Technische Universität München
Lehrstuhl für Informatik V
Univ.-Prof. Dr. M. Bader
M.Sc. A. Mo-Hellenbrand, M.Sc. J.-M. Gallard

SS 2017
Worksheet 9
05.7.2017

Algorithms of Scientific Computing

Hierarchization in Higher Dimensions, Spatial Adaptivity

Exercise 1: Hierarchization in Higher Dimensions

In this exercise we will implement the multi-recursive algorithm for hierarchization of a multi-dimensional regular sparse grid. The structure of the code resembles strongly the one-dimensional case. We have a class (`PagodaFunction`) representing our grid points.

(i) Implement the refinement criterion `MinLevelCriterion` that adds all points up to a specified level to a given grid.
Hint: In your grid traversal, try to avoid multiple visits to the same grid points.

(ii) Implement the function `hierarchize` efficiently using a recursive approach.
Hint: The underlying traversal algorithm can be implemented similar to the one in (i).

(iii) Implement a function to compute the volume of the sparse grid interpolant.

Exercise 2: Adaptive Sparse Grids

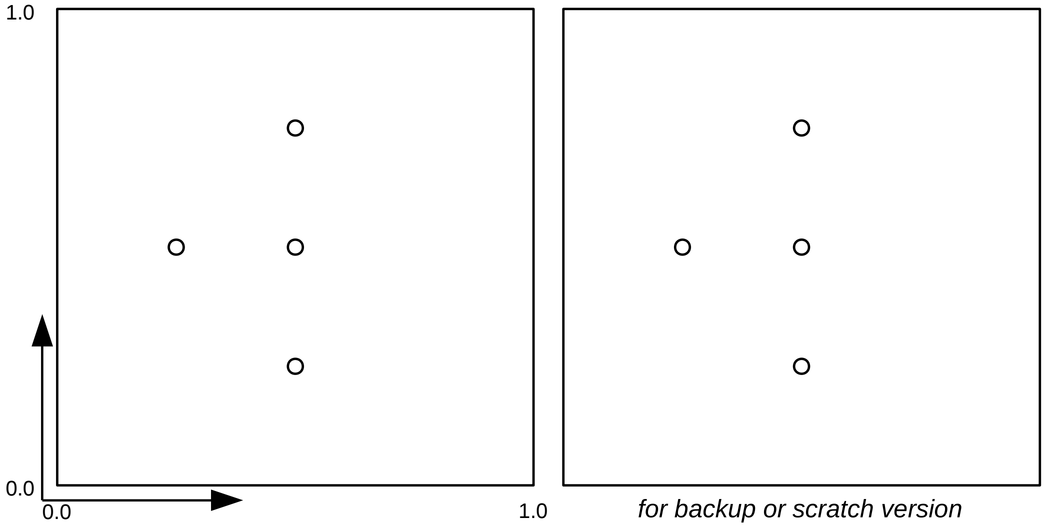
Here, the exercise is to adaptively refine a 2-dimensional sparse grid without boundary. We follow the notation introduced in the lecture and choose our domain accordingly with $\Omega = [0.0, 1.0]^2$.

1. In the following image you see an incomplete regular sparse grid V_2^1. Insert the missing grid points using small **squares**. What are the level-index-vector pairs $\vec{l}, \vec{i}$ for each of them?

2. Use the (modified) picture from the previous task to perform two steps of adaptive refinement:

   (a) Refine grid point $\vec{l}, \vec{i} = (1, 2), (1, 3)$: create all hierarchical children. Draw its children as small **triangles**. Make sure that you also insert all missing hierarchical parents (and parents of parents, . . .) of these children to make the grid suitable for typical algorithms on sparse grids.

   (b) Now refine grid point $(2, 2), (3, 3)$. Again, do not forget to create all missing parents. Draw all new points as small **crosses**.



Exercise 3: One-dimensional Sparse Grids—An Adaptive Implementation

Last week we introduced Archimedes' approach to approximate the integral $F(f,a,b) = \int_a^b f(x)\ dx$ of a function $f : \mathbb{R} \to \mathbb{R}$, respectively to approximate the function f itself.

For the one-dimensional case we want to formalize this approach and generalize it in the following ways:

- Let $\phi(x)$ be the "mother of all hat functions" with

$$\phi(x) = \begin{cases} x+1 & \text{for } -1 \le x < 0 \\ 1-x & \text{for } 0 \le x < 1 \\ 0 & \text{else} \end{cases} \tag{1}$$

- The data structure used to store the hierarchical coefficients is now called *Sparse Grid*.
- A sparse grid is defined by a particular set of interpolation points $x_{l,i}$ and associated ansatz functions $\phi_{l,i}(x)$ with

$$\phi_{l,i}(x) = \phi(\ 2^l \cdot \left(x - i \cdot \frac{1}{2^l}\right)\ ) = \phi(2^l \cdot x - i)\ , \qquad l \in \mathbb{N}^+, i \in \{1,3,\ldots,2^l-1\} \tag{2}$$

- Archimedes' approach from the lecture corresponds to a *regular* sparse grid.
- To improve the quality of approximation for arbitrary functions f we introduce spatial adaptivity.

Your task is to implement the missing parts in the members of the *SparseGrid1d* class and turn it into a fully working adaptive implementation of a one-dimensional sparse grid. Import and use the class *GridPoint* and look at the comments in the provided code snippets for some more details.

a) The constructor *__init__* creates a grid containing all grid points on levels $l \leq minLevel$. A given function f is then evaluated at those points before *hierarchization* is performed eventually to obtain the hierarchical coefficients.
Implement this behavior.

b) Implement the member function *computeVolume* that computes an approximation for $F(f, 0, 1)$ using the current sparse grid interpolant.

c) Implement the member function *refineAdaptively* that takes a certain refinement criterion (see source code) and inserts new grid points accordingly.