Technische Universität München
Lehrstuhl für Informatik V
Dr. T. Neckel, D. Pflüger

SS 2010
Sheet 7
22.6.2010

Algorithms of Scientific Computing (Algorithmen des Wissenschaftlichen Rechnens)

Combination Technique

Proposed solution

Dealing with hierarchical bases often turns out to be sophisticated. On this worksheet we will therefore see how the so-called *combination technique* finds a sparse grid interpolant, that approximates a function on a number of full grids, each consisting only of a "relatively small" number of grid points.

Let $u_{\underline{l}}$ ($\underline{l} \in \mathbb{N}^2$) for a $u : [0,1]^2 \to \mathbb{R}$ the interpolant in $V_{\underline{l}}$ (interpolating piecewise bilinearly at the inner grid points, at the boundary u is assumed to be zero again).

(i) Rewrite $u_{\underline{l}}$ as a (weighted) sum of $w_{\underline{l}} \in W_{\underline{l}}$.

No weights needed here. We only need to "collect" those $w_{\underline{l}}$ with indices bound componentwise by $\underline{l}$ (in subspace scheme: the subspaces in the rectangular left upper part relative to $\underline{l}$:

$$u_{\underline{l}} = \sum_{\underline{l} \leq \underline{l}} w_{\underline{l}}$$

(exactly those basis functions that are not vanishing in the grid points $x_{\underline{l},\underline{i}}$).

(ii) For $n \in \mathbb{N}$ write

$$\sum_{|\underline{l}|_1 = n+1} u_{\underline{l}}$$

as a weighted sum of $w_{\underline{l}}$ and arrange the summands with respect to $\underline{l}$.

Using the previous result we start with

$$\sum_{|\underline{l}|_1 = n+1} u_{\underline{l}} = \sum_{|\underline{l}|_1 = n+1} \sum_{\underline{l}' \leq \underline{l}} w_{\underline{l}'}.$$

For the reorganization part we first count which $w_{\underline{l}'}$ appears how many times in the sums. (remember, this is **2d***!)*

$|\underline{l}'|_1 = n+1$: *each $w_{\underline{l}}$ has one occurrance*

$|\underline{l}'|_1 = n$: *each $w_{\underline{l}}$ has two occurrances*

$\vdots$

$|\underline{l}'|_1 = k$: *each has $n + 2 - k$ occurrances*

Technical explanation: Let $\underline{l}$ be of level $n+1$, i.e. $|\underline{l}|_1 = n+1$. From $\underline{l}$ construct all possible $\underline{l}'$ of level n with $\underline{l}' \leq \underline{l}$. Those are exactly two, as there are two components (2d!) that can be decreased by 1. Applying this scheme down to level 1 then leads to result above.

Written as an explicit sum formula this is:

$$\sum_{|\underline{l}|_1=n+1} u_{\underline{l}} = \sum_{|\underline{l}|_1\leq n+1} (n+2-|\underline{l}|_1) w_{\underline{l}}.$$

(iii) Use the previous results to give a representation of the sparse grid interpolant

$$u_n^D := \sum_{|\underline{l}|_1\leq n+1} w_{\underline{l}}$$

as a weighted sum of $u_{\underline{l}}$.

By looking at the subspace scheme rather than by looking at the formulas it becomes clear that the following holds:

$$\sum_{|\underline{l}|_1\leq n+1} w_{\underline{l}} = \sum_{|\underline{l}|_1=n+1} u_{\underline{l}} - \sum_{|\underline{l}|_1=n} u_{\underline{l}}$$

(iv) Assume you are talking to a person who knows how to approximate the volume $F_2(u)$ through the trapezoidal rule (in 2d) with respect to $u_{\underline{l}}$. Give instructions on how to write a program that implements a sparse grid approximation of $F_2(u)$. Remember Archimedes quadrature.

- *First idea: Replace volume $F_2(u)$ by the sparse grid volume approximation $F_2(u_n^D)$.*
- *Second idea: Think of the interpolant as a sum of $u_{\underline{l}}$. We know those $u_{\underline{l}}$ (interpolating u on regular grids) as well as their volumes (trapezoidal rule in 2d).*
- *Together with the weights from the previous part we get*

$$F_2(u) \approx F_2(u_n^D) = \sum_{|\underline{l}|_1=n+1} F_2(u_{\underline{l}}) - \sum_{|\underline{l}|_1=n} F_2(u_{\underline{l}})$$

(v) Compare this method with Archimedes quadrature — what are the (dis-)advantages?

Advantages:

- *Simpler program code (Haven't you tried coding them? Do it! You'll agree...)*
- *It might be possible to reuse an existing program for the trapezoidal rule on common regular grids (advantage is even bigger for more complex applications, e.g. when computing a sparse grid solution for a fluid simulation)*
- *For comprehensive computations the program is more likely and easy to be parallelized as the single grids are processed independently from each other*

Disadvantages:

- *No straight forward approach to include adaptivity, i.e. it's not possible to automatically find the right evaluation points*
- *Recursion is much more beautiful!*