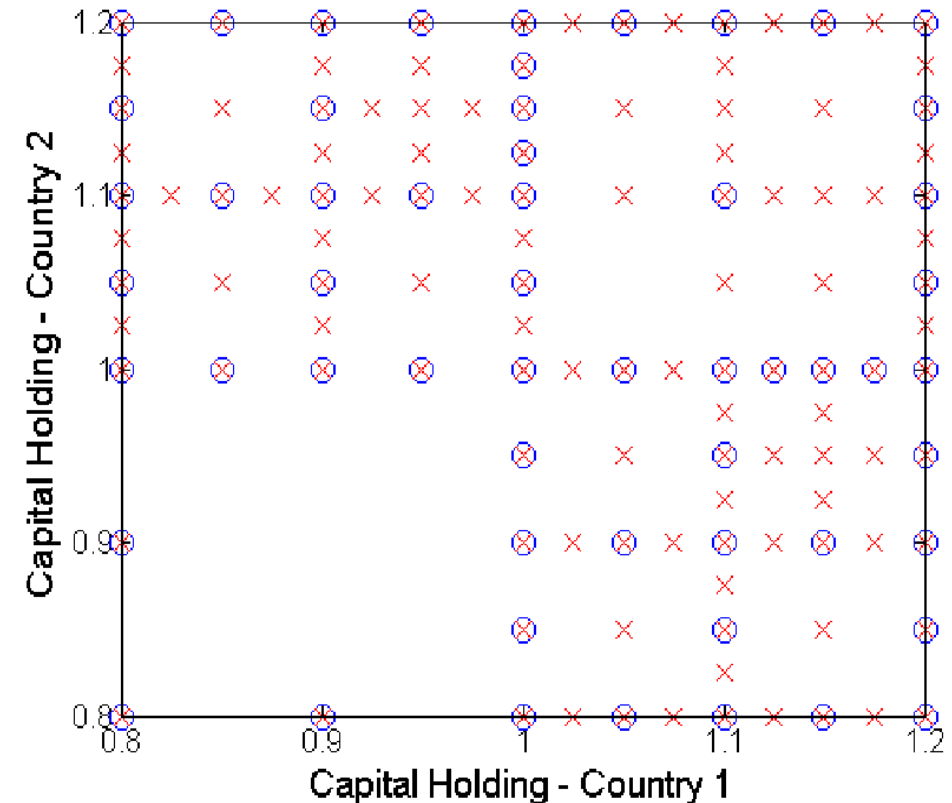


Fig.: A two-dimensional slice of an Adaptive sparse grid that was generated in the course of running a $2N = 4d$ Simulation.

Blue: grid of refinement $L = 6$,
Red: refinement level $L = 8$.

Results – IRBC with kinks

Dimension	Level	Points	Max. Error	Avg. Error
4	2	41	-1.63	-2.70
4	4	401	-1.71	-2.87
4	6	2929	-1.71	-3.08

Fig.:Sparse grid: errors hardly improve!

Level	Statistics	Refinement Threshold			
		$\epsilon_{k_i} = 0.015$	$\epsilon_{k_i} = 0.010$	$\epsilon_{\mu_i} = 0.015$	$\epsilon_{\mu_i} = 0.010$
6	Points	200	677	213	435
	Max. Error	-1.60	-1.95	-1.99	-2.41
	Avg. Error	-2.70	-2.92	-2.41	-2.75
10	Points	384	2558	334	1996
	Max. Error	-1.60	-2.43	-1.99	-2.46
	Avg. Error	-2.70	-2.91	-2.70	-2.80

Adaptive Sparse grid puts resolution where needed!

ϵ : controls average error

max. Level: controls max. error

Conclusion & Outlook

- **Method perfectly suited to solve high dimensional dynamic models with large amount of heterogeneity!**
- First time adaptive sparse grids are applied to high-dimensional economic models
- First ones to solve dynamic models on HPC systems with hybrid parallelism
- Method also well-suited for Dynamic Programming (VFI).
- **Pull bigger wagon** → we showed that the kink model does favour adaptivity
→ solve models/ hard problems 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:

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

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

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

A II: 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!

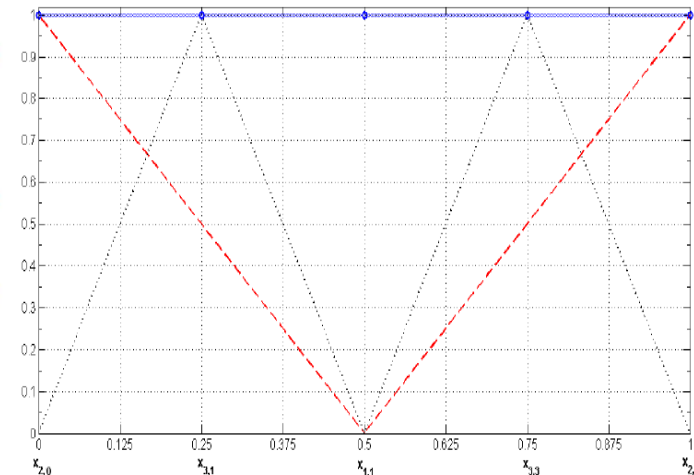
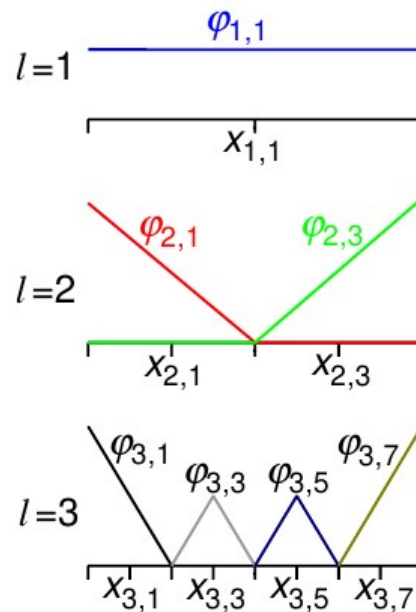


Fig.: Example of modified 1d-basis functions
According to Pflüger (2010), which are
extrapolating towards the boundary (**left**).
They are constant on level 1 and **“folded-up”**
if adjacent to the boundary on all other levels.
Right: **“Clenshaw-Curtis”** basis.