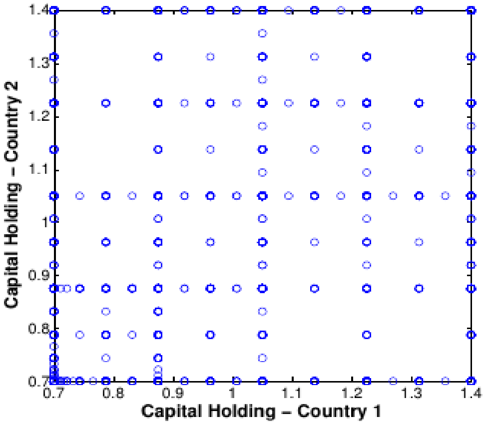
An Introduction to Sparse Grids and Adaptive Sparse Grids

Simon Scheidegger
simon.scheidegger@unil.ch
January 16th, 2019

Cowles Foundation – Yale University



Today – (Adaptive) Sparse Grids

I. Motivation – “the curse of dimensionality”

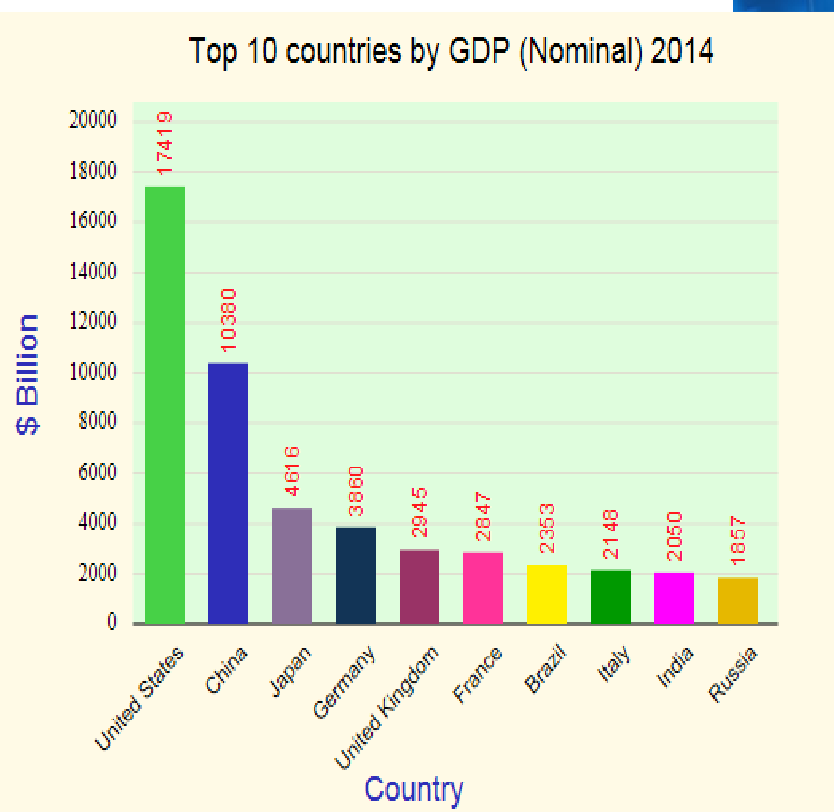
II. From Full (Cartesian) Grids to Sparse Grids

III. Adaptive Sparse Grids

IV. Gain hands-on experience with some libraries

Recall – 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**

Recall – Heterogeneity in OLG* models

*Overlapping generation models

To model e.g. social security:

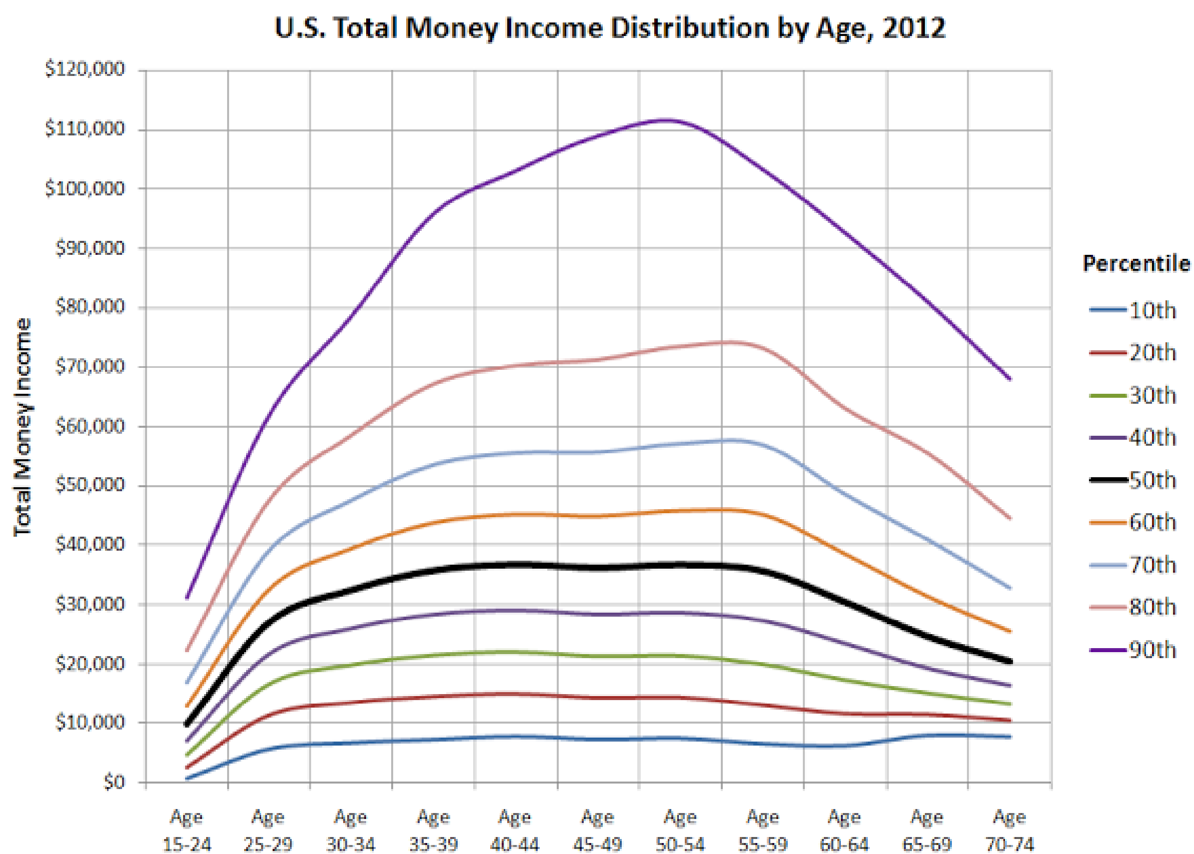
- How many age groups?

- borrowing constraints?

- aggregate shocks?

- …

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



<u>Recall – Financial markets: non-Gaussian returns</u>

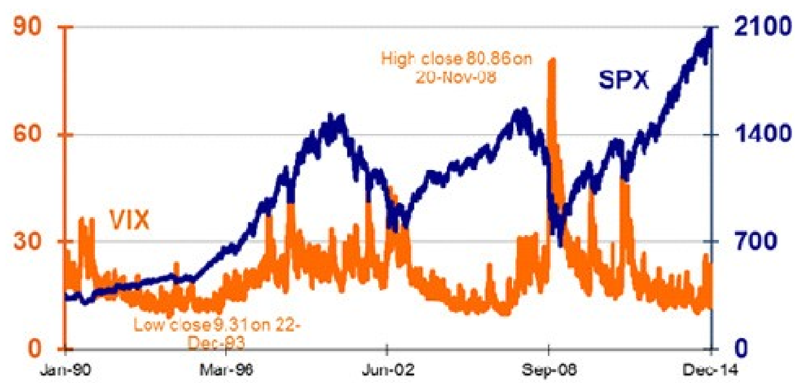
- Derivative contracts giving a <u>right to buy or sell an underlying security</u>.
    - *European* if exercise at expiration only.
    - ***American*** if exercise any time until expiration.

- 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 $j \longrightarrow \infty$ by iterations on:

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

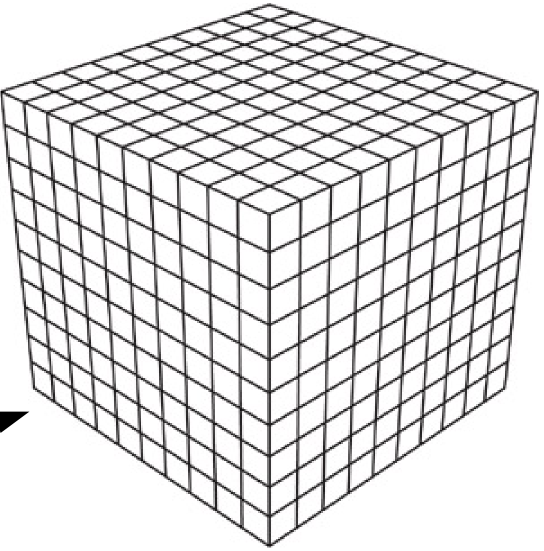
s.t.

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

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

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

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



→Use-case for (adaptive) sparse grids.

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,...

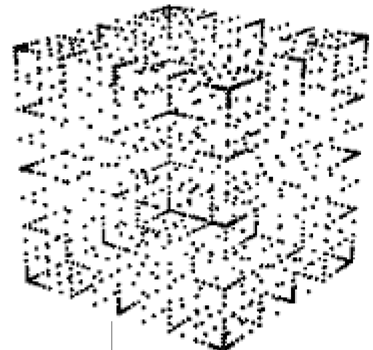
Deal with #Points
Adaptive Sparse Grids

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

→ **Focal point of todays and the next lecture**

Computational modelling

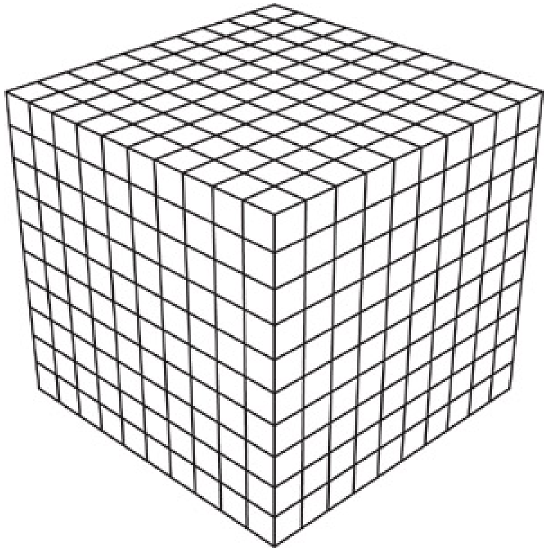
$$
\begin{aligned}
&\lambda_t \cdot \left[1 + \phi \cdot g^j_{t+1}\right] - \mu^j_t \\
&\quad - \beta \mathbb{E}_t \left\{ \lambda_{t+1} \left[a^j_{t+1} A \zeta (k^j_{t+1})^{\zeta - 1} + 1 - \delta + \frac{\phi}{2} g^j_{t+2} \left(g^j_{t+2} + 2 \right) \right] - (1 - \delta) \mu^j_{t+1} \right\} = 0,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



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**

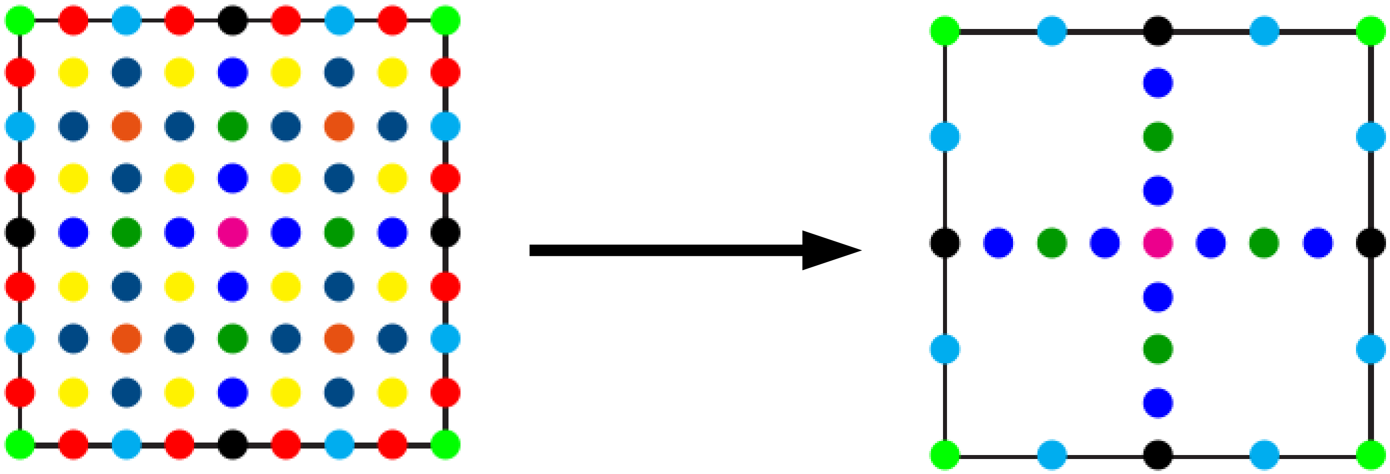
iv) **Want locality & adaptivity of interpolation scheme**

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

***see the lectures on parallel programming**

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:**

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

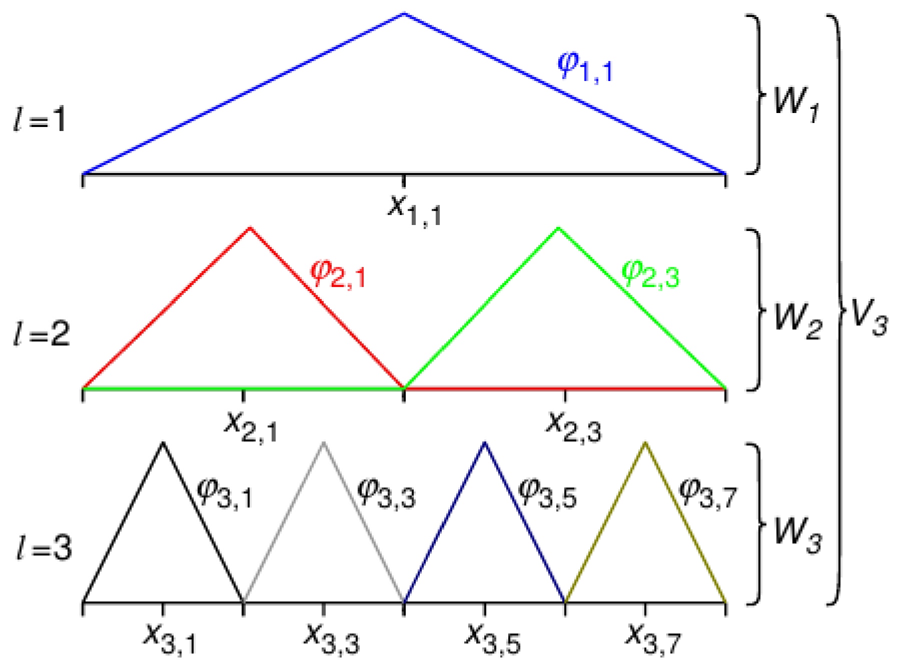


Fig.: 1-d basis functions $\phi_{l,i}$ and the corresponding **grid points** up level ***l* = 3** in the hierarchical basis.

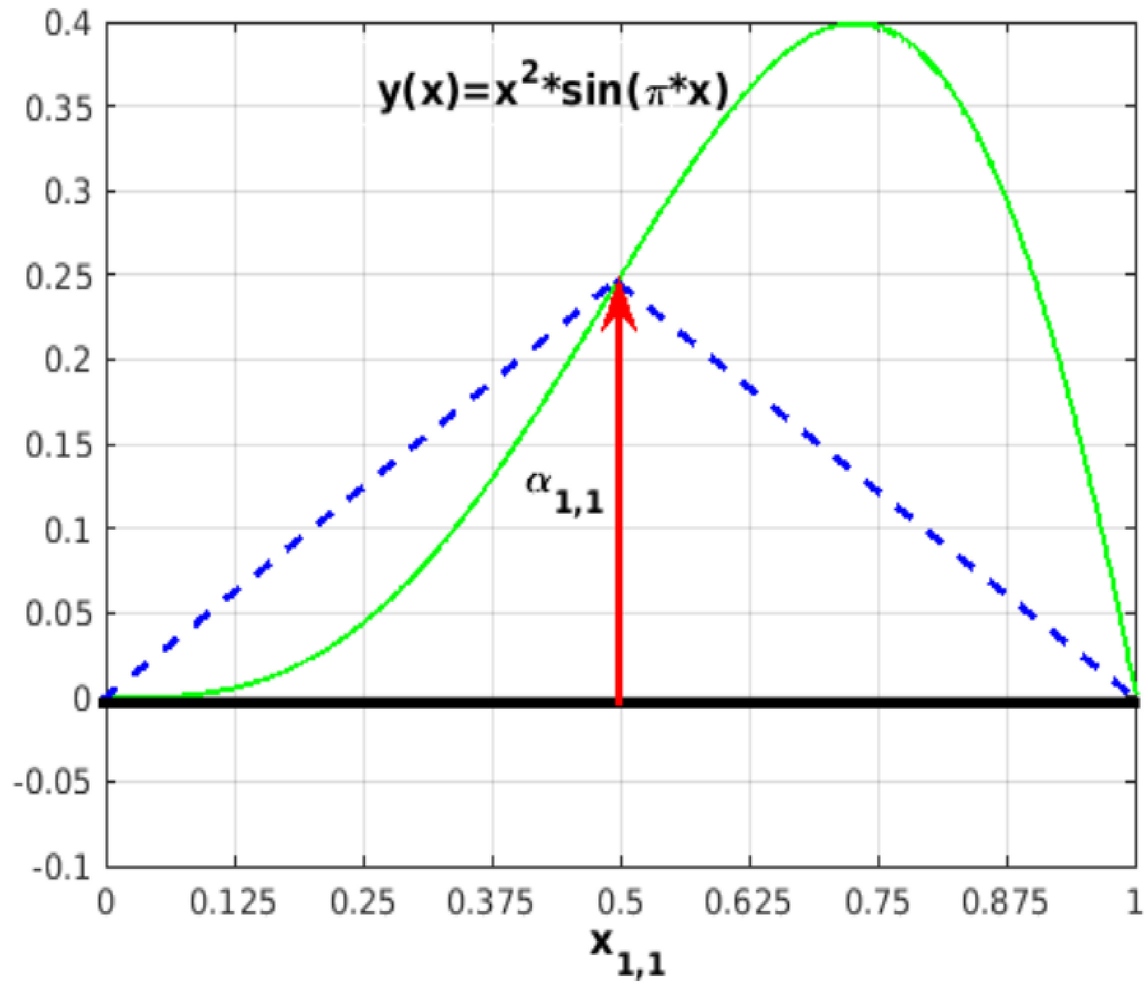
Note: supports of all basis functions of W_k mutually disjoint!

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 $f\left(\vec{x}_{l,i}\right)$

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

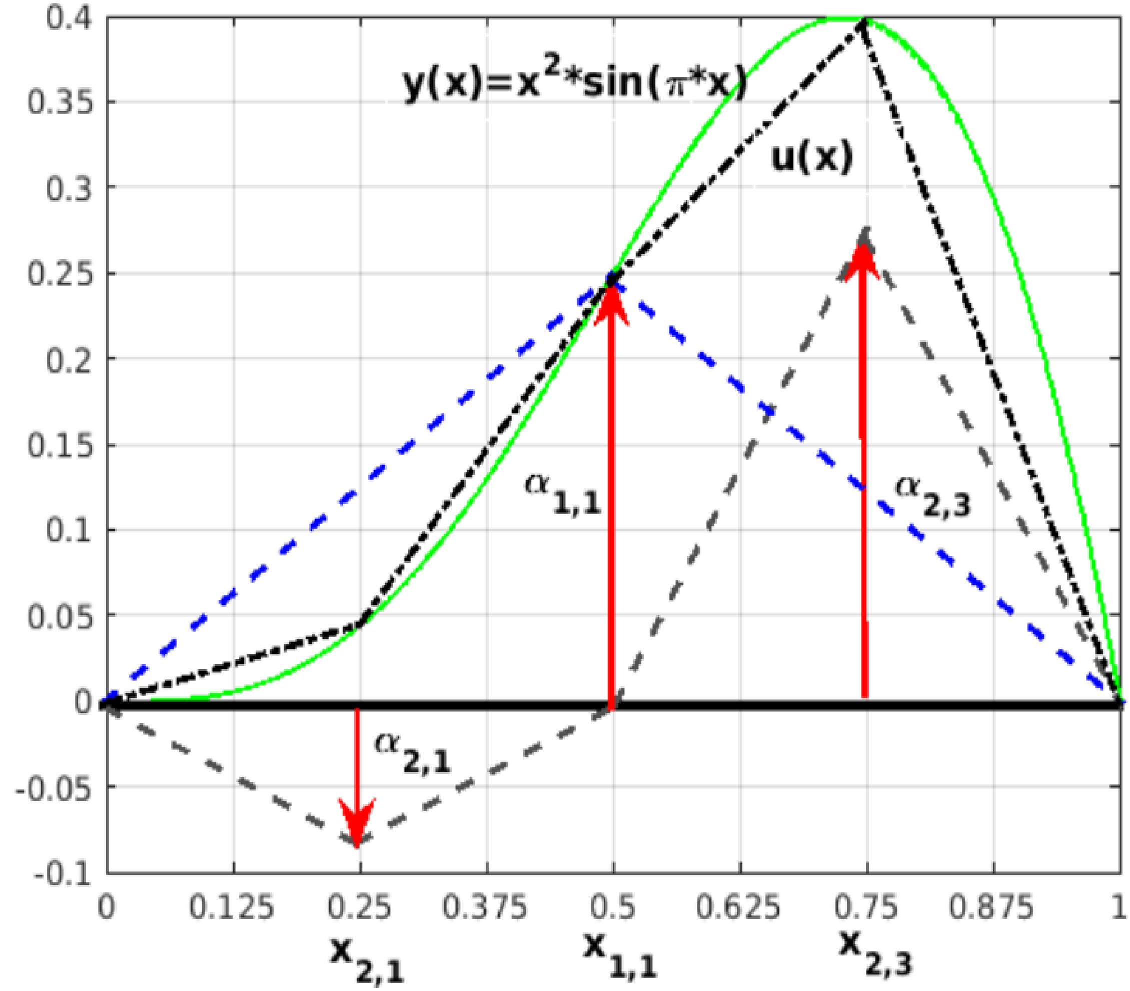


Piecewise Linear Interpolation: Level II

Coefficients:
hierarchical surpluses

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

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

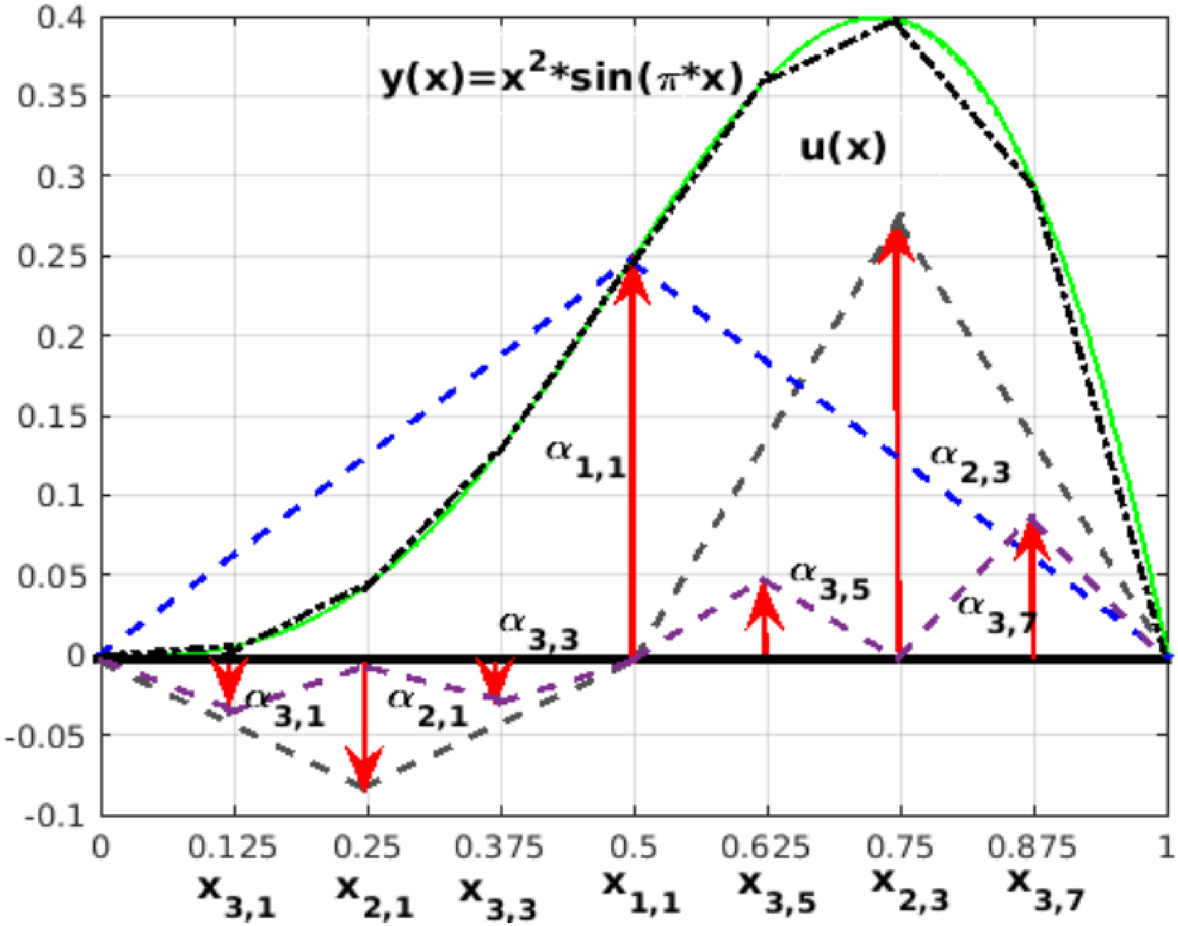


Piecewise Linear Interpolation: Level III

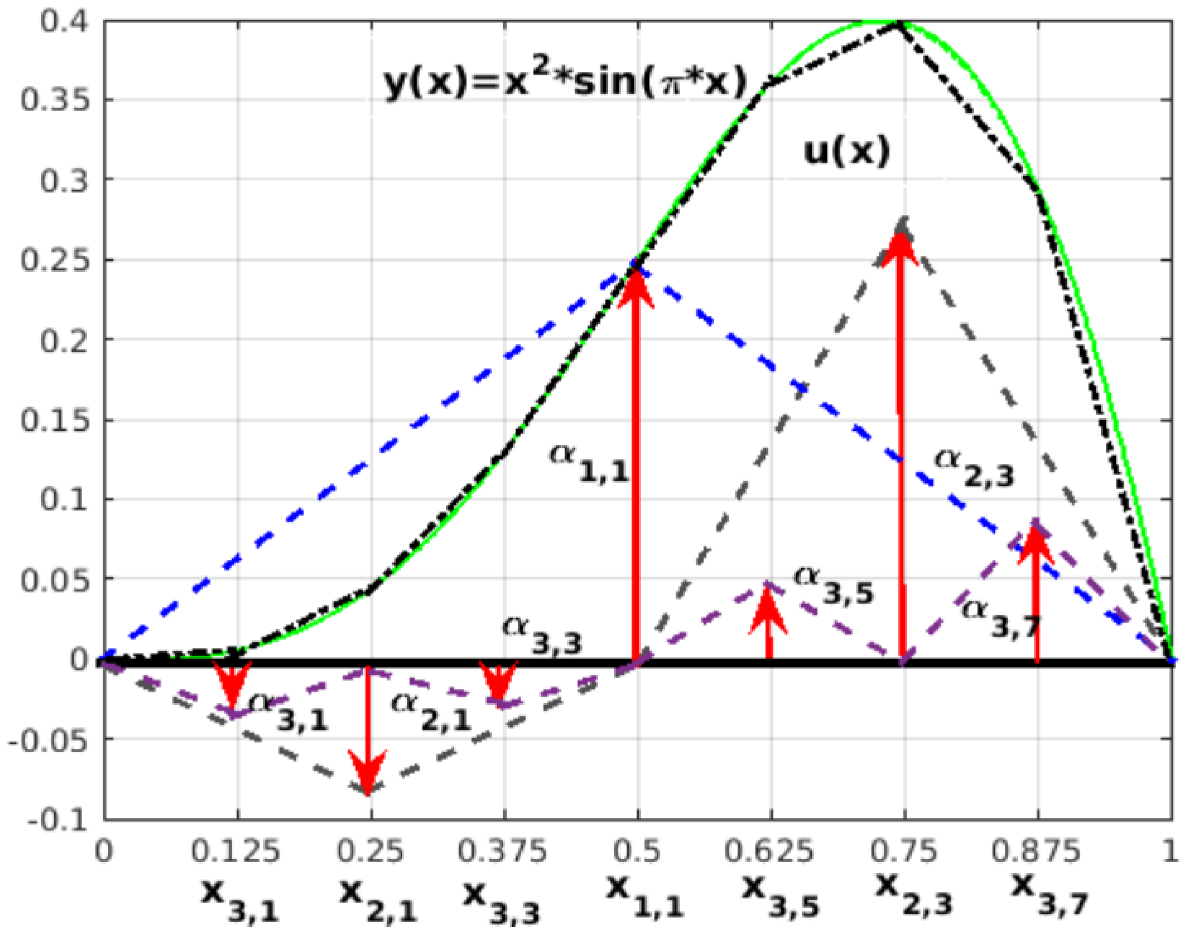
Coefficients:
hierarchical surpluses

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

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



MOVIE



y(x)=x^2*sin(π*x)
u(x)
α1,1
α2,3
α3,5
α3,7
α3,3
α3,1
α2,1
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
0.125
0.25
0.375
0.5
0.625
0.75
0.875
1
x3,1
x2,1
x3,3
x1,1
x3,5
x2,3
x3,7

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!

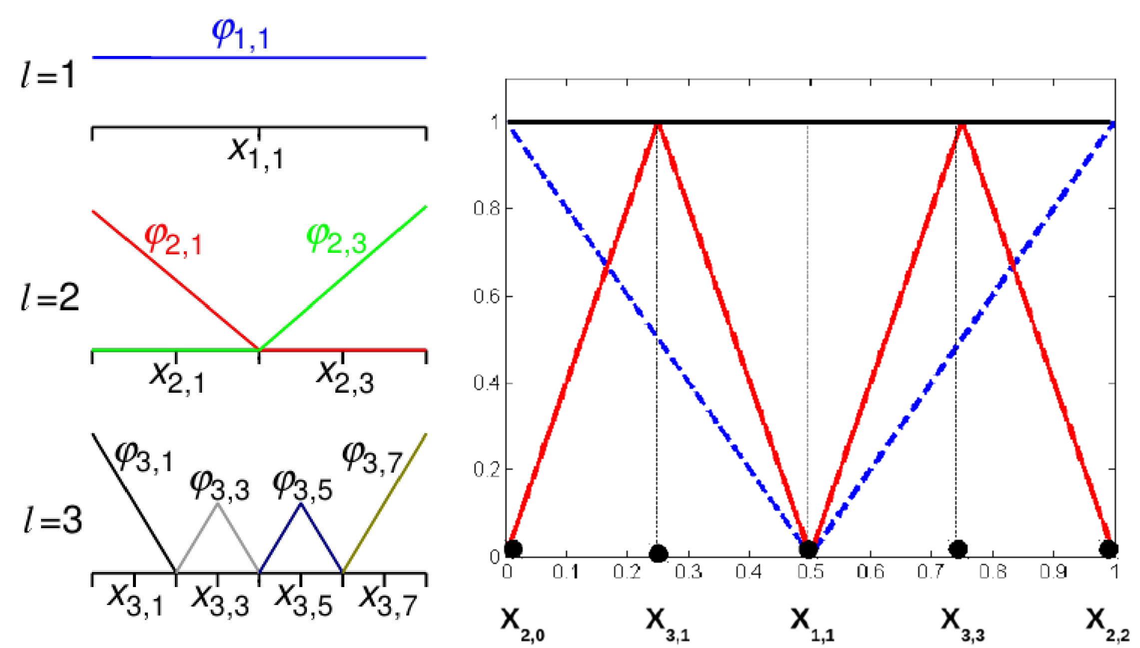


Fig.: Example of modified 1d-basis functions According to Pflüger (2010), which are extrapolating towards the boundary (**left**). They are constant on level 1 and **"folded-up"** if adjacent to the boundary on all other levels.
Right: "Modified" hat basis.

Examples for basis functions

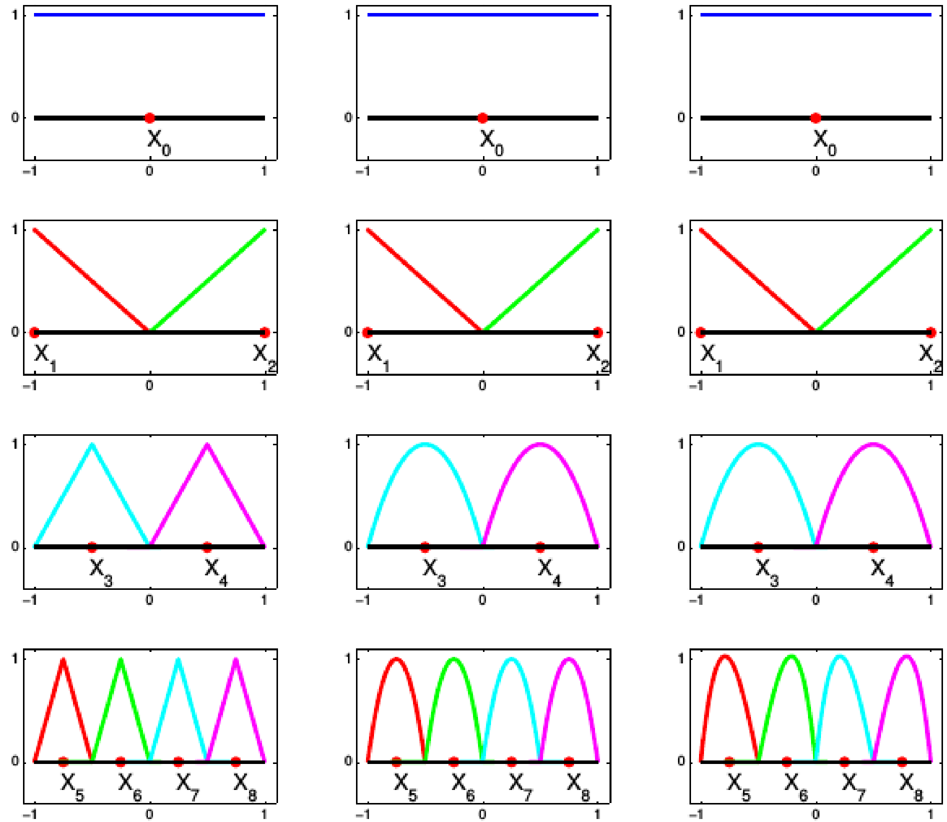


Figure 1: Local polynomial points (*rule_localp*) and functions, left to right: linear, quadratic, and cubic functions.

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: $\boxed{\vec{x}_{\vec{l},\vec{i}} := (x_{l_1,i_1}, \ldots, x_{l_d,i_d})}$

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

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

Multi-d interpolant:

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

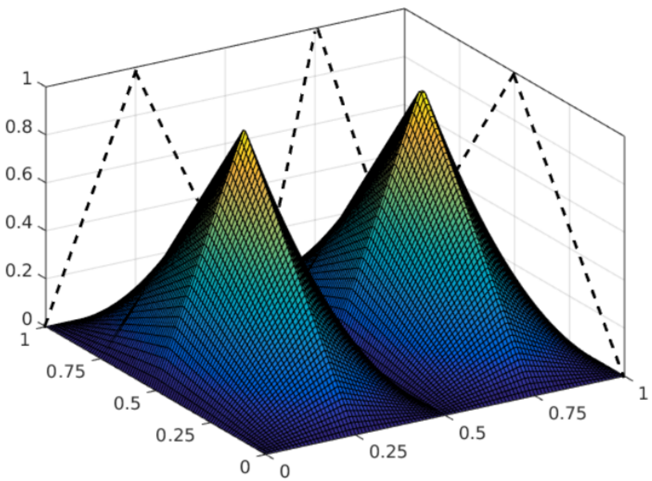


Fig.: Basis functions of the **subspace $W_{2,1}$**

Why reality bites...

Interpolant consists of ***O(2nd)*** 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, $\mathbf{5.8 \times 10^{11}}$ 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***.

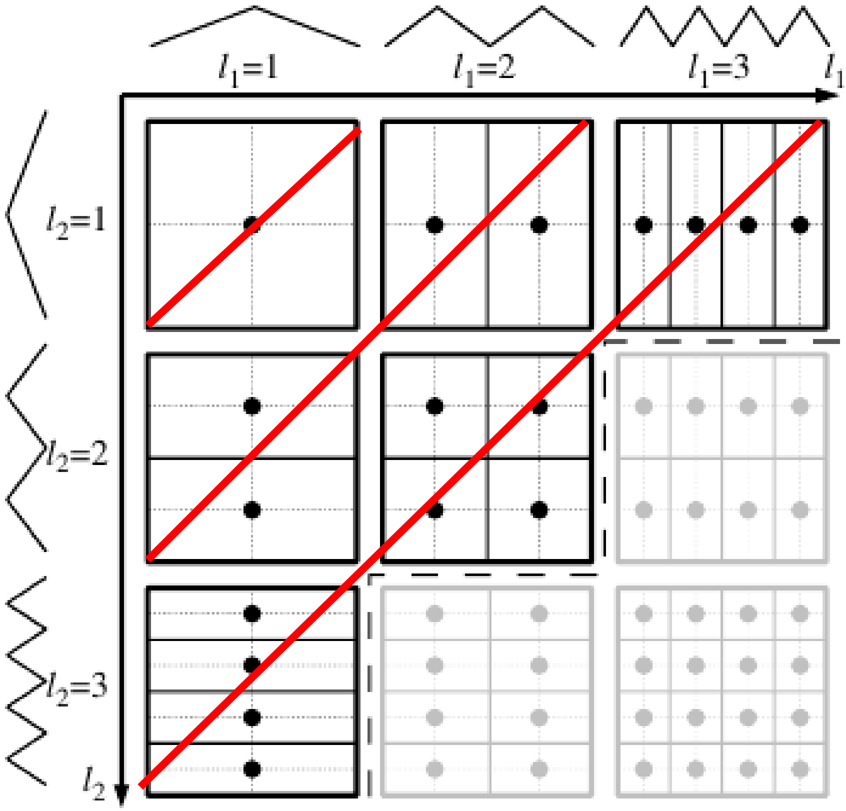
$$V_{0,n}^S := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}.$$

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

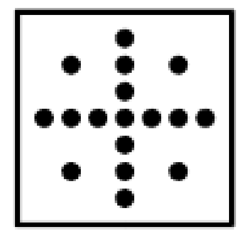
grid points: $\mathcal{O}\left(h_n^{-1} \cdot \left(\log(h_n^{-1})\right)^{d-1}\right) = \mathcal{O}\left(2^n \cdot n^{d-1}\right)$ << $\mathcal{O}\left(h_n^{-d}\right) = \mathcal{O}\left(2^{nd}\right)$

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



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



V_3

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.

Formal Details of Sparse Grids

Proposition 1: *The number of grid points in the sparse grid space V_n^S is given by*

$$|V_n^S| = \sum_{i=0}^{n-1} 2^i \binom{d-1+i}{d-1} = 2^n \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2}) \right).$$

For the proof of this proposition see Lemma 3.6 of [1].
Note that Proposition 1 directly implies that

$$|V_n^S| = \mathcal{O}(2^n \cdot n^{d-1}).$$

[1] H.-J. Bungartz and M. Griebel. Sparse grids. Acta Numerica, 13:1–123, 2004.

Formal Details of Sparse Grids (II)

- As mentioned before, SGs arise from a "cost-benefit" consideration:
  → find the approximation space

$$V^{opt} \subset V := \bigcup_{n=1}^{\infty} V_n$$

  that provides the highest accuracy for a given number of grid points.

- Clearly, the answer to this question depends on the class of functions we would like to interpolate.

- The theory of SGs considers the Sobolev space of functions with bounded second-order mixed derivatives:

$$H_2(\Omega) := \{f : \Omega \to \mathbb{R} : D^{\vec{l}} f \in L_2(\Omega), |\vec{l}|_\infty \leq 2, f|_{\partial\Omega} = 0\},$$

where $$D^{\vec{l}} f := \frac{\partial^{|\vec{l}|_1}}{\partial x_1^{l_1} \cdots \partial x_d^{l_d}} f.$$

Formal Details of Sparse Grids (III)

- V^{opt} depends on the norm $||\cdot||$ in which the interpolation error is measured, and also on the semi-norm used to measure the "variability" of the functions to be interpolated.

- Proposition 2, below, holds for the L_2 and L_∞ norms as well as for the two seminorms

$$|f|_{\alpha,2} := \left(\int_\Omega |D^\zeta f|^2 d\vec{x}\right)^{\frac{1}{2}}, \quad |f|_{\alpha,\infty} := ||D^\alpha f||_\infty$$

with α = 2.

Formal Details of Sparse Grids (IV)

- In order to leverage on the hierarchical setting introduced, we only allow discrete spaces of the type

$$U_{\vec{I}} := \bigoplus_{\vec{l} \subset \vec{I}} W_{\vec{l}}$$

- for an arbitrary index set $\vec{I}$ as candidates for the optimization process.

- We use $f_{U_{\vec{I}}} \in U_{\vec{I}}$ to denote the interpolant of f from the approximation space $U_{\vec{I}}$.

Formal Details of Sparse Grids (V)

We are now in the position to state precisely in which sense “classical” SGs are optimal.

Proposition 2: *The sparse grid approximation space*

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

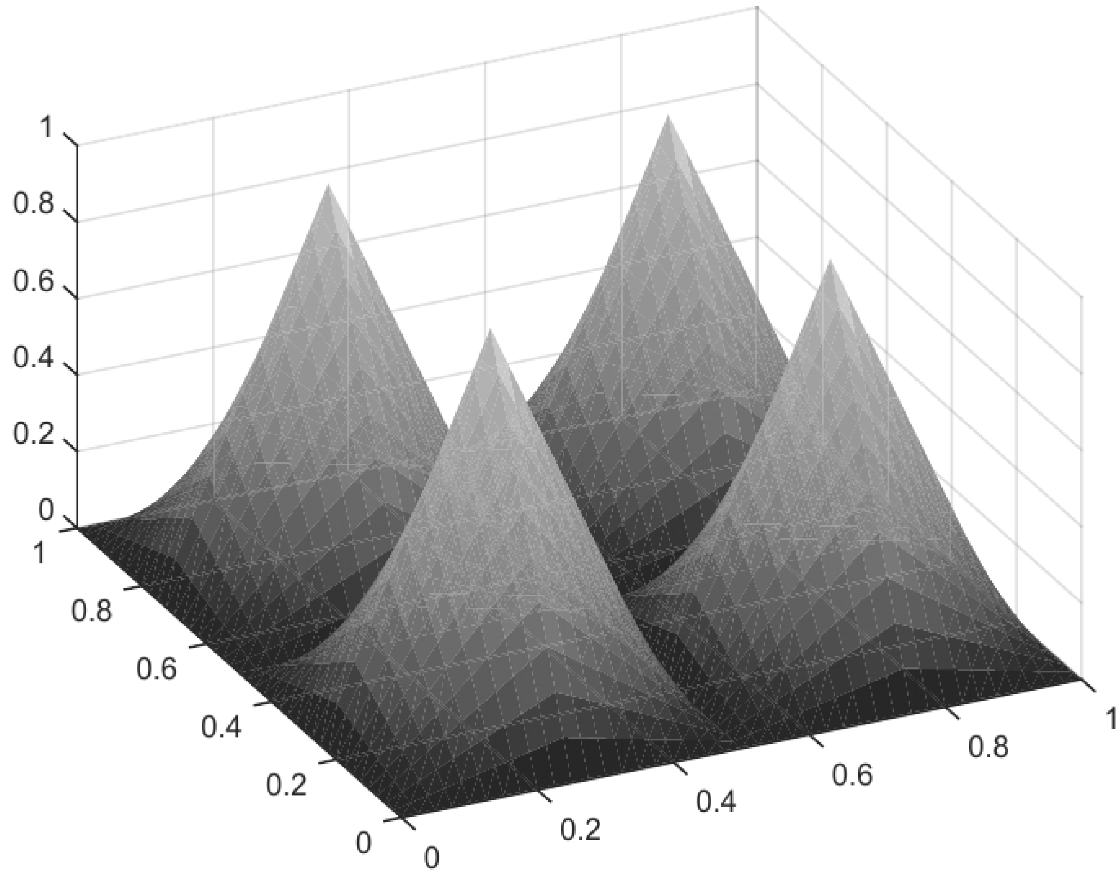
is the solution to the constrained optimization problem

$$\min_{U_{\vec{I}} \subset V : |U_{\vec{I}}| \leq |V_n^S|} \quad \max_{f \in H_2(\Omega) : |f| = 1} \|f - u_{U_{\vec{I}}}\|.$$

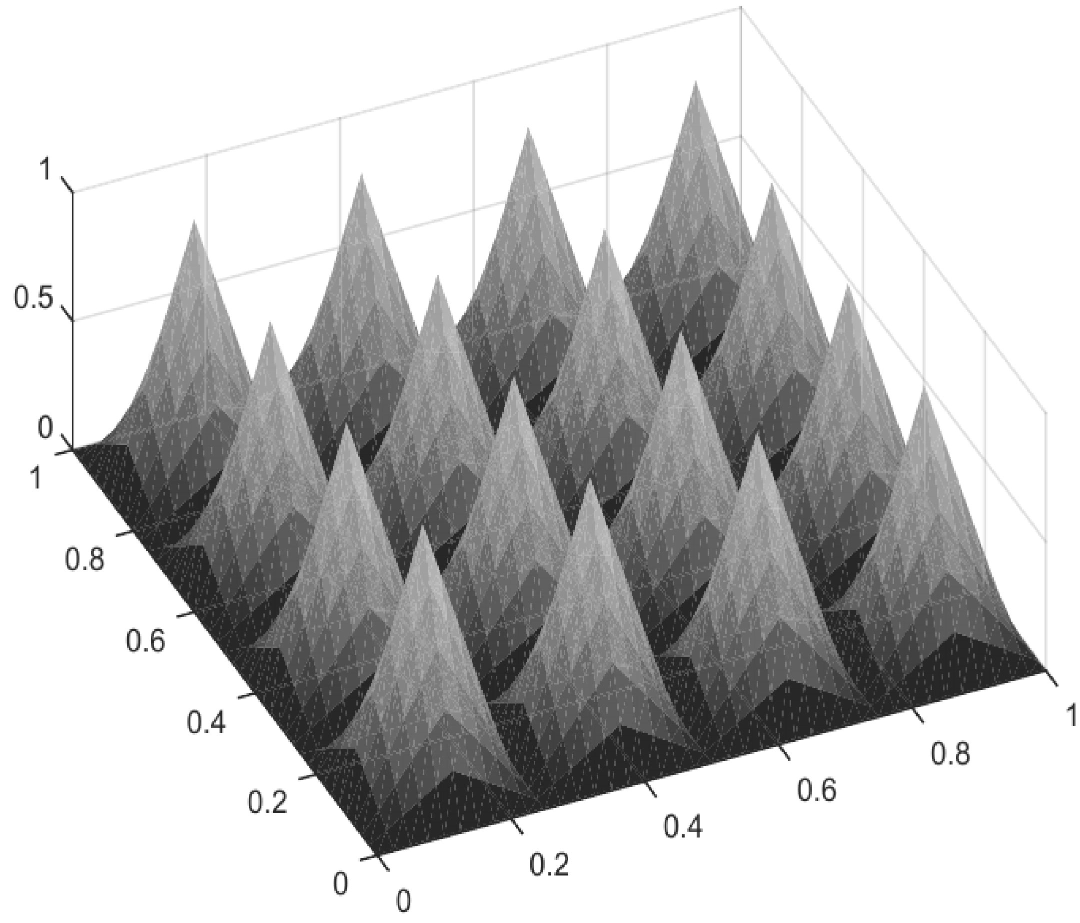
Interpretation of Proposition 2

- In words, the sparse grid V_n^S minimizes among all approximation spaces $U_{\vec{I}}$ which do not have more grid points $(|U_{\vec{I}}| \leq |V_n^S|)$ the maximal approximation error reached when interpolating functions with bounded second-order mixed derivatives ($f \in H_2(\Omega)$) and a given variability.
- Finally, note that “classical” SGs are also the solution to the reverse optimization problem of achieving some desired accuracy with the smallest possible number of grid points.

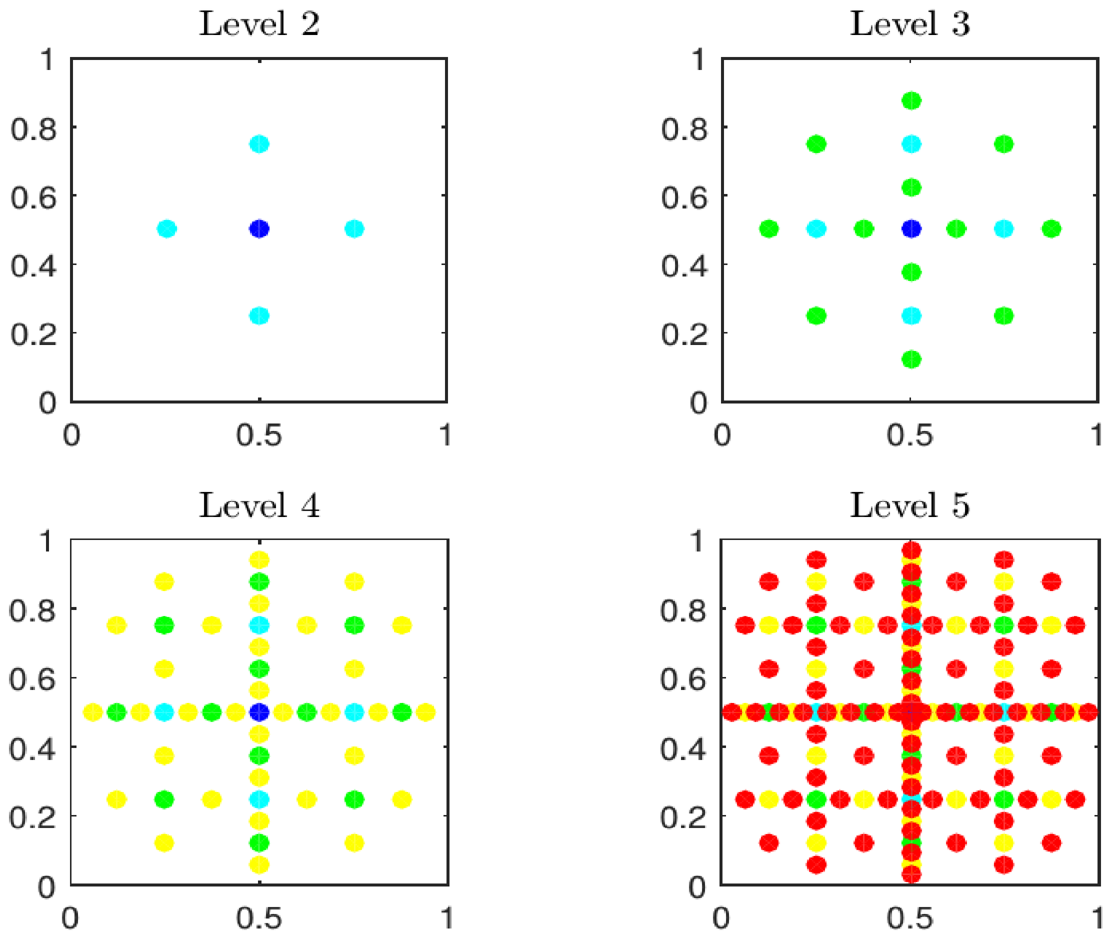
Basis Functions of $W_{2,2}$ — Included in V_3



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

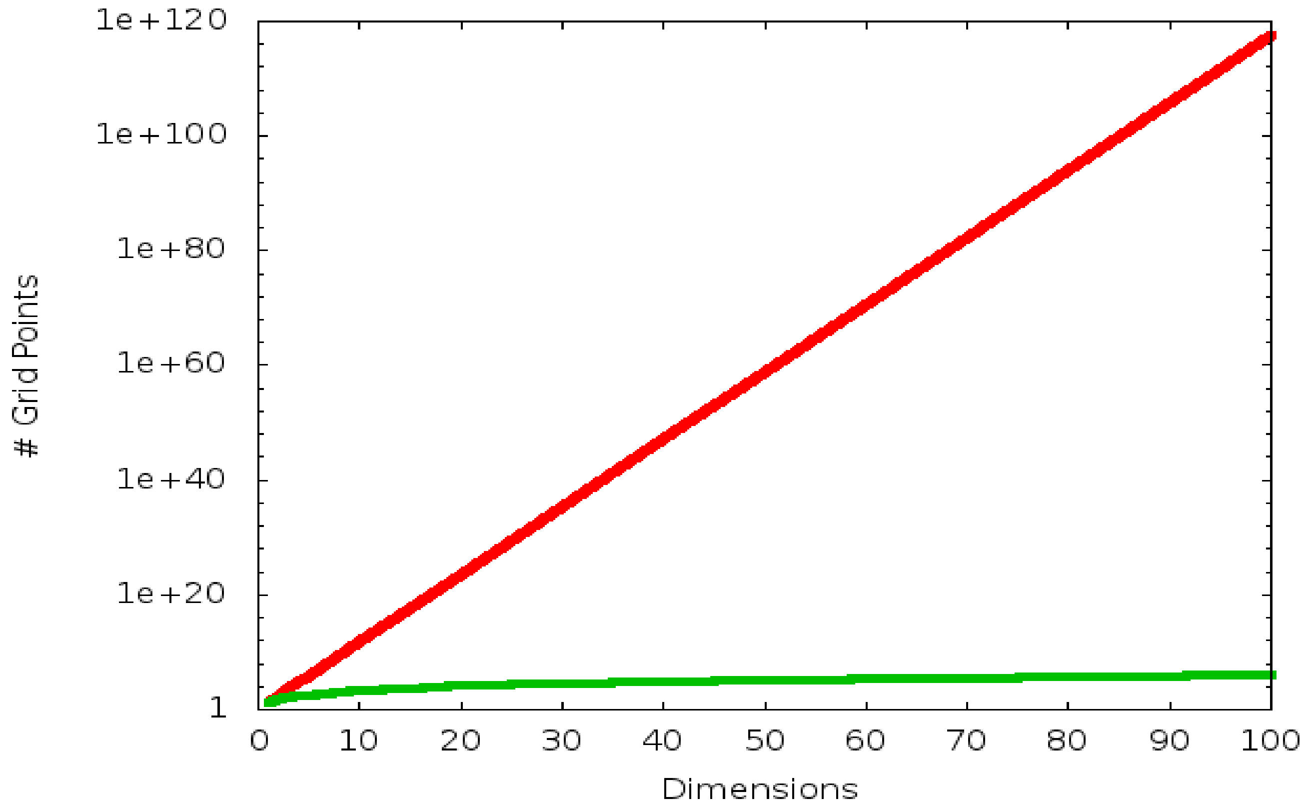


Sparse Grid of Increasing level



Grid Points

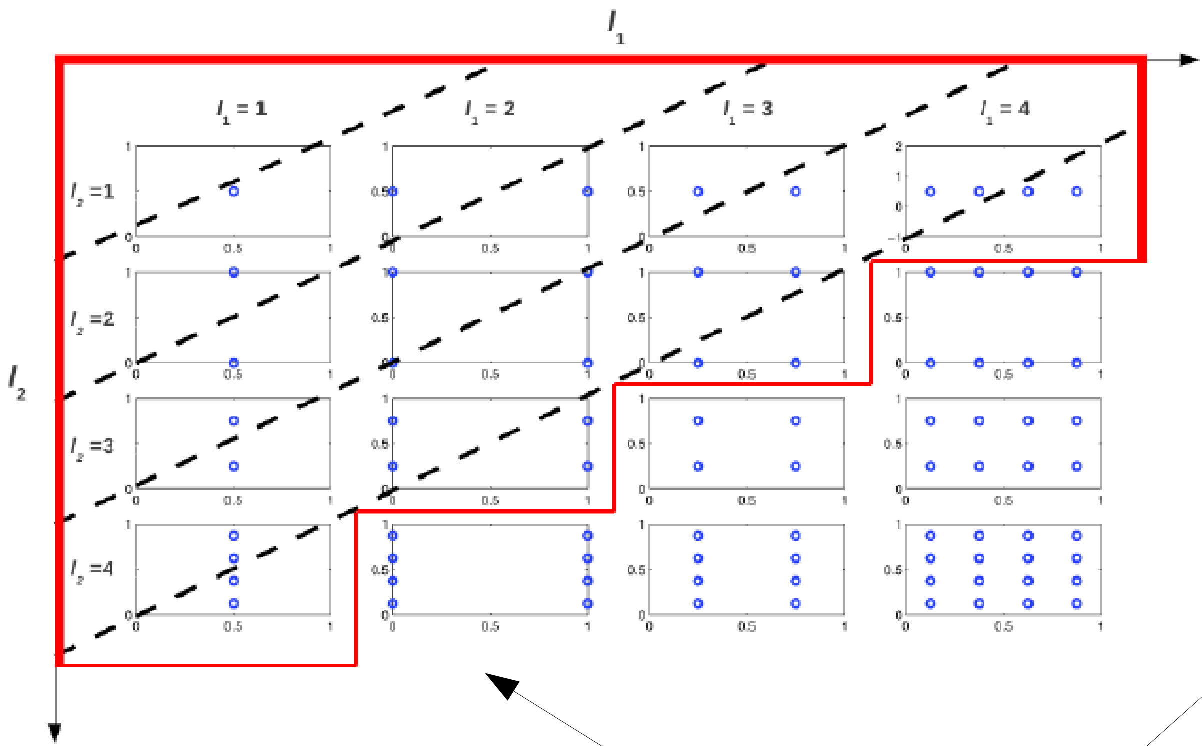
d	$\|V_n\|$	$\|V_{0,n}^S\|$
1	15	15
2	225	49
3	3375	111
4	50'625	209
5	759'375	351
10	$5.77 \cdot 10^{11}$	2'001
15	$4.37 \cdot 10^{17}$	5'951
20	$3.33 \cdot 10^{23}$	13'201
30	$1.92 \cdot 10^{35}$	41'601
40	$1.11 \cdot 10^{47}$	95'201
50	$6.38 \cdot 10^{58}$	182'001
100	>Googol	1'394'001

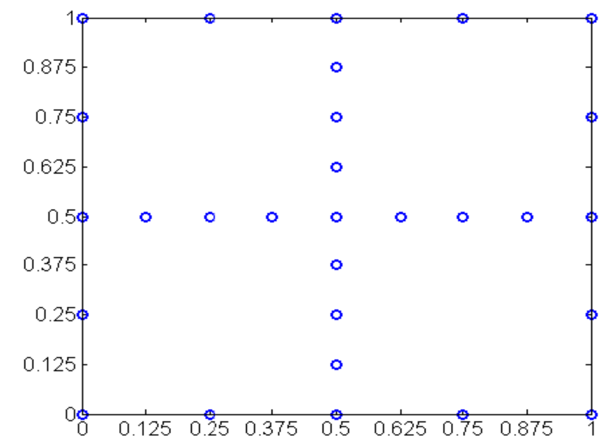


Tab.: Number of grid points for several types of sparse grids of level *n* = 4.
Middle: Full grid; **right:** **classical sparse grid** with **no points at the boundaries.**
Fig.: Number of grid points growing with dimension (full grid vs. sparse grid).

Sparse Grid with non-zero boundaries

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





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

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

Hierarchical Integration

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

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

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

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

where were 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),...

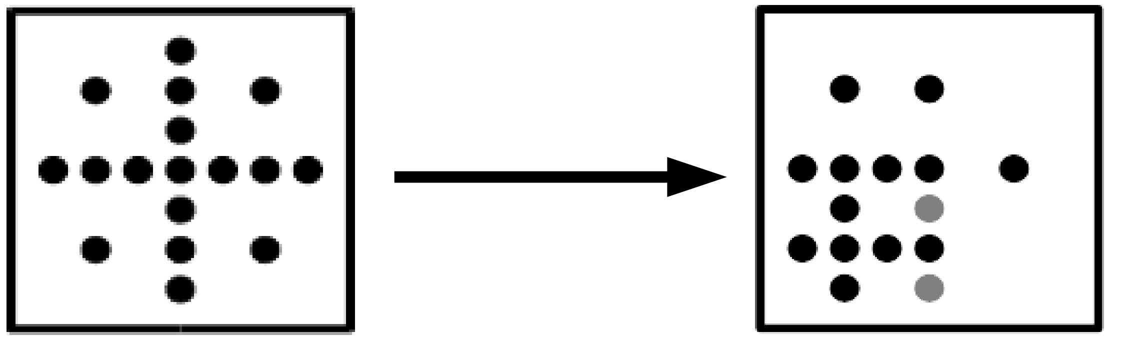
-Solution of PDEs

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

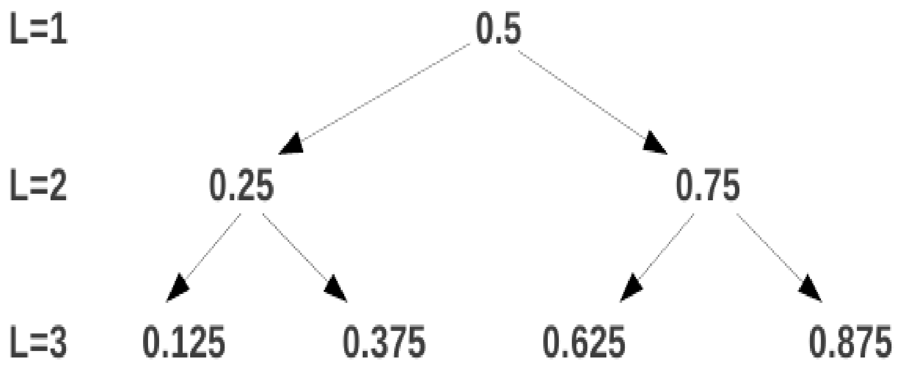
-**Use hierarchical surplus as error indicator**.

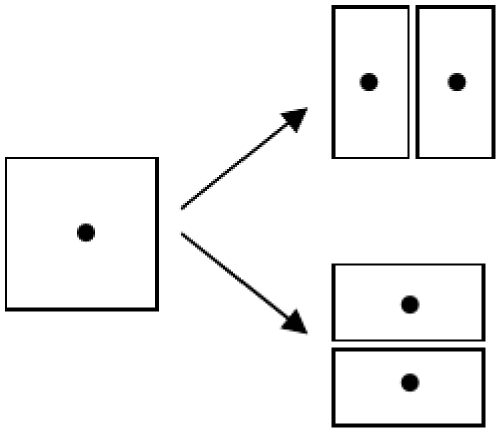
-**Automatically detect "discontinuity regions"** and adaptively refine the points in this region.

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

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

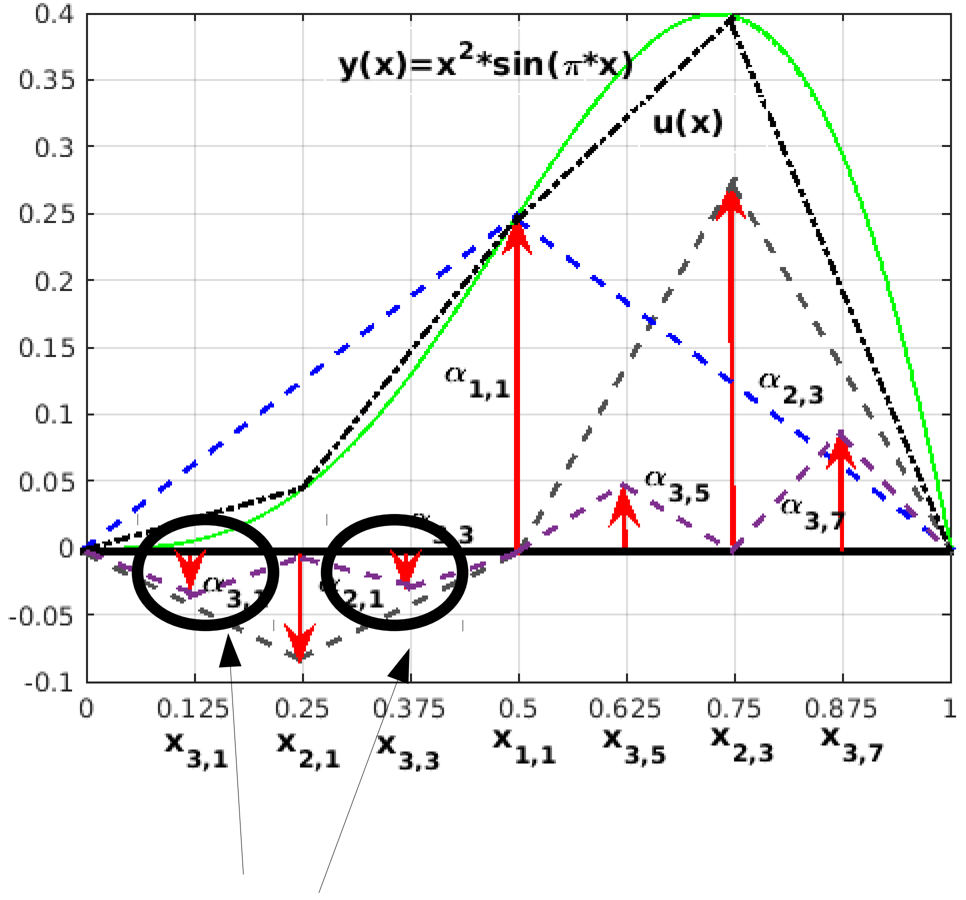
-Criterion: **e.g.** $|\alpha_{\vec{l},\vec{i}}| \geq \epsilon$





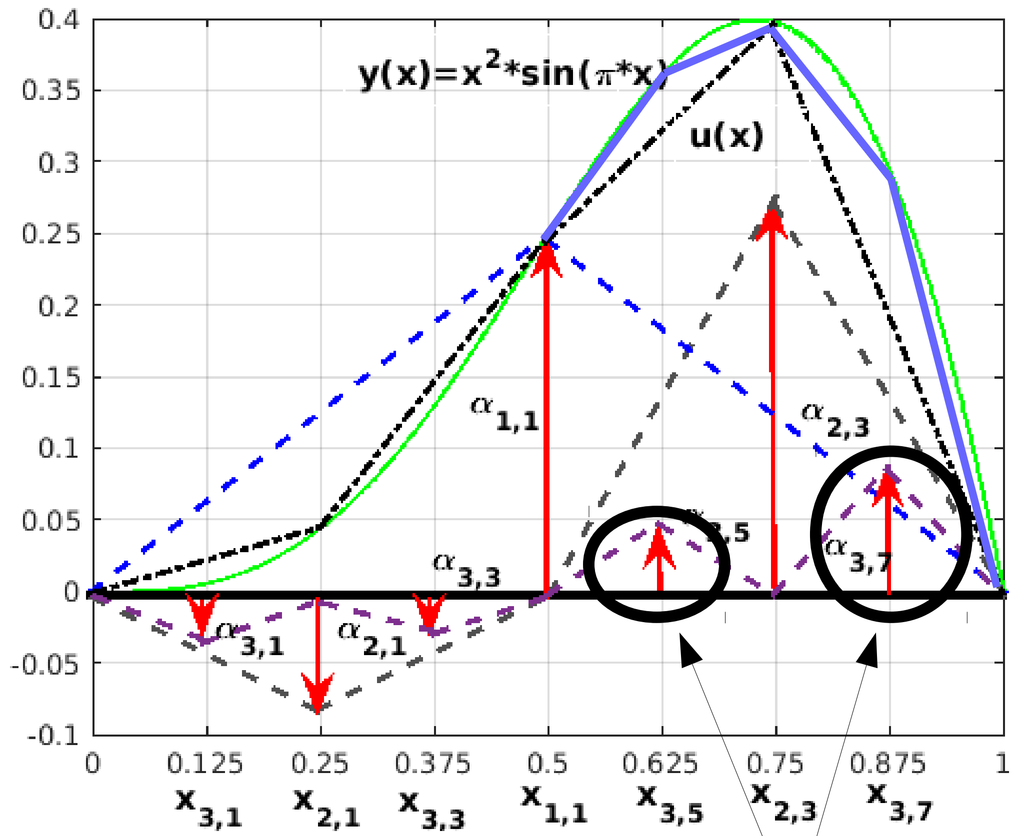
top panel: tree-like structure of sparse grid.
lower panel: locally refined sparse grid in 2D.

Example I



Small – below threshold

Example II



Add points – above threshold

Test in 1d

(See Genz (1984) for test functions)

Test function:

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

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

Error measure:

→ 1000 random points from [0,1]

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

Full grid: **1023** points

Adaptive sparse grid: **109** points.

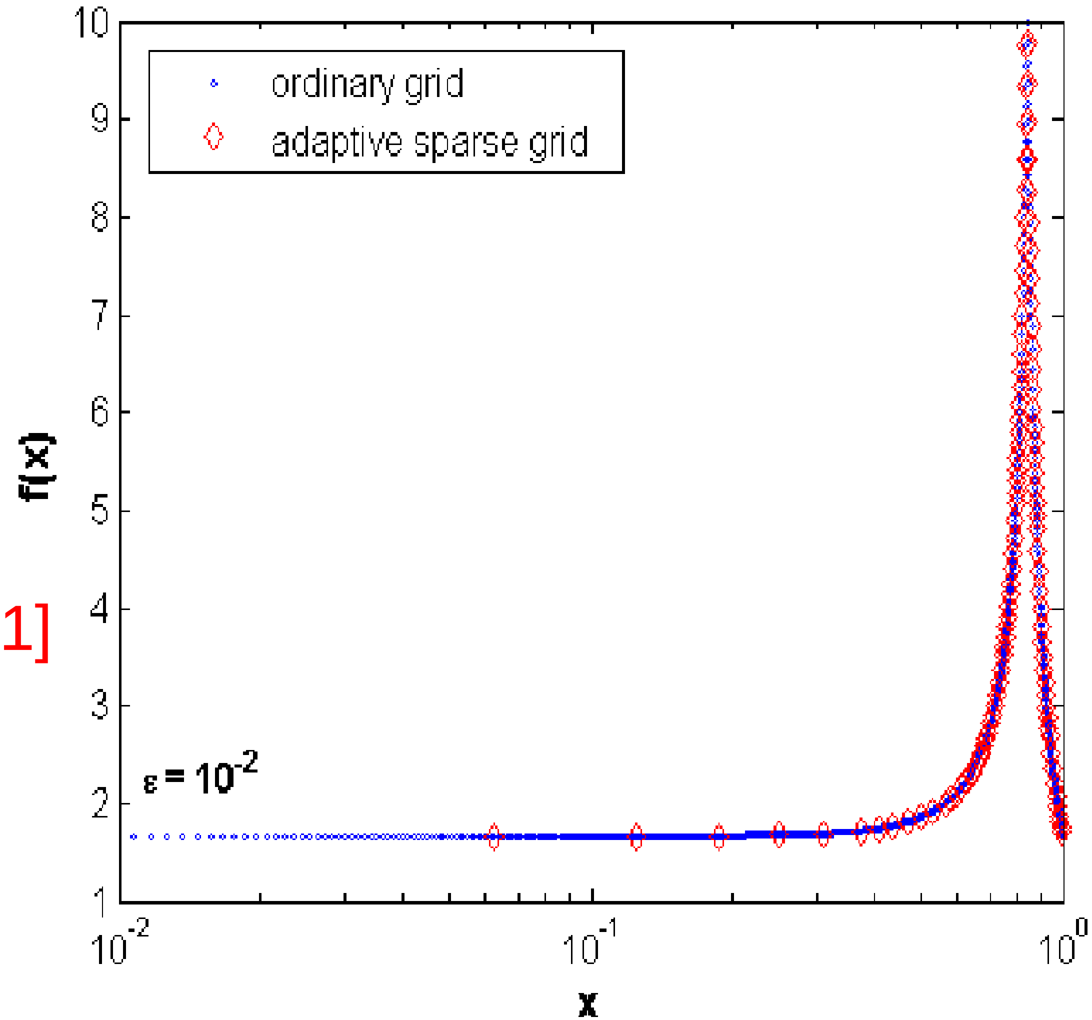


Fig.: Blue: Full grid;red: adaptive sparse grid.

Test in 2d

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

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

Full grid:
→ $O(10^9)$ points

Sparse grid:
→ **311,297 points**

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

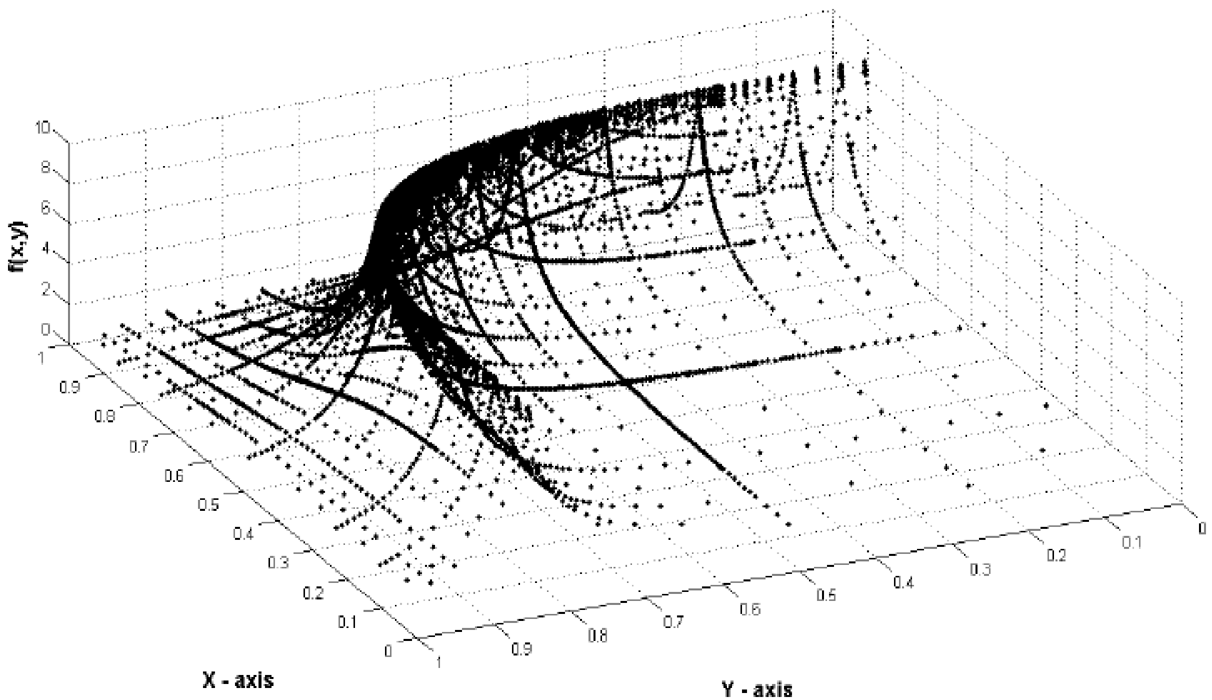


Fig.: 2d test function and its corresponding grid points after 15 refinement steps.

Movie

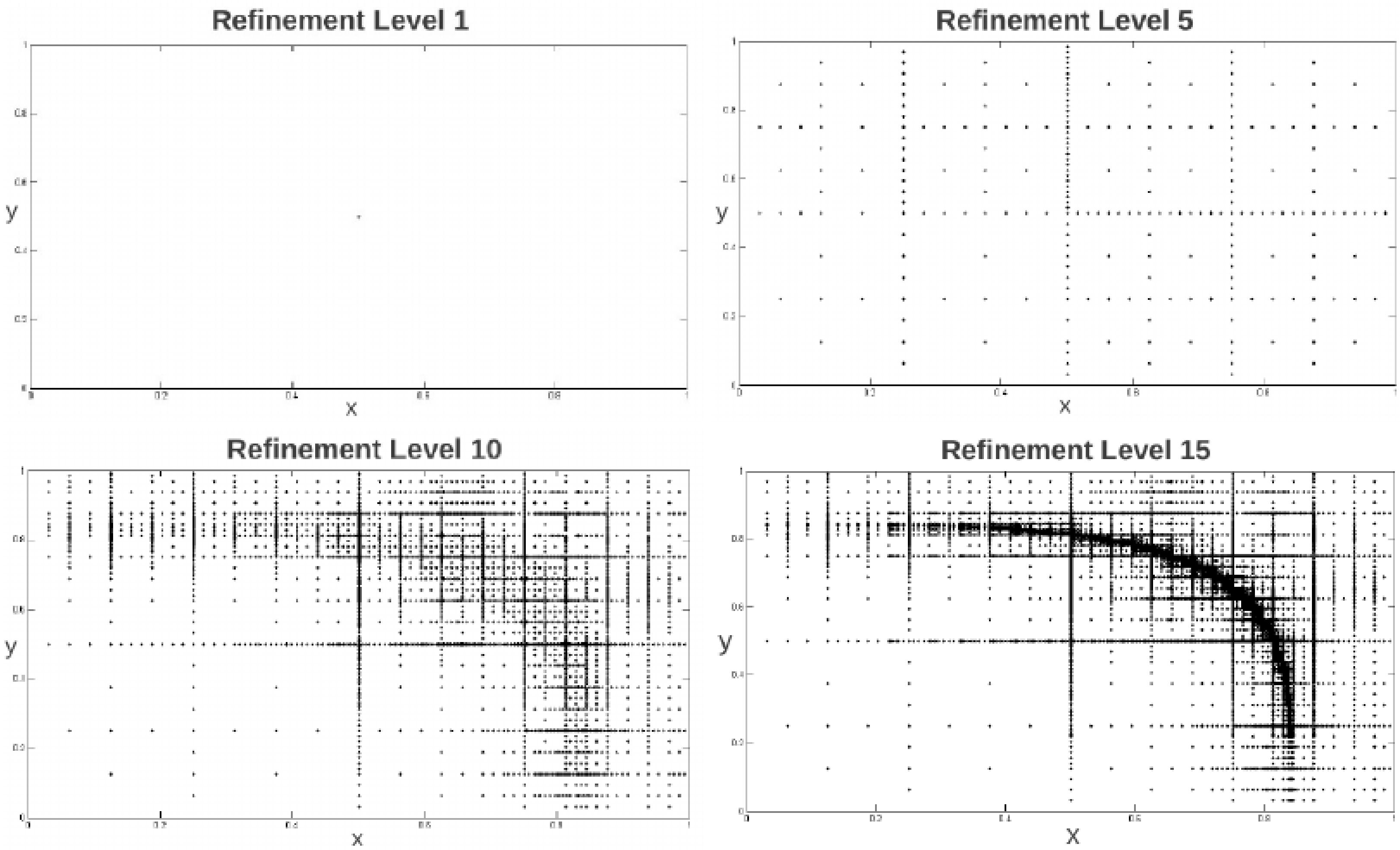
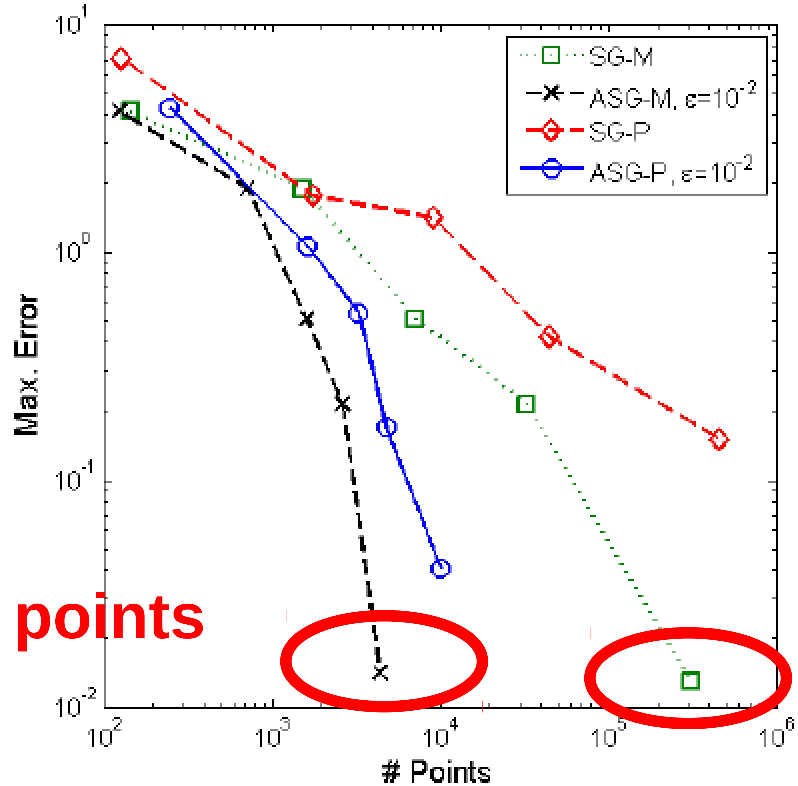


Fig.: Evolution of the adaptive sparse grid with a **threshold for refinement of 10^{-2}** . The refinement levels displayed are L = 1, 5, 10, 15.

Convergence

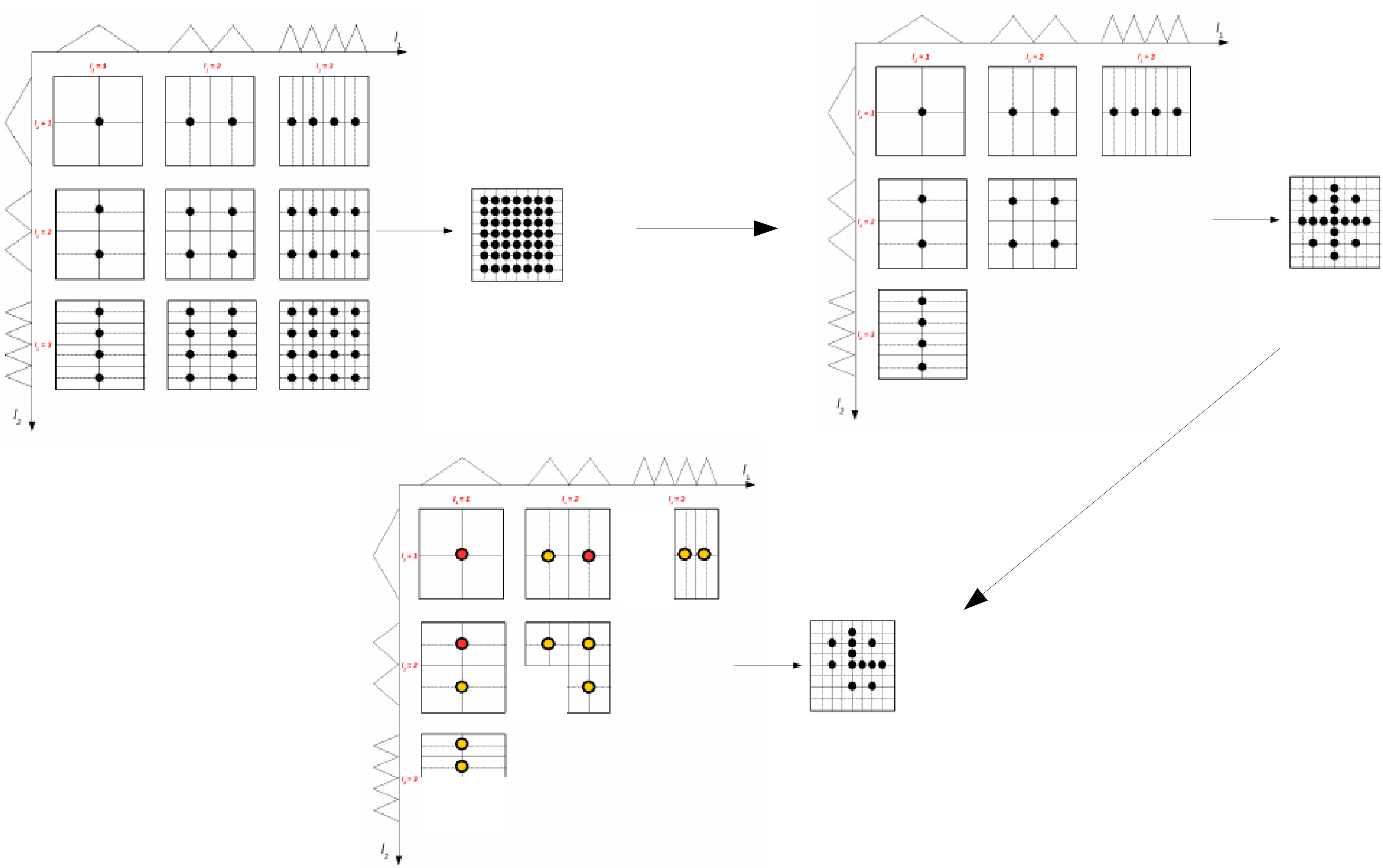


4,411 points

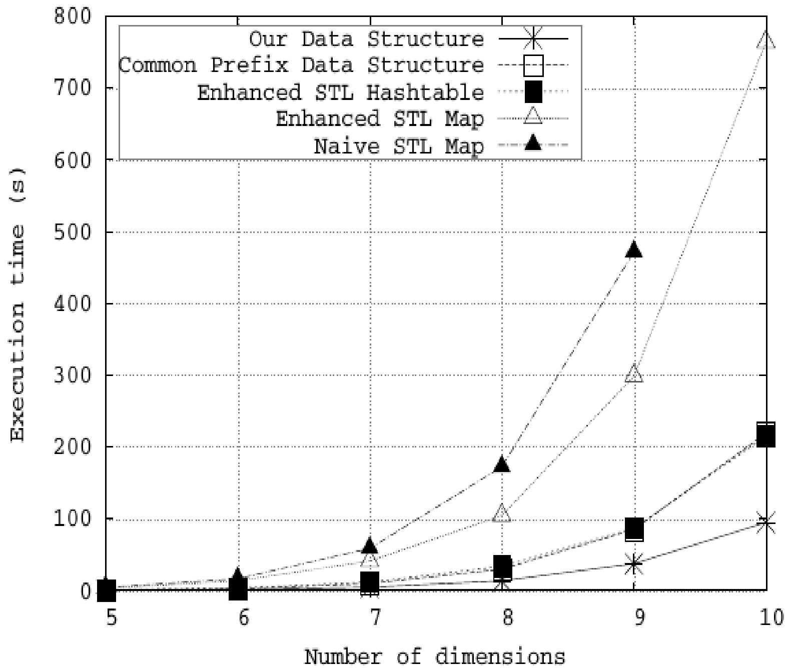
311,297 points

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

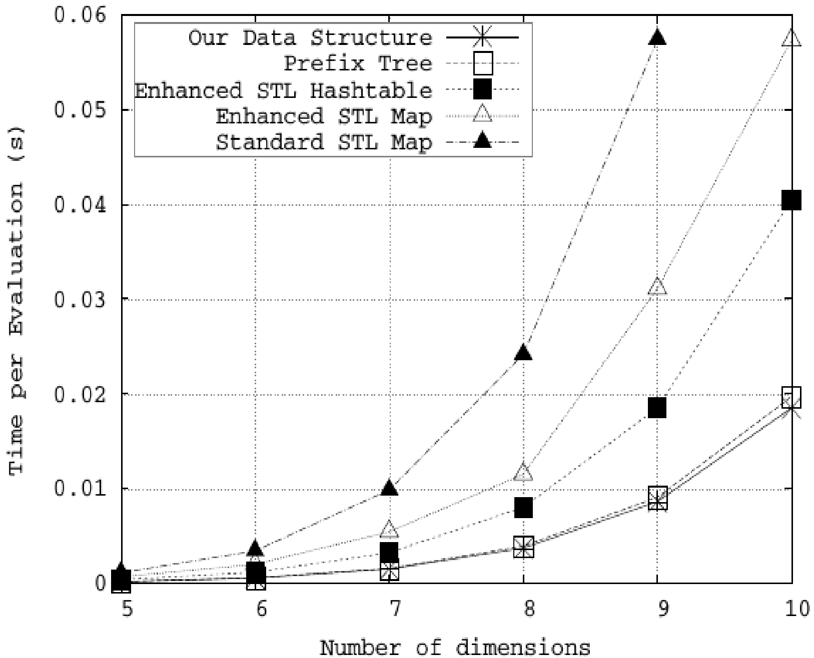
From Cartesian to adaptive sparse grids



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)

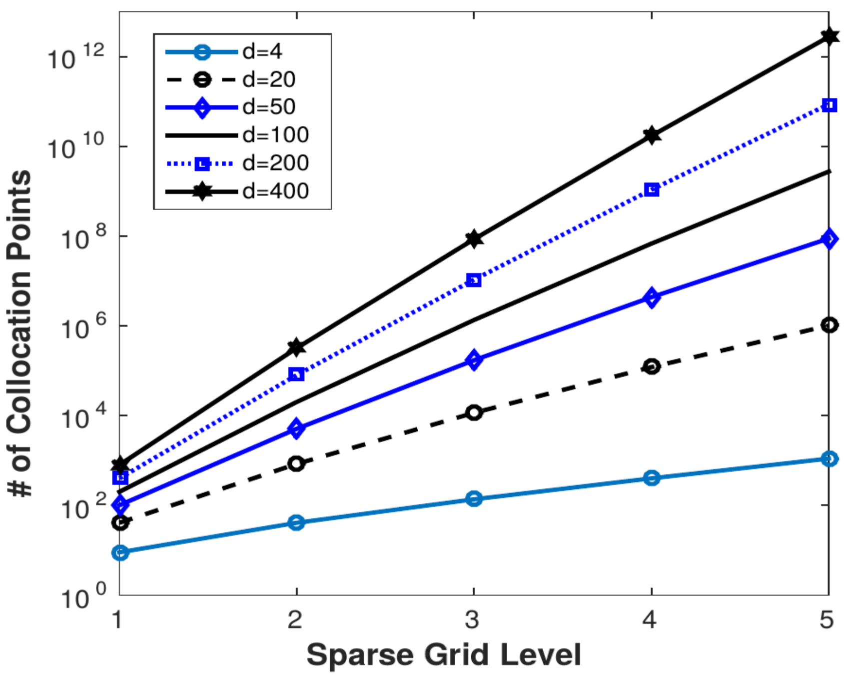


Fig.: classical sparse grids of varying dimension and increasing refinement level

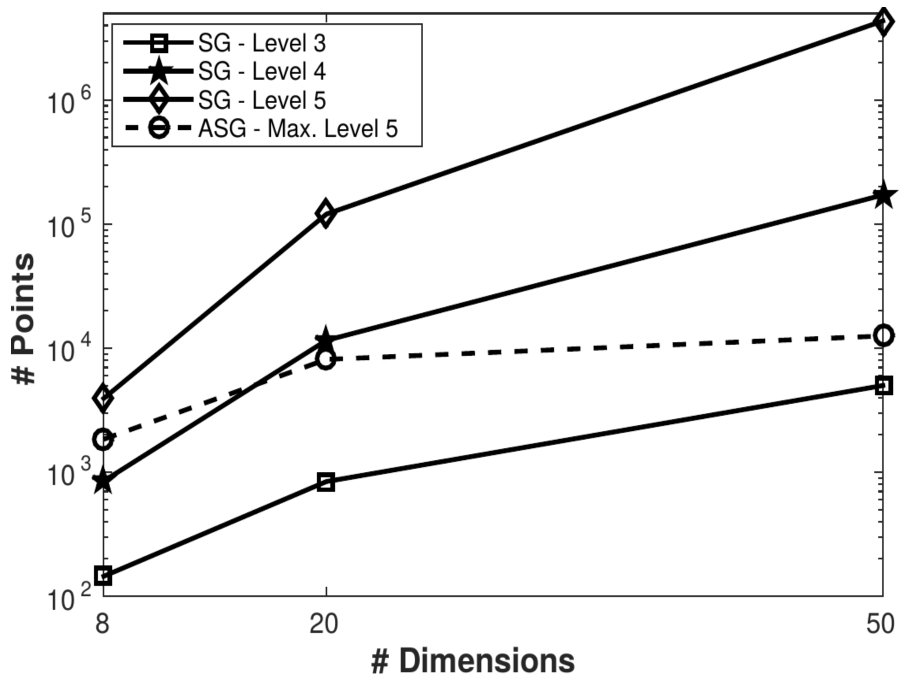


Fig.: IRBC model, solved both with classical sparse grids of varying dimension and increasing refinement level.

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)**.

Note – Install all libraries

- Note: I prepared you the packages with all the dependencies.

- in order to install, follow these steps:

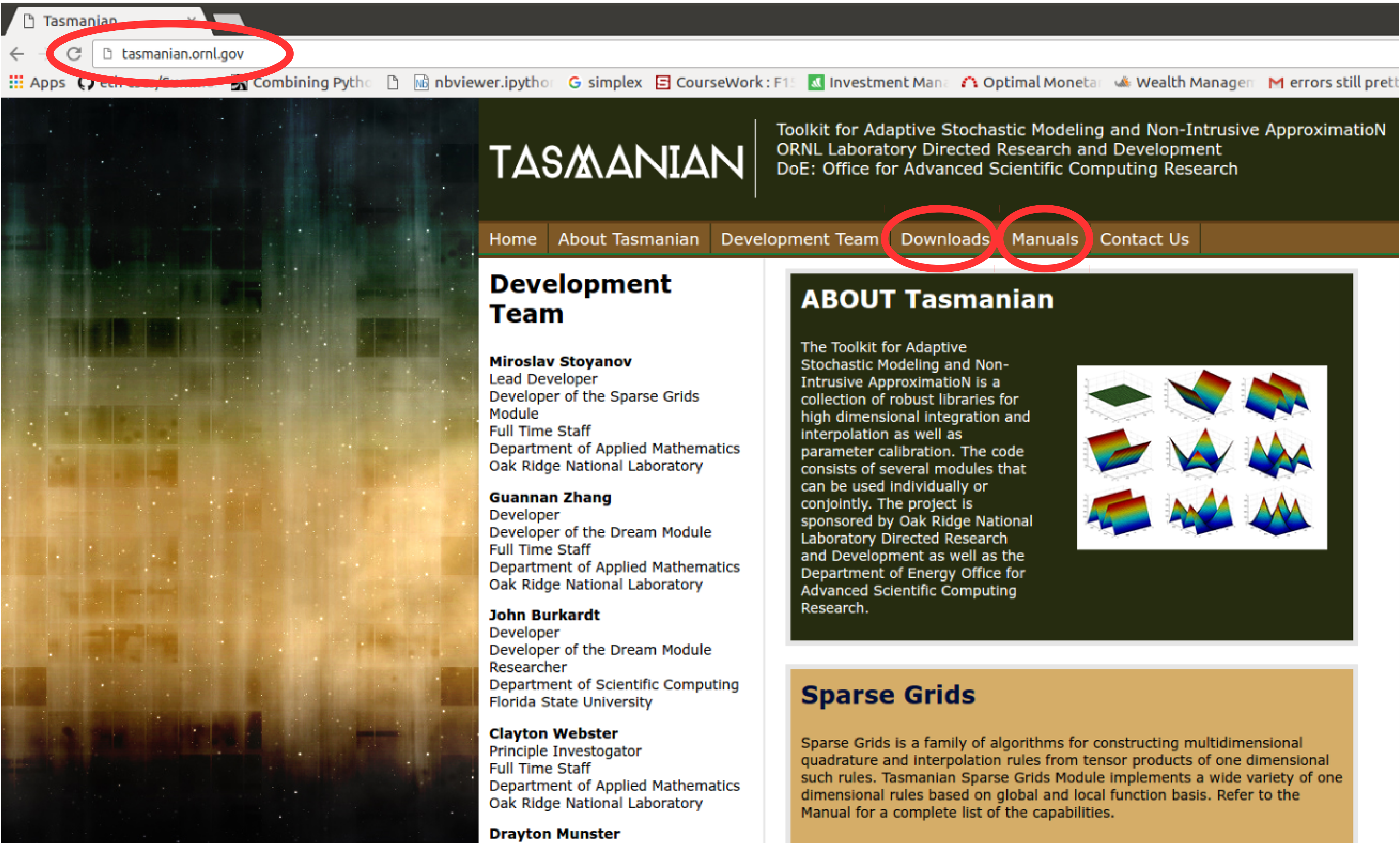
1. log onto Yale's HPC system, and go to your git repository.

> cd global_solution_yale19/Lecture_2/SparseGridCode

2. Install Tasmanian (SG library), SPINTERP (SG library), IPOPT, and PYIPOPT (optimizer)

```
> install_SG.sh
```

TASMANIAN – open source code



Software tutorial

The Toolkit for Adaptive Stochastic Modeling and Non-Intrusive ApproximatioN

http://tasmanian.ornl.gov/

TASMANIAN Sparse Grids.

Very recent open source library written in CPP:

→ Contains “ordinary and adaptive” sparse grids.

→ Many more basis functions (global polynomials, wavelets,…).

→ Interfaces to **Python** and **Matlab**.

→ **You better use it out of C++ or Python**

→ Moderately parallelized.

Compile & run TASMANIAN*

!!! READ THE * MANUAL (RTFM) !!!**

→ **Log on to MIDWAY**

> ssh USERNAME@@grace.hpc.yale.edu

→ **Load matlab**

> module load matlab

→ **Start MATLAB without graphical interface**

> matlab -nojvm

1. go to simple example:

> cd global_solution_yale19/Lecture_2/SparseGridCode/analytical_examples/TASMANIAN_Matlab

2. let's have a look at the example:

> tsg_example.m

3. launch matlab & run example:

> tsg_example()

4. NOTE: Tasmanian [-1,1]^d instead of [0,1]^d

If you are interested in CPP code examples, TASMANIAN provides examples here:
> cd TasmanianSparseGrids/Example/example.cpp
> make

***more examples provided from TASMANIAN in InterfaceMATLAB/tsgExample.m → beyond scope of lecture.**

TASMANIAN in Python

!!! READ THE * MANUAL (RTFM) !!!**

→ Log on to MIDWAY

> ssh USERNAME@@grace.hpc.yale.edu

1. go to simple example:

> cd global_solution_yale19/Lecture_2/SparseGridCode/analytical_examples/TASMANIAN_Python

2. let's have a look at the example:

> tsg_example.py

3. run example:

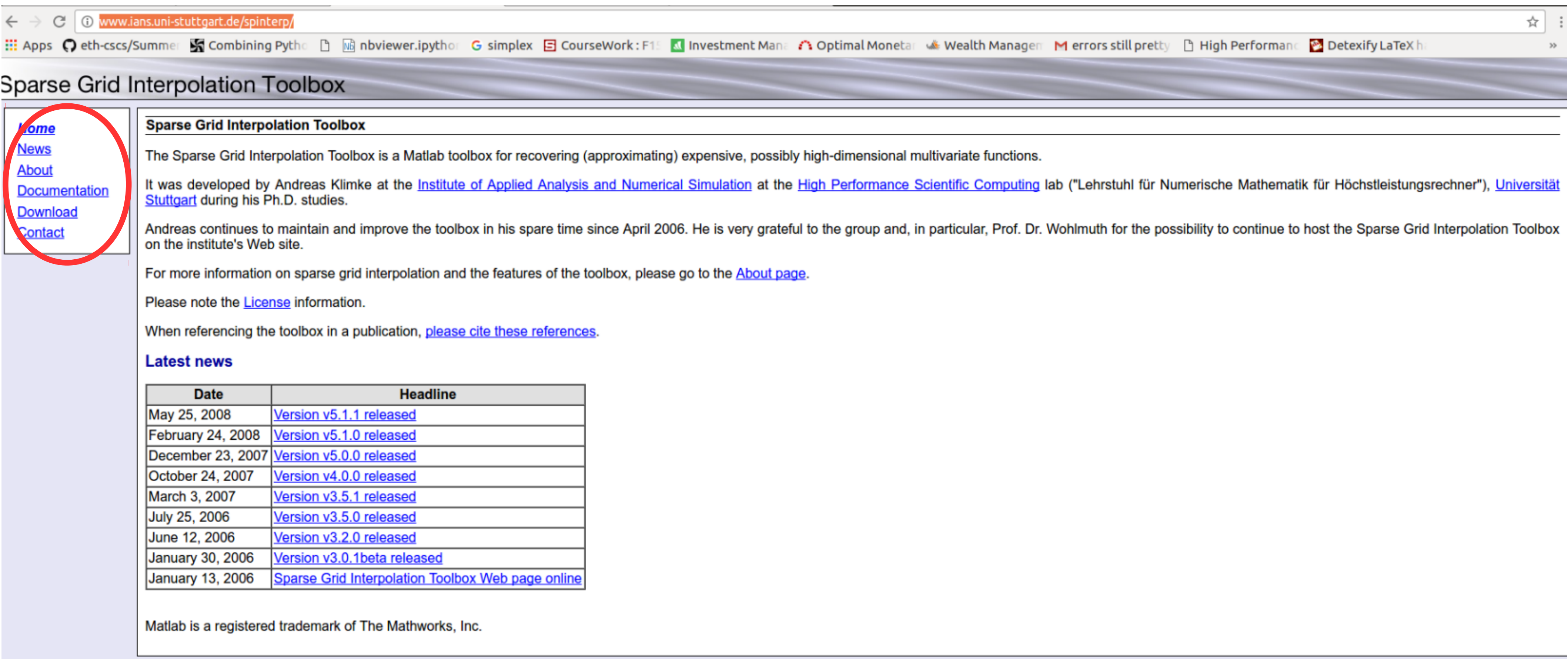
> python tsg_example.py

4. NOTE: Tasmanian [-1,1]^d instead of [0,1]^d

***more examples provided from TASMANIAN in InterfacePython/example.py → beyond scope of lecture.**

Alternative Toolboxes (I)

http://www.ians.uni-stuttgart.de/spinterp/



www.ians.uni-stuttgart.de/spinterp/

Sparse Grid Interpolation Toolbox

- Home
- News
- About
- Documentation
- Download
- Contact

Sparse Grid Interpolation Toolbox

The Sparse Grid Interpolation Toolbox is a Matlab toolbox for recovering (approximating) expensive, possibly high-dimensional multivariate functions.

It was developed by Andreas Klimke at the Institute of Applied Analysis and Numerical Simulation at the High Performance Scientific Computing lab ("Lehrstuhl für Numerische Mathematik für Höchstleistungsrechner"), Universität Stuttgart during his Ph.D. studies.

Andreas continues to maintain and improve the toolbox in his spare time since April 2006. He is very grateful to the group and, in particular, Prof. Dr. Wohlmuth for the possibility to continue to host the Sparse Grid Interpolation Toolbox on the institute's Web site.

For more information on sparse grid interpolation and the features of the toolbox, please go to the About page.

Please note the License information.

When referencing the toolbox in a publication, please cite these references.

Latest news

Date	Headline
May 25, 2008	Version v5.1.1 released
February 24, 2008	Version v5.1.0 released
December 23, 2007	Version v5.0.0 released
October 24, 2007	Version v4.0.0 released
March 3, 2007	Version v3.5.1 released
July 25, 2006	Version v3.5.0 released
June 12, 2006	Version v3.2.0 released
January 30, 2006	Version v3.0.1beta released
January 13, 2006	Sparse Grid Interpolation Toolbox Web page online

Matlab is a registered trademark of The Mathworks, Inc.

Paper for this code: global_solution_yale19/Lecture_2/lit/p561-klimke.pdf

Alternative Toolboxes (II)

http://www.ians.uni-stuttgart.de/spinterp/

→ spinterp.

→ Matlab-based implementation of sparse grids

→ Not updated since 2008 (page sometimes even down)

→ Piecewise linear basis function and few others (global)

→ Dimensional adaptivity as options

→ **no general adaptivity**

→ **not parallel**

Run Example Code on Grace

→ **Log on to MIDWAY**
> ssh USERNAME@@grace.hpc.yale.edu

→ **Load matlab**
> module load matlab

→ **Start MATLAB without graphical interface**
> matlab -nojvm

→ **Go to example and run it.**
> cd global_solution_yale19/Lecture_2/SparseGridCode/spinterp_v5.1.1
> addpath('spinterp_v5.1.1')
> spinit
> cd examples
> spdemo

```
%
% A 2D-example for multi-linear sparse grid interpolation using the
% Clenshaw-Curtis grid and vectorized processing of the function.
%
%    See also SPINTERP, SPVALS.

% Author : Andreas Klimke, Universität Stuttgart
% Version: 1.1
% Date   : September 29, 2003

% ------------------------------------------------------------
% Sparse Grid Interpolation Toolbox
% Copyright (c) 2006 W. Andreas Klimke, Universitaet Stuttgart
% Copyright (c) 2007-2008 W. A. Klimke. All Rights Reserved.
% See LICENSE.txt for license.
% email: klimkeas@ians.uni-stuttgart.de
% web  : http://www.ians.uni-stuttgart.de/spinterp
% ------------------------------------------------------------

% Some function f
f = inline('1./((x*2-0.3).^4 +(y*3-0.7).^2+1)');
```

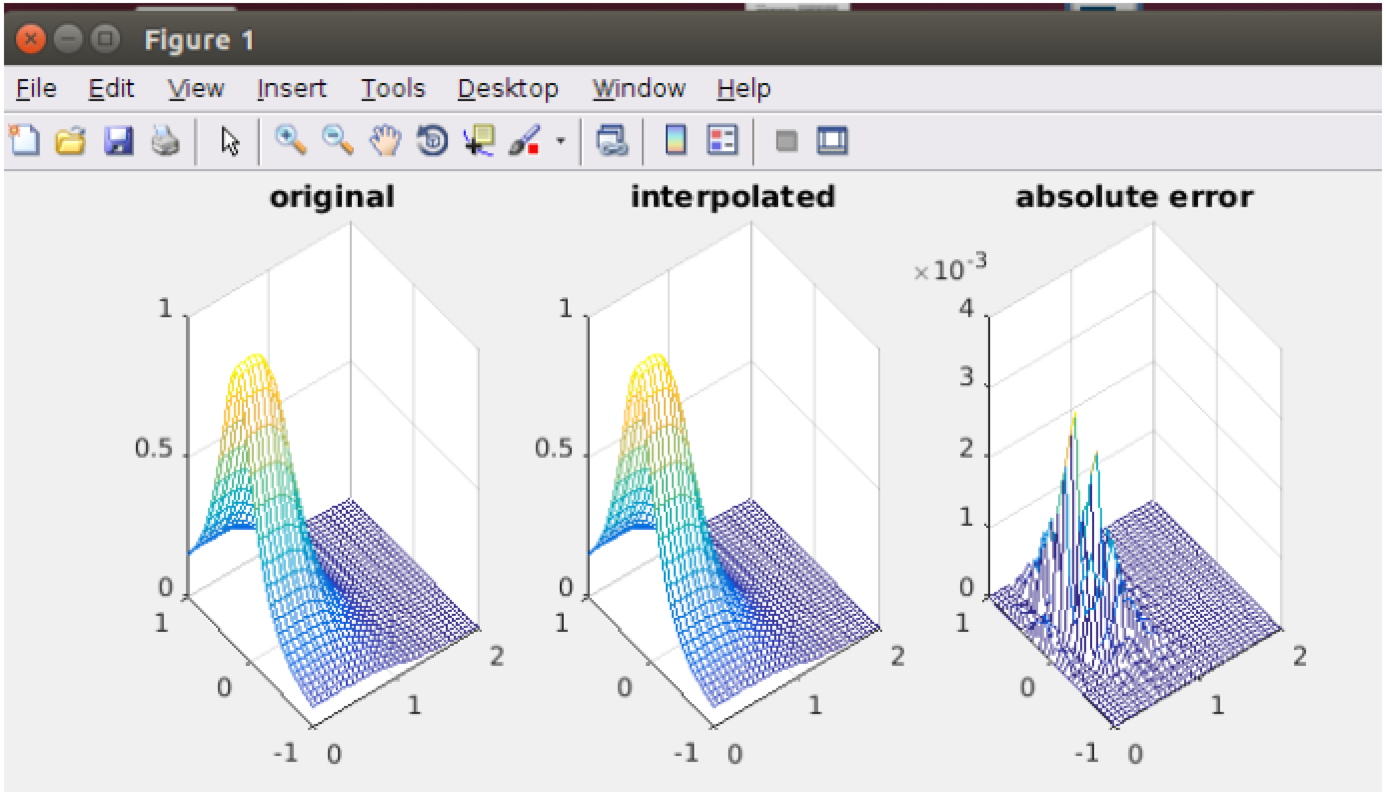
Test function

```
% Define problem dimension
d = 2;

% Create full grid for plotting
gs = 33;
[X,Y] = meshgrid(linspace(0,2,gs),linspace(-1,1,gs));

% Set options: Switch vectorized processing on.
options = spset('Vectorized', 'on', 'SparseIndices', 'off');

% Compute sparse grid weights over domain [0,2]x[-1,1]
z = spvals(f, d, [0 2; -1 1], options);
```

Interpolate

```
% Compute inpterpolated values at full grid
ip = spinterp(z, X, Y);

% Plot original function, interpolation, and error
subplot(1,3,1);
mesh(X,Y,f(X,Y));
title('original');

subplot(1,3,2);
mesh(X,Y,ip);
title('interpolated');

subplot(1,3,3);
mesh(X,Y,abs(f(X,Y)-ip));
title('absolute error');

disp(' ');
disp('Sparse grid representation of the function:');
```

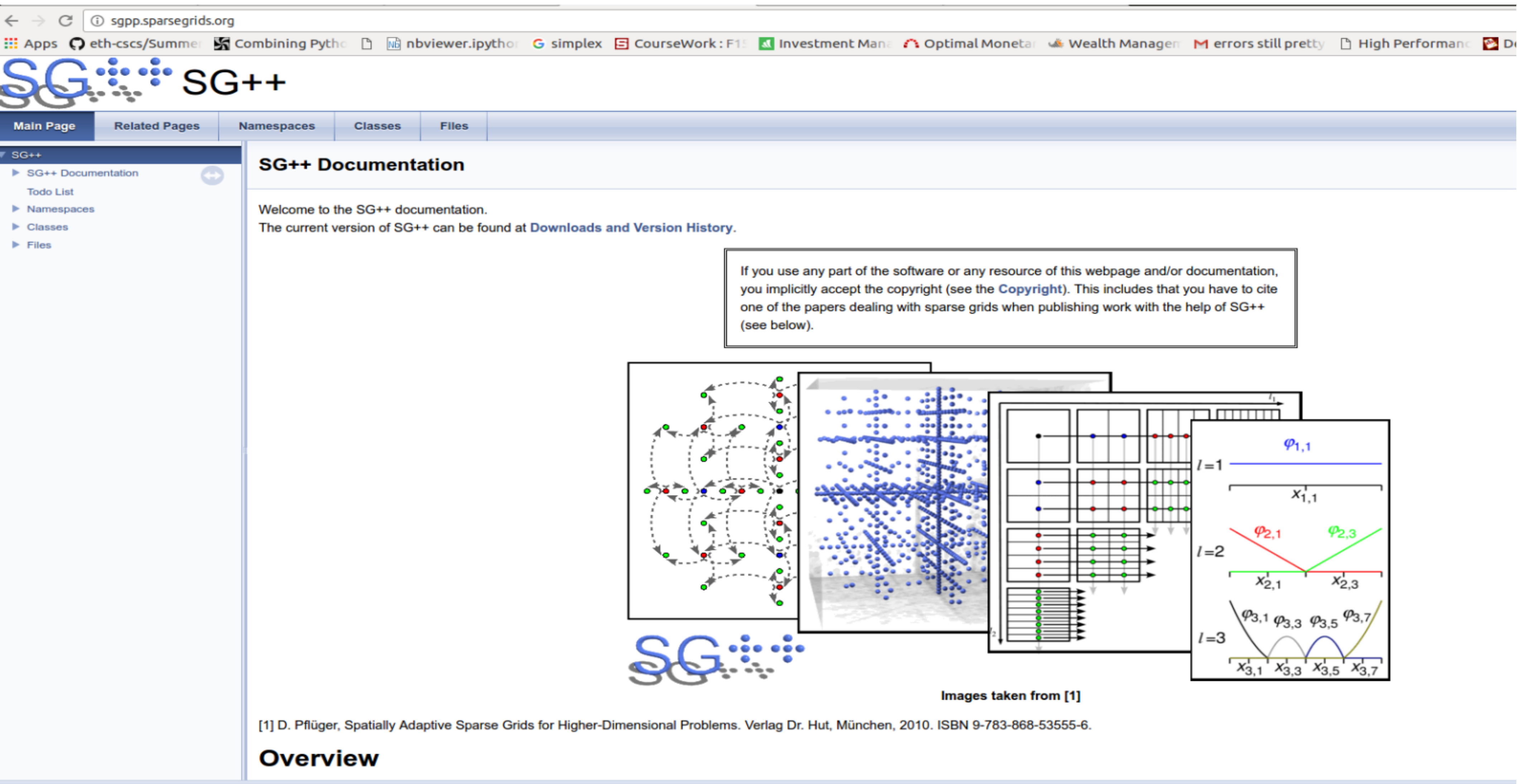
What you should see...



Other Toolboxes (III)

http://sgpp.sparsegrids.org/

- C++ with some plug-ins
- Multiple local basis functions



Exercises – for later

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