2015:07

Tree Approximation for hp-Adaptivity

Peter Binev

Tree Approximation for hp-Adaptivity

Peter Binev [*]

May 7, 2014

Abstract

The hp-adaptive approximation is formulated as an approximation problem on a full binary tree T, where for each of the leaves Δ an order $p(\Delta) \geq 1$ is assigned in such a way that the sum of all such $p(\Delta)$ does not exceed N, called complexity of the approximation. The leaves Δ correspond to the cells of the partition, while $p(\Delta)$ is the order of the polynomial used for the local approximation on Δ. Devising an incremental algorithm for near-best adaptive approximation for the problem of finding the best possible tree T and assignments $p(\Delta)$ leads to building a construction that attaches a ghost tree with $p(\Delta)$ leaves to each leaf Δ of T with $p(\Delta) > 1$. The resulting full binary tree $\mathcal{T}$ that has at most N leaves and can be used as a proxy of T for assembling hp-adaptive procedures. Under the standard assumptions about the local errors, we prove that our approximation of complexity N is bounded by $\frac{2N-1}{N-n+1}\sigma_n$, where σ_n, $n \leq N$, is the best possible approximation of complexity n.

Keywords: adaptive methods, hp-adaptivity, tree based algorithms, near-best approximation, instance optimality.
MSC numbers: 41A15, 41A63, 65D15, 65M55, 68Q32, 68W25, 97N50.

1 Introduction

There are two basic approaches to adaptivity when approximating a function on a domain Ω. The first one considers the current partition of Ω and chooses some of its elements Δ for refinement. The tool for local approximation does not change and the improvement of the approximation is based on decreasing of the size h of the elements. Thus, this is usually called an *h-refinement.* The second one considers the same partition of Ω but the order $p(\Delta)$ of the approximation tool is increased on some elements Δ, and thus, it is called a *p-refinement.* The combination of both approaches results in approximation strategies often referred to as *hp-adaptivity.* The goal of this paper is to investigate the hp-adaptivity in very general settings and to introduce a framework for which one can find a near-optimal algorithm with a reasonable complexity.

In general, adaptive refinement can be linked to building a tree structure. The initial partition consists of just one element, the domain Ω itself, that we relate to the root $\mathcal{R}$ of the tree. Going forward, subdividing an element of the current partition corresponds to taking a terminal node, called a *leaf* of the current tree and attaching to it new leaves that are related to the newly created smaller elements. While the subdivided element Δ is no longer an element of the partition, the corresponding node remains as an *internal* node of the tree but is no longer a leaf. To simplify

[*]This research was supported by the NSF grant DMS 1222390.

the presentation, from now on we will consider the case of binary subdivision, namely, that Δ is subdivided into two smaller elements Δ' and Δ'', such that $\Delta = \Delta' \cup \Delta''$ and $\Delta' \cap \Delta''$ is an empty set or a set of measure zero. We shall call Δ' and Δ'' children of Δ that will be referred as their parent. The resulting graph will be a full binary tree with a root $\mathcal{R}$ and the elements of the current partition will be identified with the current leaves of the tree. With a slight abuse of the notation, we will refer to Δ as both an element of the partition and a node of the binary tree.

For a given tree T we denote the set of its leaves by $\mathcal{L}(T)$. The internal nodes of T also form a binary tree $T \backslash \mathcal{L}(T)$ but it might not be a full tree, i.e. there might be a parent node with just one child. We define the complexity N of a full tree to be $\#\mathcal{L}(T)$, the number of its leaves. This conveniently corresponds to the number of elements in the partition of Ω constituted by T.

The local errors at an element Δ of the partition are denoted by $e_p(\Delta)$, where $p = \mathcal{P}(\Delta)$ is the order of the polynomial space used in the approximation at Δ. It is important to emphasize that we want to work with additive quantities and to have that the total error $\mathcal{E}(T, \mathcal{P}(T))$ for a given partition defined by T and the assignments

$$\mathcal{P}(T) := \Big(\mathcal{P}(\Delta)\Big)_{\Delta \in \mathcal{L}(T)} \tag{1.1}$$

is defined by

$$\mathcal{E}(T, \mathcal{P}(T)) := \sum_{\Delta \in \mathcal{L}(T)} e_{\mathcal{P}(\Delta)}(\Delta)\,. \tag{1.2}$$

In particular, this means that when working with the L_2-norm, we define the error to be the square of the L_2-norm of the difference between the function and the approximating polynomial. This presentation uses very general settings about the error $e_p(\Delta)$, $p \geq 1$ only requiring the following two properties:
(i) subadditivity of the error for the lowest order approximation:

$$e_1(\Delta) \geq e_1(\Delta') + e_1(\Delta'')\,, \tag{1.3}$$

where Δ' and Δ'' are the children of Δ;
(ii) reduction of the error when the polynomial order increases

$$e_p(\Delta) \geq e_{p+1}(\Delta)\,. \tag{1.4}$$

In some cases it is convenient and often necessary to consider a less demanding variant of (1.3) known as weak subadditivity:

$$e_1(\Delta) \geq c \sum_{\Delta' \in (\mathcal{T}_\Delta \cap T)} e_1(\Delta')\,, \tag{1.5}$$

where $c > 0$ is a fixed constant independent of the choice of Δ or the tree T. In this formula and everywhere else in the paper $\mathcal{T}_\Delta$ stands for the infinite full binary tree rooted at Δ. It is easy to see that (1.5) holds with $c = 1$ in case (1.3) is true. While the proofs presented below can be modified to use (1.5) instead of (1.3), this will result in additional complications and less favorite constants. To keep the presentation simple, we choose to use the property (1.3) and refer the reader to [4] for considerations in full generality in the case of h-adaptivity.

The definition of the best approximation depends on the notion of complexity. Here we set the complexity of the hp-adaptive approximation by the pair $(T, \mathcal{P}(T))$ to be the sum of orders at all the elements of the partition

$$\#\mathcal{P}(T) := \sum_{\Delta \in \mathcal{L}(T)} \mathcal{P}(\Delta)\,. \tag{1.6}$$

Given $N > 0$, the best hp-adaptive approximation of complexity N is then defined by

$$\sigma_N := \inf_{T} \inf_{\#\mathcal{P}(T)\le N} \mathcal{E}(T, \mathcal{P}(T)) . \tag{1.7}$$

The aim of this paper is to define an incremental algorithm, similar to the one in [4], that for given error assignments satisfying (1.3) and (1.4) produces for each N a pair $(T_N, \mathcal{P}(T_N))$ of a tree T_N and the corresponding polynomial orders $\mathcal{P}(T_N)$ with $\#\mathcal{P}(T_N) = N$ such that it provides a near-best hp-adaptive approximation, namely

$$\mathcal{E}(T_N, \mathcal{P}(T_N)) \le C_1 \sigma_{c_2 N} . \tag{1.8}$$

First, we have to define a framework that allows to increase the complexity of the pair $(T_N, \mathcal{P}(T_N))$ by preserving the local distribution of the degrees of freedom and, at the same time, allowing the flexibility to make substantial changes in $(T_{N+1}, \mathcal{P}(T_{N+1}))$. The idea is to create a tree $\mathcal{T}$ with $\mathcal{L}(\mathcal{T}) = N$ leaves, for which T_N is a subtree in such a way that for any leaf $\Delta \in \mathcal{L}(T_N) \subset \mathcal{T}$ the number of leaves of $\mathcal{T}$ that are descendants of Δ is equal exactly to the order $\mathcal{P}(\Delta)$. In other words, at each leaf $\Delta \in \mathcal{L}(T_N) \subset \mathcal{T}$ we want to add a "ghost" tree $\mathcal{T}_\Delta \cap \mathcal{T}$ that has exactly $\mathcal{P}(\Delta)$ leaves and after adding all such trees we will compose the tree $\mathcal{T}$. Of course, the tree $\mathcal{T}$ is not uniquely defined, in general. However, it can carry information how the approximation would look like if we decide to use lower order polynomials at some location. Given $\mathcal{T}$, for each of the nodes $\Delta \in \mathcal{T}$ we define its order by

$$\mathcal{P}(\Delta) := \mathcal{P}(\Delta, \mathcal{T}) := \#\Big(\mathcal{L}(\mathcal{T}) \cap \mathcal{T}_\Delta\Big) . \tag{1.9}$$

Now, the hp-approximation can be identified with the pair $(\mathcal{T}, T)$ instead of the pair $(T, \mathcal{P}(T))$ and the same definition (1.2) applies for $\mathcal{E}(\mathcal{T}, T)$. It is easy to see that the best approximation from (1.7) can be expressed as

$$\sigma_N = \inf_{\mathcal{T}:\#\mathcal{L}(T)\le N} \inf_{T\subset\mathcal{T}} \mathcal{E}(\mathcal{T}, T) . \tag{1.10}$$

This leads to a different approach to finding a near-best approximant, namely, to find a tree $\mathcal{T}_N$ with $\#\mathcal{L}(\mathcal{T}_N) = N$ first, and then to examine all possible subtrees T of $\mathcal{T}_N$ and choose the one for which the error $\mathcal{E}(\mathcal{T}_N, T)$ is minimal. This also gives the possibility to define the trees $\mathcal{T}_N$ *incrementally* and by this minimize the computational cost.

Remark 1.1 *Note that the* inf *in the definitions* (1.7) *and* (1.10) *is acting on a finite set of possibilities and therefore can be replaced by* min.

The strategy of building the tree $\mathcal{T}$ is to identify at each step the leaf with the highest potential to decrease the total error. A straightforward greedy approach, namely to choose the leaf with the highest local error, is not going to work since a very localized singularity would attract several consecutive refinements without an (essential) improvement of the total error. In the case of h-adaptive refinements, one can modify the local errors to account for the depth of the tree in such a way that the choice for refinement of the leaf with the largest *modified* local error would result in a near-best approximation (see [4]). We present the variant of this strategy from [1] and prove some results about it in Section 2. It is important to note that the h-adaptive algorithm is driven by a quantity (the modified error) defined at the nodes but completely independent of the current tree. While this will pave the road to designing a strategy for the hp-adaptivity, it seems that a much more involved setup is needed. We describe our approach in Section 3 and prove the following result.

Theorem 1.2 *Let the local errors e_p satisfy the conditions* (1.3) *and* (1.4)*. Then there exists a constructive incremental algorithm for finding a tree $\mathcal{T}_N$ starting from $\mathcal{T}_1 = \mathcal{R}$ and for each $j \geq 1$ receiving $\mathcal{T}_{j+1}$ from $\mathcal{T}_j$ by adding two children nodes to a leaf of $\mathcal{T}_j$. In addition, for each tree $\mathcal{T}_N$ there exists a subtree T_N such that the hp-adaptive approximation provided by the pair $(\mathcal{T}_N, T_N)$ is near-best, namely,*

$$\mathcal{E}(\mathcal{T}_N, T_N) \leq \frac{2N-1}{N-n+1}\,\sigma_n \tag{1.11}$$

for any integer $n \leq N$. The complexity of the algorithm for obtaining $(\mathcal{T}_N, T_N)$ is bounded by $\mathcal{O}\Big(\sum\limits_{\Delta\in\mathcal{T}_N} \mathcal{P}(\Delta, \mathcal{T}_N)\Big)$, where $\mathcal{P}(\Delta, \mathcal{T}_N)$ is defined by (1.9)*.*

Remark 1.3 *The sum $\sum\limits_{\Delta\in\mathcal{T}_N} \mathcal{P}(\Delta, \mathcal{T}_N)$ estimating the complexity of the algorithm varies from $\mathcal{O}(N\log N)$ for well balanced trees $\mathcal{T}_N$ to $\mathcal{O}(N^2)$ for highly unbalanced ones.*

2 Near-Best Results for an h-Refinement Strategy

The setup in the case of h-adaptive approximation is much more simple. In particular, the local approximations are using the same polynomial space, so we denote the local errors by $e(\Delta)$. Since this is the basic approximation, although the orders could be higher, we relate $e(\Delta)$ to $e_1(\Delta)$ in the hp-setup and therefore assume that it satisfies the subadditivity property corresponding to (1.3):

$$e(\Delta) \geq e(\Delta') + e(\Delta'') \tag{2.1}$$

where Δ' and Δ'' are the children of Δ. The global h-error for the tree $\mathcal{T}$ is then defined by

$$\mathcal{E}^{\mathrm{h}}(\mathcal{T}) := \sum_{\Delta\in\mathcal{L}(\mathcal{T})} e(\Delta) \tag{2.2}$$

and the best N-tern h-adaptive approximation is

$$\sigma_N^{\mathrm{h}} := \min_{\mathcal{T}\,:\,\#\mathcal{L}(\mathcal{T})\leq N} \mathcal{E}^{\mathrm{h}}(\mathcal{T})\,. \tag{2.3}$$

To build the algorithm for near-best h-adaptive tree approximation, we define the modified errors $\tilde{e}(\Delta)$ as follows:

$$\tilde{e}(\mathcal{R}) := e(\mathcal{R}) \quad \text{for the root} \quad \text{and} \qquad \tilde{e}(\Delta) := \frac{e(\Delta)\tilde{e}(\Delta^\star)}{e(\Delta) + \tilde{e}(\Delta^\star)}\,, \tag{2.4}$$

where $\Delta^\star$ is the parent of Δ. In case both $e(\Delta)$ and $\tilde{e}(\Delta^\star)$ are zeros, we define $\tilde{e}(\Delta) = 0$, as well. It is sometimes beneficial to use the following equivalent formula for $\tilde{e}(\Delta)$

$$\frac{1}{\tilde{e}(\Delta)} = \frac{1}{e(\Delta)} + \frac{1}{\tilde{e}(\Delta^\star)}\,. \tag{2.5}$$

The modified error is independent on the tree and can be used to devise a greedy algorithm for finding a tree $\mathcal{T}_N$ that provides a near-best approximation. Define $\mathcal{T}_1 := \{\mathcal{R}\}$ and receive the tree $\mathcal{T}_{N+1}$ from $\mathcal{T}_N$ by subdividing a leaf $\Delta \in \mathcal{L}(\mathcal{T}_N)$ with the largest $\tilde{e}(\Delta)$ among all leaves from $\mathcal{L}(\mathcal{T}_N)$. The following theorem holds.

Theorem 2.1 *Let the local errors $e(\Delta)$ satisfy the condition (2.1) and the tree $\mathcal{T}_N$ is received by applying a greedy refinement strategy with respect to the quantities $\tilde{e}(\Delta)$ defined by (2.4). Then the tree $\mathcal{T}_N$ provides a near-best h-adaptive approximation*

$$\mathcal{E}^{\mathrm{h}}(\mathcal{T}_N) \leq \frac{N}{N-n+1}\, \sigma_n^{\mathrm{h}} \tag{2.6}$$

for any integer $n \leq N$. The complexity of the algorithm for obtaining $\mathcal{T}_N$ is $\mathcal{O}(N)$ safe for the sorting of $\tilde{e}(\Delta)$ that requires $\mathcal{O}(N \log N)$ operations.

Remark 2.2 *The sorting can be avoided by binning the values of $\tilde{e}(\Delta)$ into binary bins and choosing for subdivision any of the Δ that provides a value for the largest nonempty bin. This will only increase the constant in (2.6) by 2. In this case the total complexity of the algorithm is $\mathcal{O}(N)$.*

To prepare for the proof we should make some remarks and prove two lemmas. First, let us mention that the following quantities are decreasing with respect to N

$$t_N := \max_{\Delta \in \mathcal{L}(\mathcal{T}_N)} \tilde{e}(\Delta)\ . \tag{2.7}$$

Indeed, from (2.5) it follows that the the value $\tilde{e}(\Delta)$ for a node Δ is smaller than the value $\tilde{e}(\Delta^\star)$ for its parent $\Delta^\star$ and thus $\max\limits_{\Delta \in \mathcal{L}(\mathcal{T}_N)} \tilde{e}(\Delta) \geq \max\limits_{\Delta \in \mathcal{L}(\mathcal{T}_{N+1})} \tilde{e}(\Delta)$ since in the set $\mathcal{L}(\mathcal{T}_{N+1})$ two children nodes replace their parent which is in $\mathcal{L}(\mathcal{T}_N)$. Next, we consider a general binary tree T (not necessarily full) and a threshold t such that for all of its leaves $\tilde{e}(\Delta) \leq t$, $\Delta \in \mathcal{L}(T)$ and for all other nodes $\tilde{e}(\Delta) \geq t$, $\Delta \in (T \backslash \mathcal{L}(T))$. The following two results hold.

Lemma 2.3 *Let $t > 0$ and T be a general binary tree rooted at $\mathcal{R}$ and such that $\tilde{e}(\Delta) \leq t$ for all leaves $\Delta \in \mathcal{L}(\mathcal{T})$. Then*

$$\sum_{\Delta \in \mathcal{L}(\mathcal{T})} e(\Delta) \leq (\#T)t\,. \tag{2.8}$$

Proof. Given a leaf Δ we set $\Delta = \Delta^{(0)}$ and denote by $\Delta^{(j+1)}$ the parent of $\Delta^{(j)}$, $j = 0, 1, ..., \ell - 1$, where ℓ is such that $\Delta^{(\ell)} = \mathcal{R}$. Then by (2.5) and $\tilde{e}(\mathcal{R}) = e(\mathcal{R})$ we receive

$$\frac{1}{\tilde{e}(\Delta^{(0)})} = \frac{1}{e(\Delta^{(0)})} + \frac{1}{\tilde{e}(\Delta^{(1)})} = \frac{1}{e(\Delta^{(0)})} + \frac{1}{e(\Delta^{(1)})} + \frac{1}{\tilde{e}(\Delta^{(2)})} = ... = \sum_{j=0}^{\ell} \frac{1}{e(\Delta^{(j)})}\,. \tag{2.9}$$

Multiplying both parts of the equation by $e(\Delta^{(0)})\tilde{e}(\Delta^{(0)})$ we receive

$$e(\Delta^{(0)}) = \tilde{e}(\Delta^{(0)}) \sum_{j=0}^{\ell} \frac{e(\Delta^{(0)})}{e(\Delta^{(j)})} = \tilde{e}(\Delta^{(0)}) \left(1 + \sum_{j=1}^{\ell} \frac{e(\Delta^{(0)})}{e(\Delta^{(j)})}\right)\,.$$

Now denoting by $\mathcal{A}(\Delta)$ the set of ancestors of $\Delta = \Delta^{(0)}$ in the tree and using that $\tilde{e}(\Delta) \leq t$, we have

$$e(\Delta) \leq t \left(1 + \sum_{\Delta' \in \mathcal{A}(\Delta)} \frac{e(\Delta)}{e(\Delta')}\right)$$

and therefore

$$\sum_{\Delta\in\mathcal{L}(T)} e(\Delta) \le t \sum_{\Delta\in\mathcal{L}(T)} \left(1 + \sum_{\Delta'\in\mathcal{A}(\Delta)} \frac{e(\Delta)}{e(\Delta')}\right) = t\left(\#\mathcal{L}(T) + \sum_{\Delta'\in(T\setminus\mathcal{L}(T))} \frac{\sum_{\Delta\in(\mathcal{T}_{\Delta'}\cap\mathcal{L}(T))} e(\Delta)}{e(\Delta')}\right), \tag{2.10}$$

where in the derivation of the last expression we change the order of summation and take into account that the set of all leaves of T to which Δ' is an ancestor is exactly $\mathcal{T}_{\Delta'}\cap\mathcal{L}(T)$. Now we use that (1.3) yields (1.5) with $c = 1$ to conclude that each of the fractions in the sum over Δ' does not exceed 1. Thus, the sum is at most $\#(T\setminus\mathcal{L}(T))$ which proves the lemma since $\mathcal{L}(T)\subset T$. ∎

Lemma 2.4 *Let $t > 0$, ∇ be a node in a tree $\mathcal{T}$ for which e and $\tilde{e}$ are defined, and let T be a general binary subtree of $\mathcal{T}$ rooted at ∇ and such that $\tilde{e}(\Delta) \ge t$ for all nodes $\Delta\in T$. Then*

$$e(\nabla) \ge (\#T)t\,. \tag{2.11}$$

Proof. For any node Δ of T we have

$$e(\Delta) \ge \tilde{e}(\Delta) \ge t\,.$$

This gives the estimate in the case $\#T = 1$ and $\Delta = \nabla$. If, in addition, Δ has a parent node $\Delta^\star \in T$, then from (2.5) we get

$$e(\Delta) \ge t\left(\frac{e(\Delta)}{e(\Delta)} + \frac{e(\Delta)}{\tilde{e}(\Delta^\star)}\right) = t\left(1 + \frac{e(\Delta)}{\tilde{e}(\Delta^\star)}\right). \tag{2.12}$$

We are going to prove by induction that if for a node $\Delta'\in T$ the tree $\mathcal{T}_{\Delta'}$ rooted at Δ' has k' nodes in T, then

$$e(\Delta') \ge tk' \tag{2.13}$$

and in case Δ' has a parent $\Delta\in T$

$$e(\Delta') \ge t\left(k' + \frac{e(\Delta')}{\tilde{e}(\Delta)}\right). \tag{2.14}$$

Assume that $\Delta\in T$ has children $\Delta'\in T$ and $\Delta''\in T$ such that the trees $\mathcal{T}_{\Delta'}$ and $\mathcal{T}_{\Delta''}$ have k' and k'' nodes in T, correspondingly. Then, the tree $\mathcal{T}_\Delta$ rooted at Δ has $k := k' + k'' + 1$ nodes in T. Let (2.14) holds for both Δ' and Δ''. Adding the corresponding inequalities together gives

$$e(\Delta') + e(\Delta'') \ge t\left(k' + k'' + \frac{e(\Delta') + e(\Delta'')}{\tilde{e}(\Delta)}\right). \tag{2.15}$$

Multiplying both sides by $\frac{e(\Delta)}{e(\Delta')+e(\Delta'')} \ge 1$ we receive

$$e(\Delta) \ge t\left(k' + k'' + \frac{e(\Delta)}{\tilde{e}(\Delta)}\right) \ge t(k' + k'' + 1) = tk \tag{2.16}$$

to establish (2.13) for Δ and k. If Δ has a parent $\Delta^\star$, then we use (2.4) to get

$$e(\Delta) \geq t\left(k' + k'' + \frac{e(\Delta)}{e(\Delta)} + \frac{e(\Delta)}{\tilde{e}(\Delta^\star)}\right) = t\left(k + \frac{e(\Delta)}{\tilde{e}(\Delta^\star)}\right) \tag{2.17}$$

to establish (2.14) for Δ and k. In case $\Delta \in T$ has just one child $\Delta' \in T$, the considerations are the same as above, setting $k'' = 0$ and $e(\Delta'') = 0$ where appropriate. The induction argument completes the proof. ∎

Proof of Theorem 2.1. Let $\mathcal{T}_n^\star$ be a tree of best approximation for n, so $\mathcal{E}^{\mathrm{h}}(\mathcal{T}_n^\star) = \sigma_n^{\mathrm{h}}$. We want to compare the trees $\mathcal{T}_N$ and $\mathcal{T}_n^\star$. If $(\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star)) \subset (\mathcal{T}_N \backslash \mathcal{L}(\mathcal{T}_N))$, then $\mathcal{T}_n^\star \subset \mathcal{T}_N$ and therefore $\mathcal{E}^{\mathrm{h}}(\mathcal{T}_N) \leq \mathcal{E}^{\mathrm{h}}(\mathcal{T}_n^\star)$. So, we can assume that there is at least one internal node of $\mathcal{T}_n^\star$ that is not an internal node of $\mathcal{T}_N$. We use Lemma 2.4 to estimate $\mathcal{E}^{\mathrm{h}}(\mathcal{T}_n^\star) = \sigma_n^{\mathrm{h}}$ from below in terms of t_N from (2.6). To this end we consider the set F of all nodes in the trees T_∇ steming from leaves $\nabla \in \mathcal{L}(\mathcal{T}_n^\star)$ and having as nodes only the nodes from $\mathcal{T}_N \backslash \mathcal{L}(\mathcal{T}_N)$. Of course, some of the trees T_∇ could be empty. The total number of nodes from these trees is $\#F \geq (N-1) - (n-2) = N - n + 1$ since at list one node from $\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star)$ is not in $\mathcal{T}_N \backslash \mathcal{L}(\mathcal{T}_N)$. The application of (2.11) gives

$$\sigma_n^{\mathrm{h}} = \mathcal{E}^{\mathrm{h}}(\mathcal{T}_n^\star) = \sum_{\nabla \in \mathcal{L}(\mathcal{T}_n^\star)} e(\nabla) \geq \sum_{\nabla \in \mathcal{L}(\mathcal{T}_n^\star)} (\#T_\nabla) t_N = (\#F) t_N \geq (N - n + 1) t_N \,. \tag{2.18}$$

To estimate $\mathcal{E}^{\mathrm{h}}(\mathcal{T}_N)$ from above we divide the set of its leaves into two parts: the leaves that are nodes in $(\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))$ and the rest of them which combined errors can be estimated by $\mathcal{E}^{\mathrm{h}}(\mathcal{T}_n^\star) = \sigma_n^{\mathrm{h}}$

$$\mathcal{E}^{\mathrm{h}}(\mathcal{T}_N) = \sum_{\Delta \in \mathcal{L}(\mathcal{T}_N)} e(\Delta) \leq \sigma_n^{\mathrm{h}} + \sum_{\Delta \in [\mathcal{L}(\mathcal{T}_N) \cap (\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))]} e(\Delta) \,. \tag{2.19}$$

We apply Lemma 2.3 for the minimal tree T with leaves $\mathcal{L}(\mathcal{T}_N) \cap (\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))$ to receive

$$\sum_{\Delta \in [\mathcal{L}(\mathcal{T}_N) \cap (\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))]} e(\Delta) \leq \#(\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))\; t_N \leq \frac{n-1}{N-n+1}\, \sigma_n^{\mathrm{h}} \,, \tag{2.20}$$

where we have used that $T \subset (\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))$ and (2.18). Finally, the combination of (2.19) and (2.20) completes the proof of (2.6). ∎

3 Adaptive Strategy for hp-Refinements

Before formulating our hp-refinement strategy, we describe a recursive algorithm to find the subtree T_N for a given $\mathcal{T}_N$. We define the local hp-errors $E(\Delta) = E(\Delta, \mathcal{T}_N)$ starting with

$$E(\Delta) := e_1(\Delta) \quad \text{for} \quad \Delta \in \mathcal{L}(\mathcal{T}_N) \,.$$

If $E(\Delta')$ and $E(\Delta'')$ are already defined for the children Δ' and Δ'' of Δ, we set

$$E(\Delta) := \min\left\{E(\Delta') + E(\Delta''), e_{\mathcal{P}(\Delta)}(\Delta)\right\} \,, \tag{3.1}$$

where the order $\mathcal{P}(\Delta) = \mathcal{P}(\Delta, \mathcal{T}_N)$ is given by (1.9). To receive the tree T_N we start with $\mathcal{T}_N$ and trim it at Δ every time $E(\Delta) = e_{\mathcal{P}(\Delta)}(\Delta)$ in (3.1). In this way T_N becomes the minimal tree for which $E(\Delta) = e_{\mathcal{P}(\Delta)}(\Delta)$ at all its leaves.

An important observation about the dependence of quantities $E(\Delta) = E(\Delta, \mathcal{T}_N)$ from $\mathcal{T}_N$ is that it is based only on changes in the subtree rooted at Δ. Therefore, $E(\Delta, \mathcal{T}_N)$ will change with the increasing of N only if the quantity $\mathcal{P}(\Delta, \mathcal{T}_N)$ changes. It is then convenient to consider the notation $E_j(\Delta)$ independently of $\mathcal{T}_N$ setting that $E_j(\Delta) := E(\Delta, \mathcal{T}_N)$ whenever $j = \mathcal{P}(\Delta, \mathcal{T}_N)$.

As in the h-refinement case, we define via (2.4) the quantities $\tilde{e}(\Delta)$ based on the local errors $e(\Delta) = e_1(\Delta)$. These quantities are independent of $\mathcal{T}_N$ and give information about the local error at the node Δ only in the case Δ is a leaf of $\mathcal{T}_N$. To extend our ability to monitor the local error behavior in the process of finding a good hp-adaptive approximation, we define modified local hp-errors $\tilde{E}(\Delta)$ as follows

$$\tilde{E}_1(\Delta) := \tilde{e}(\Delta) \qquad \text{and} \qquad \tilde{E}_j(\Delta) := \frac{E_j(\Delta)\tilde{E}_{j-1}(\Delta)}{E_j(\Delta) + \tilde{E}_{j-1}(\Delta)} \quad \text{for} \quad j > 1\,. \tag{3.2}$$

In case both $E_j(\Delta)$ and $\tilde{E}_{j-1}(\Delta)$ are zeros, we define $\tilde{E}_j(\Delta) := 0$ as well. For a fixed tree $\mathcal{T}_N$ we set $j = \mathcal{P}(\Delta, \mathcal{T}_N)$ and consider $\tilde{E}(\Delta) := \tilde{E}(\Delta, \mathcal{T}_N) := \tilde{E}_{\mathcal{P}(\Delta, \mathcal{T}_N)}(\Delta)$.

Remark 3.1 *The independence of the quantities $E_j(\Delta)$ from the tree $\mathcal{T}_N$ is understood in the sense that the sequence $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots$ is predetermined by the local errors $e_p(\Delta)$ and a given hp-refinement strategy. If a different refinement strategy is applied, then it may result in different values of the $E_j(\Delta)$. This is one of the issues that makes the analysis of the hp-adaptive algorithm more complicated than the one from Section 2.*

The definition (3.2) of $\tilde{E}_j(\Delta)$ and (2.9) used with the set of ancestors $\mathcal{A}(\Delta)$ give

$$\frac{1}{\tilde{E}_j(\Delta)} = \frac{1}{E_j(\Delta)} + \frac{1}{\tilde{E}_{j-1}(\Delta)} = \sum_{k=2}^{j} \frac{1}{E_k(\Delta)} + \frac{1}{\tilde{E}_1(\Delta)} = \sum_{k=1}^{j} \frac{1}{E_k(\Delta)} + \sum_{\Delta' \in \mathcal{A}(\Delta)} \frac{1}{e(\Delta')}\,. \tag{3.3}$$

Next, we introduce two functions $q : \mathcal{T}_N \to \mathbb{R}_+$ and $s : \mathcal{T}_N \to \mathcal{L}(\mathcal{T}_N)$ that are critical for defining the hp-adaptive algorithm. For the leaves $\Delta \in \mathcal{L}(\mathcal{T})$ define

$$q(\Delta) := \tilde{e}(\Delta) = \tilde{E}_1(\Delta) \qquad \text{and} \qquad s(\Delta) := \Delta\,. \tag{3.4}$$

Recursively, for a node $\Delta \in \mathcal{T}_N \backslash \mathcal{L}(\mathcal{T}_N)$ with children Δ' and Δ'', for which q and s have been determined, define

$$q(\Delta) := \min\Big\{ \max\{q(\Delta'), q(\Delta'')\}, \tilde{E}_{\mathcal{P}(\Delta)}(\Delta) \Big\} \qquad \text{and} \qquad s(\Delta) := s(\operatorname{argmax}\{q(\Delta'), q(\Delta'')\})\,. \tag{3.5}$$

The principal algorithm for incrementally growing the tree $\mathcal{T}_N$ is the following:

- for $N = 1$ set $\mathcal{T}_N := \{\mathcal{R}\}$;
- while $N < N_{\max}$: for the tree $\mathcal{T}_N$ subdivide the leaf $s(\mathcal{R})$ to form $\mathcal{T}_{N+1}$ and set $N := N + 1$.

Before analyzing the near-best performance of the sequence $(\mathcal{T}_N, T_N)$, we first give a detail description of the algorithm to allow to account for its complexity.

hp-Algorithm:

(i) set $N = 1$, $\mathcal{T}_1 := \{\mathcal{R}\}$, $\tilde{e}(\mathcal{R}) := e(\mathcal{R})$, $E_1(\mathcal{R}) := e(\mathcal{R})$, $\tilde{E}_1(\mathcal{R}) := \tilde{e}(\mathcal{R})$, $q(\mathcal{R}) := \tilde{e}(\mathcal{R})$, $s(\mathcal{R}) := \mathcal{R}$;

(ii) expand the current tree $\mathcal{T}_N$ to $\mathcal{T}_{N+1}$ by subdividing $\Delta_N := s(\mathcal{R})$ and adding two children nodes Δ'_N and Δ''_N to it;

(iii) for $\nabla = \Delta'_N$ and $\nabla = \Delta''_N$ calculate the quantities: $\tilde{e}(\nabla) := \frac{e(\nabla)\tilde{e}(\Delta)}{e(\nabla)+\tilde{e}(\Delta)}$, $E_1(\nabla) := e(\nabla)$, $\tilde{E}_1(\nabla) := \tilde{e}(\nabla)$, $q(\nabla) := \tilde{e}(\nabla)$, $s(\nabla) := \nabla$;

(iv) set $\Delta := \Delta_N$;

(v) set $N := N + 1$; **exit** if $N \geq N_{\max}$

(vi) set $\mathcal{P}(\Delta) := \mathcal{P}(\Delta) + 1$ and calculate $e_{\mathcal{P}(\Delta)}(\Delta)$;

(vii) set Δ' and Δ'' to be the children of Δ;

(viii) set $E_{\mathcal{P}(\Delta)}(\Delta) := \min\{E_{\mathcal{P}(\Delta')}(\Delta') + E_{\mathcal{P}(\Delta'')}(\Delta'')\ ,\ e_{\mathcal{P}(\Delta)}(\Delta)\}$;

(ix) set $\tilde{E}_{\mathcal{P}(\Delta)}(\Delta) := \frac{E_{\mathcal{P}(\Delta)}(\Delta)\tilde{E}_{\mathcal{P}(\Delta)-1}(\Delta)}{E_{\mathcal{P}(\Delta)}(\Delta)+\tilde{E}_{\mathcal{P}(\Delta)-1}(\Delta)}$;

(x) set $\mathcal{D} := \operatorname{argmax}\{q(\Delta'), q(\Delta'')\}$ and $q(\Delta) := \min\left\{q(\mathcal{D}), \tilde{E}_{\mathcal{P}(\Delta)}(\Delta)\right\}$, $s(\Delta) := s(\mathcal{D})$;

(xi) if $\Delta \neq \mathcal{R}$, replace Δ with its parent and **go to** (vi);

(xii) **go to** (ii);

Lemma 3.2 *To obtain* $(\mathcal{T}_N, T_N)$ *the* **hp-Algorithm** *performs* $\sum_{\Delta\in\mathcal{T}_N} \mathcal{P}(\Delta, \mathcal{T}_N)$ *steps.*

Proof. In analysis of the algorithm one can see that $\mathcal{P}(\Delta, \mathcal{T}_{N+1}) = \mathcal{P}(\Delta, \mathcal{T}_N) + 1$ for all nodes Δ, for which the quantities E, $\tilde{E}$, q, and s are updated. ***to be substantiated a bit more*** ∎

Theorem 3.3 *The pair* $(\mathcal{T}_N, T_N)$ *produced by the* **hp-Algorithm** *provides near-best approximation and satisfies* (1.11) *for any positive integer* $n \leq N$.

Proof. Let the pair $(T_n^\star, \mathcal{P}^\star)$ be the one providing the optimal error of complexity n in (1.7) and such that $\sigma_n = \mathcal{E}(T_n^\star, \mathcal{P}^\star)$. We define the threshold parameter $q_N := q(\mathcal{R})$ for the tree $\mathcal{T}_N$. From the fact that the quantities involved in the definition of $q(\Delta)$ decrease in the process of growing the trees $\mathcal{T}_k$ it follows that the quantities q_k are decreasing with k.

To estimate σ_n from below we consider the leaves $\nabla \in \mathcal{L}(T_n^\star)$ and their orders $\mathcal{P}^\star(\nabla)$. In case $\mathcal{P}(\nabla, \mathcal{T}_N) \leq \mathcal{P}^\star(\nabla)$ we ignore the contribution of $e_{\mathcal{P}^\star(\nabla)}(\nabla)$ to the $\mathcal{E}(T_n^\star, \mathcal{P}^\star)$. If $\mathcal{P}(\nabla, \mathcal{T}_N) > \mathcal{P}^\star(\nabla)$, we consider the quantity $q_k \geq q_N$ at the stage $\mathcal{T}_k$ of growing the tree $\mathcal{T}_N$ at which the last decision to subdivide a descendent of ∇ was made. From the definitions of q and s it follows that at this stage $q(\nabla) \geq q_k$ and therefore $\tilde{E}_j(\nabla) \geq q(\nabla) \geq q_N$ for $j = \mathcal{P}(\nabla, \mathcal{T}_N) - 1$. From the intermediate relation in (3.3) we have

$$\frac{1}{\tilde{E}_j(\nabla)} = \sum_{i=\mathcal{P}^\star(\nabla)+1}^{j} \frac{1}{E_i(\nabla)} \; + \; \frac{1}{\tilde{E}_{\mathcal{P}^\star(\nabla)}(\nabla)} \, .$$

Multiplying by $\tilde{E}_j(\nabla) E_{\mathcal{P}^\star(\nabla)}(\nabla)$ and taking into account that quantities $E_i(\nabla)$ are monotone decreasing with i and that $\tilde{E}_i(\nabla) \leq E_i(\nabla)$, we receive

$$E_{\mathcal{P}^\star(\nabla)}(\nabla) = \tilde{E}_j(\nabla) \left(\sum_{i=\mathcal{P}^\star(\nabla)+1}^{j} \frac{E_{\mathcal{P}^\star(\nabla)}(\nabla)}{E_i(\nabla)} \; + \; \frac{E_{\mathcal{P}^\star(\nabla)}(\nabla)}{\tilde{E}_{\mathcal{P}^\star(\nabla)}(\nabla)} \right) \geq q_N (j - \mathcal{P}^\star(\nabla) + 1)$$

and therefore

$$e_{\mathcal{P}^\star(\nabla)}(\nabla) \geq E_{\mathcal{P}^\star(\nabla)}(\nabla) \geq q_N \left(\mathcal{P}(\nabla, \mathcal{T}_N) - \mathcal{P}^\star(\nabla) \right)_+ \ , \tag{3.6}$$

where we have used the standard notation $(x)_+ := \max\{x, 0\}$. Before applying this inequality for the estimate of σ_n, we exclude the case that $\mathcal{P}(\nabla, \mathcal{T}_N) \geq \mathcal{P}^\star(\nabla)$ for all $\nabla \in \mathcal{L}(T_n^\star)$ since then $E_{\mathcal{P}(\nabla,\mathcal{T}_N)} \leq E_{\mathcal{P}^\star(\nabla)} \leq e_{\mathcal{P}^\star(\nabla)}(\nabla)$ and therefore $\mathcal{E}(\mathcal{T}_N, T_N) \leq \mathcal{E}(T_n^\star, \mathcal{P}^\star) = \sigma_n$ which gives (1.11). For notational purposes only, we set $\mathcal{P}^\star(\nabla) := 1$ for all $\nabla \in (T_n^\star \backslash \mathcal{L}(T_n^\star))$ to derive

$$\begin{aligned}
\sum_{\nabla \in \mathcal{L}(T_n^\star)} (\mathcal{P}(\nabla, \mathcal{T}_N) - \mathcal{P}^\star(\nabla))_+ &= \sum_{\nabla \in \mathcal{L}(T_n^\star \cap \mathcal{T}_N)} (\mathcal{P}(\nabla, \mathcal{T}_N) - \mathcal{P}^\star(\nabla))_+ \\
> \sum_{\nabla \in \mathcal{L}(T_n^\star \cap \mathcal{T}_N)} (\mathcal{P}(\nabla, \mathcal{T}_N) - \mathcal{P}^\star(\nabla)) &= N - \sum_{\nabla \in \mathcal{L}(T_n^\star \cap \mathcal{T}_N)} \mathcal{P}^\star(\nabla) \geq N - n \, ,
\end{aligned} \tag{3.7}$$

where we have used that $(\mathcal{P}(\nabla, \mathcal{T}_N) - \mathcal{P}^\star(\nabla))_+ = 0$ in case ∇ is in the symmetric difference of the sets $\mathcal{L}(T_n^\star)$ and $\mathcal{L}(T_n^\star \cap \mathcal{T}_N)$, as well as

$$\sum_{\nabla \in \mathcal{L}(T_n^\star \cap \mathcal{T}_N)} \mathcal{P}^\star(\nabla) \leq \sum_{\nabla \in \mathcal{L}(T_n^\star)} \mathcal{P}^\star(\nabla) = n \, .$$

Now the combination of (3.6) and (3.7) gives

$$\sigma_n = \sum_{\nabla \in \mathcal{L}(T_n^\star)} e_{\mathcal{P}^\star(\nabla)}(\nabla) \geq q_N \sum_{\nabla \in \mathcal{L}(T_n^\star)} (\mathcal{P}(\nabla, \mathcal{T}_N) - \mathcal{P}^\star(\nabla))_+ \geq q_N (N - n + 1) \, . \tag{3.8}$$

To obtain an estimate of $\mathcal{E}(\mathcal{T}_N, T_N)$ from above, we consider the function $q(\Delta)$ for the tree $\mathcal{T}_N$ and denote by L the set of nodes Δ for which $q(\Delta) = \tilde{E}_{\mathcal{P}(\Delta)}$ in (3.5). Let $\mathcal{Q}$ be the maximal subtree (not necessarily full) of $\mathcal{T}_N$ for which $L \cap \mathcal{Q} = \mathcal{L}(\mathcal{Q})$. Then $q(\Delta) = \tilde{E}_{\mathcal{P}(\Delta)}$ for all leaves $\Delta \in \mathcal{L}(\mathcal{Q})$. From the procedure of defining q it follows that for all these leaves $q(\Delta) \leq q(\mathcal{R}) = q_N$. To estimate $E_{\mathcal{P}(\Delta)}$ for $\Delta \in \mathcal{L}(\mathcal{Q})$ we multiply both sides of (3.3) by $\tilde{E}_{\mathcal{P}(\Delta)} E_{\mathcal{P}(\Delta)}$ to receive for $j = \mathcal{P}(\Delta, \mathcal{T}_N)$

$$E_j(\Delta) = \tilde{E}_j(\Delta) \left(\sum_{k=1}^{j} \frac{E_j(\Delta)}{E_k(\Delta)} + \sum_{\Delta' \in \mathcal{A}(\Delta)} \frac{E_j(\Delta)}{e(\Delta')} \right) \leq q_N \left(j + \sum_{\Delta' \in \mathcal{A}(\Delta)} \frac{e(\Delta)}{e(\Delta')} \right) \tag{3.9}$$

using that $E_k(\Delta)$ are monotone decreasing and $E_1(\Delta) = e(\Delta)$. Utilizing that T_N is the optimal subtree in terms of total hp-error the inequality (3.9) gives

$$\mathcal{E}(\mathcal{T}_N, T_N) \le \mathcal{E}(\mathcal{T}_N, \mathcal{Q}) = \sum_{\Delta\in\mathcal{L}(Q)} E_{\mathcal{P}(\Delta,\mathcal{T}_N)}(\Delta) \le q_N \left(\sum_{\Delta\in\mathcal{L}(Q)} \mathcal{P}(\Delta, \mathcal{T}_N) + \sum_{\Delta\in\mathcal{L}(Q)} \sum_{\Delta'\in\mathcal{A}(\Delta)} \frac{e(\Delta)}{e(\Delta')} \right) .$$

Applying the same estimate as in (2.10) in Lemma 2.3 to the double sum gives

$$\mathcal{E}(\mathcal{T}_N, T_N) \le q_N(N - \#\mathcal{L}(\mathcal{Q}) + \#\mathcal{Q}) = q_N(N + \#\mathcal{L}(\mathcal{Q}) - 1) \le q_N(2N - 1) , \tag{3.10}$$

which concludes the proof of (1.11). ■

Theorem 1.2 now follows from Theorem 3.3 and Lemma 3.2. ■

Remark 3.4 *The estimate* (3.10) *is a bit rough since it was derived treating in the same way all possible subtrees $\mathcal{Q}$ including the worst case scenario $\mathcal{Q} = \mathcal{T}_N$. However, in this particular case one could derive a much better estimate using that the p-option was never taken in the calculation of the quantities q and applying an argument similar to the one for* (2.19)*. Further exploration of such ideas and thorough analysis of the relative placement of the trees $\mathcal{Q}$ and $T_n^\star$ will result in a slightly better constant in* (1.11) *but would complicate significantly the proof. Since the advances are marginal, we have chosen clarity.*

Still to come:

- adding a paragraph or two about different strategies for hp-adaptivity and their realizations
- adding citations
- polishing of the text

Acknowledgment: The author wish to thank Ricardo Nochetto for the fruitful discussions and the various valuable suggestions.

References

[1] P. Binev, *Adaptive Methods and Near-Best Tree Approximation*, Oberwolfach Report **29/2007**, 1669-1673.

[2] P. Binev, *Instance optimality for hp-type approximation*, Oberwolfach Report **39/2013**, 3p., to appear.

[3] P. Binev, W. Dahmen, and R. DeVore, *Adaptive Finite Element Methods with Convergence Rates*, Numer. Math. **97** (2004), 219-268.

[4] P. Binev and R. DeVore, *Fast Computation in Adaptive Tree Approximation*, Numer. Math. **97** (2004), 193-217.

Peter Binev
Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA
binev@math.sc.edu