

Algorithms of Scientific Computing

From Quadtrees to Space-Filling Curves

Michael Bader

Summer Term 2014



Overview: Modelling of Geometric Objects

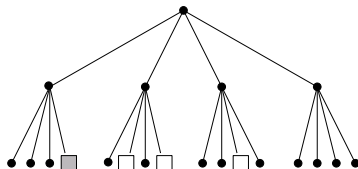
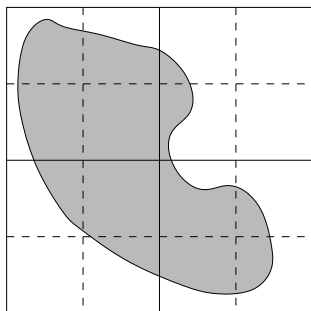
Surface-oriented models:

- wire-frame models
- augmented models using Bezier curves and planes
- typically described by graphs on nodes, edges, and faces

Volume-oriented models:

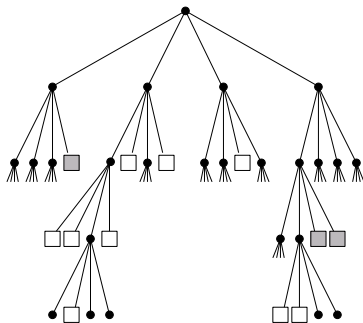
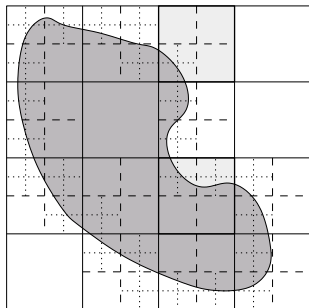
- Constructive Solid Geometry (boolean operations on primitives)
- voxel models: place object in a grid
- octrees: recursive refinement of voxel grids

Quadrees to Describe Geometric Objects



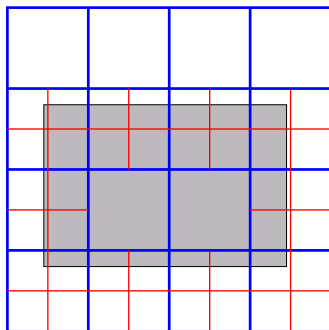
- start with an initial square (covering the entire domain)
- recursive substructuring in four subsquares

Quadrees to Describe Geometric Objects

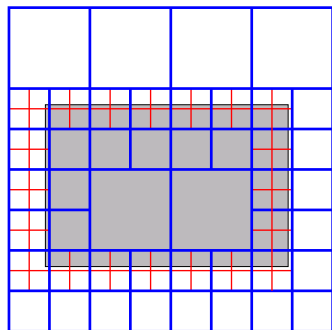


- start with an initial square (covering the entire domain)
- recursive substructuring in four subsquares
- adaptive refinement possible
- terminate, if squares entirely within or outside domain

Number of Quadtree Cells to Store a Rectangle



$k=2$

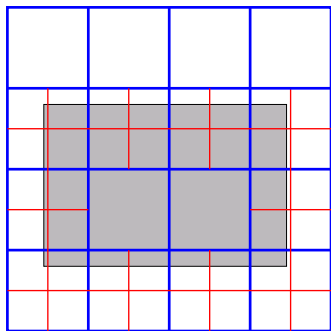


$k=3$

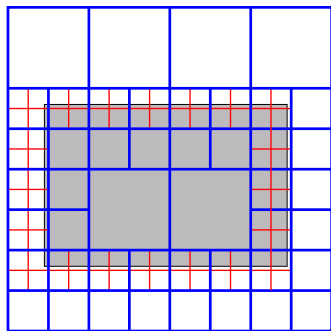
Terminal (t_k) and boundary (b_k) cells after k refinement steps:

$$\begin{aligned}
 b_k &= 2 \cdot b_{k-1} \\
 t_k &= t_{k-1} + 2 \cdot b_{k-1}
 \end{aligned}
 \Rightarrow
 \begin{aligned}
 b_k &= 2^{k-2} \cdot b_2 = \frac{5}{2} \cdot 2^k \\
 t_k &= \dots = 5 \cdot 2^k - 14
 \end{aligned}$$

Number of Quadtree Cells to Store a Rectangle



$k=2$

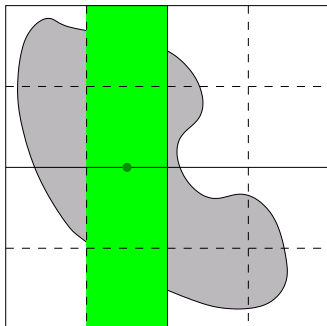


$k=3$

- uniformly ref. voxel-grid (level k): $(2^{d=2})^k = (2^k)^2 =: \mathcal{O}(N^2)$ cells
 - quadtree-refined grid (level k): $\frac{15}{2} \cdot 2^k - 14 =: \mathcal{O}(N)$ cells
- \Rightarrow number of cells proportional to length of boundary ($N := 2^k$)

Hierarchical Basis vs. Quadtree

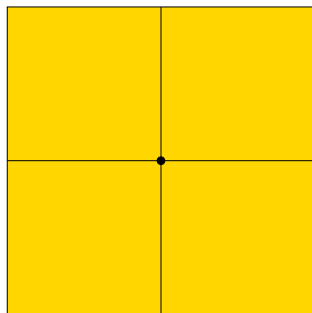
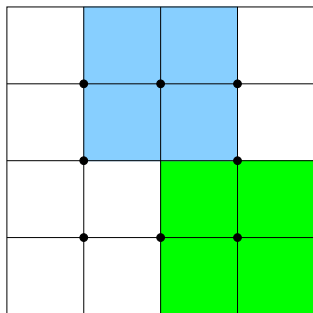
Use hierarchical basis as in 2D sparse grids?



⇒ tensor basis functions do not match quadtree cells

Hierarchical Basis for Quadrees

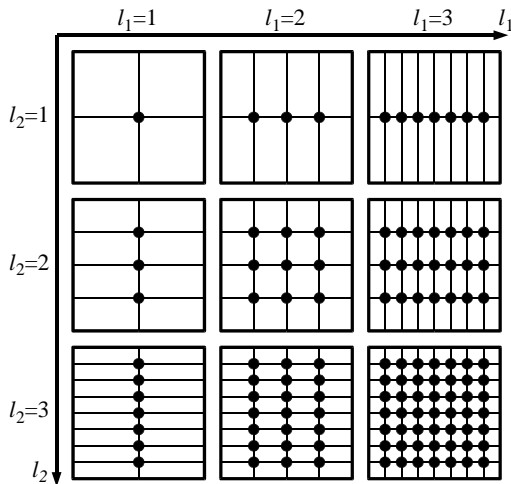
Use hierarchical multilevel basis:



hierarchical concept (again): skip basis functions that exist on previous level!

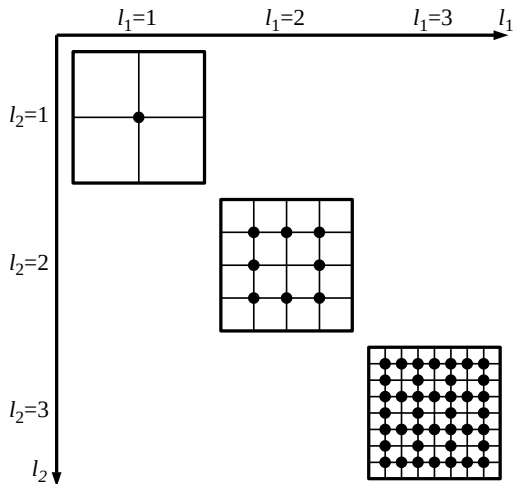
Quadtree-Compatible Hierarchical Basis

Picking Subspaces



Quadtree-Compatible Hierarchical Basis

Picking Subspaces



Quadtree-Compatible Hierarchical Basis

Basis Functions

Similar to tensor-product basis:

- Only use diagonal subspaces

$$\vec{n} := \{n, \dots, n\}$$

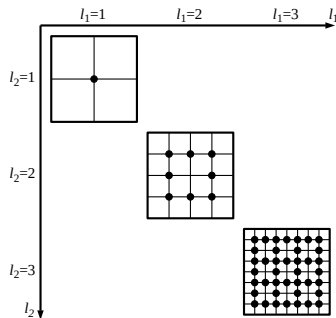
- Omit grid points with even index
- Now index even in all directions

$$\hat{\mathcal{I}}_{\vec{n}} := \{\vec{i} : \vec{1} \leq \vec{i} < 2^{\vec{n}}, \text{ any } i_j \text{ odd}\}$$

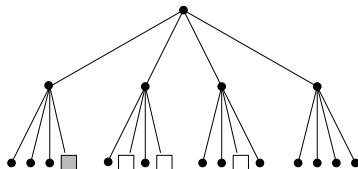
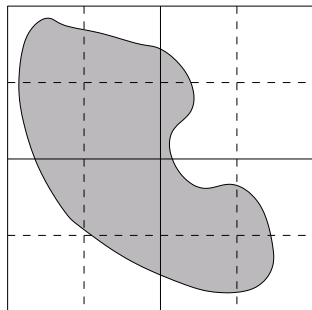
⇒ Hierarchical increments

$$W_{\vec{n}} := \text{span}\{\phi_{\vec{n}, \vec{i}}\}_{\vec{i} \in \hat{\mathcal{I}}_{\vec{n}}}$$

again contain all functions of $V_{\vec{n}}$ that vanish at grid points of coarser grids

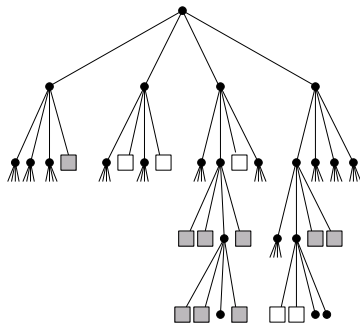
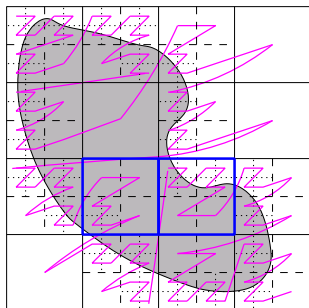


Storing a Quadtree – Sequentialisation



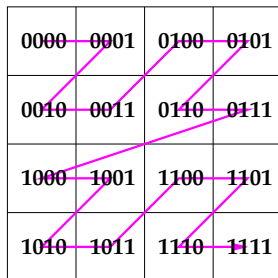
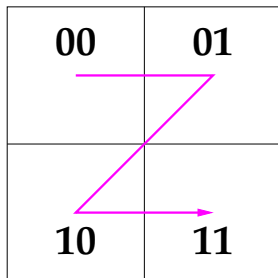
- sequentialise cell information according to **depth-first traversal**
- relative numbering of the child nodes determines sequential order

Storing a Quadtree – Sequentialisation



- sequentialise cell information according to **depth-first traversal**
- relative numbering of the child nodes determines sequential order
- here: leads to so-called **Morton order**

Morton Order ("Z curve")



Relation to bit arithmetics:

- odd digits: position in vertical direction
- even digits: position in horizontal direction

Morton Order and Cantor's Mapping

Georg Cantor (1877):

$$0.01111001\dots \rightarrow \begin{pmatrix} 0.0110\dots \\ 0.1101\dots \end{pmatrix}$$

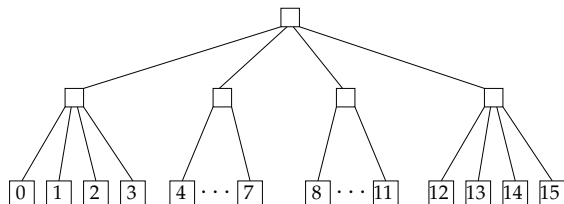
- **bijjective** mapping $[0, 1] \rightarrow [0, 1]^2$
- proved identical cardinality of $[0, 1]$ and $[0, 1]^2$
- provoked the question: is there a **continuous** mapping?
(i.e. a curve)

Preserving Neighbourship for a 2D Octree

Requirements:

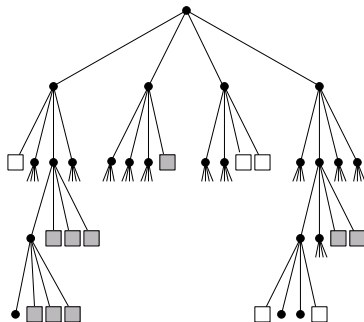
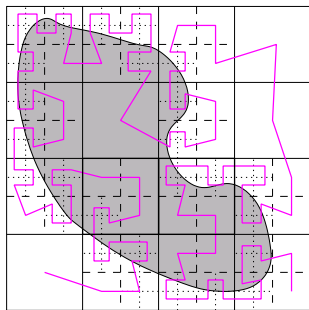
- consider a simple 4×4 -grid
- uniformly refined
- subsequently numbered cells should be neighbours in 2D

Leads to (more or less unique) numbering of children:



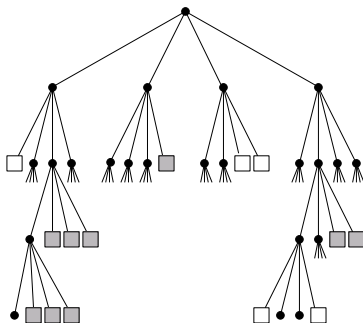
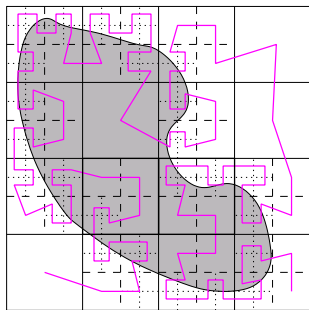
5	6	9	10
4	7	8	11
3	2	13	12
0	1	14	15

Preserving Neighbourship for a 2D Octree (2)



- adaptive refinement possible
- neighbours in sequential order remain neighbours in 2D

Preserving Neighbourship for a 2D Octree (2)



- adaptive refinement possible
- neighbours in sequential order remain neighbours in 2D
- here: similar to the concept of **Hilbert curves**

Open Questions

Algorithmics:

- How do we describe the sequential order algorithmically?
- What kind of operations are possible?
- Are there further “orderings” with the same or similar properties?

Applications:

- Can we quantify the “neighbour” property?
- In what applications can this property be useful?
- Which other properties and/or operations can be useful?