SIAM J. NUMER. ANAL.
Vol. 56, No. 6, pp. 3346–3357

TREE APPROXIMATION FOR hp-ADAPTIVITY*

PETER BINEV†

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 orders $p(\Delta)$ does not exceed N, which is called the complexity of the approximation. The leaves Δ correspond to the cells of the partition, while $p(\Delta)$ is the dimension of the polynomial space 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 $\mathcal{T}$ 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 the error of our approximation of complexity N is bounded by $\frac{2N-1}{N-n+1}\sigma_n$, where σ_n, $n \leq N$, is the error of the best possible approximation of complexity n.

Key words. adaptive methods, hp-adaptivity, tree based algorithms, near-best approximation, instance optimality

AMS subject classifications. 41A15, 41A63, 65D15, 65M55, 68Q32, 68W25, 97N50

DOI. 10.1137/18M1175070

1. Introduction.

There are two basic approaches to adaptivity when approximating a function on a domain Ω. The first considers the current partition of Ω and chooses some of its elements Δ for refinement. The polynomial space used for local approximation does not change, and the improvement of the approximation is based on decreasing the size h of the elements Δ. Thus, this is usually called an *h-refinement*. The second approach considers the same partition of Ω, but the dimensionality $p(\Delta)$ of the local approximation space 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-adaptive*. 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. The results of this paper have already been used and cited (see, e.g., [5, 6, 7, 8]).

In general, the 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 it is no longer a leaf. To simplify 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 call Δ' and Δ'' *children* of

*Received by the editors March 12, 2018; accepted for publication (in revised form) September 27, 2018; published electronically November 29, 2018.

http://www.siam.org/journals/sinum/56-6/M117507.html

Funding: This work was supported by NSF grants DMS 1222390 and DMS 1720297.

†Department of Mathematics, University of South Carolina, Columbia, SC 29208 (binev@math.sc.edu, http://people.math.sc.edu/binev/).

Δ, which will be referred to 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 $p = p(\Delta)$ is the dimension of the polynomial space used in the approximation at Δ. It is important to emphasize that we want to work with additive quantities describing the local errors. For a given partition defined by T and the assignments

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

we define the total error $\mathcal{E}(T,\mathcal{P}(T))$ by

$$\mathcal{E}(T,\mathcal{P}(T)) := \sum_{\Delta\in\mathcal{L}(T)} e_{p(\Delta)}(\Delta)\,. \tag{1.1}$$

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 $e_p(\Delta)$, $p \geq 1$, requiring only the following two properties:

(i) subadditivity of the error for the lowest order approximation:

$$e_1(\Delta) \geq e_1(\Delta') + e_1(\Delta'')\,, \tag{1.2}$$

where Δ' and Δ'' are the children of Δ;

(ii) reduction of the error when the dimension $p = k$ of the polynomial space increases:

$$e_k(\Delta) \geq e_{k+1}(\Delta)\,. \tag{1.3}$$

In some cases (e.g., spline approximation defined via quasi-interpolants) it is convenient and often necessary to consider a less demanding variant of (1.2) known as weak subadditivity:

$$e_1(\Delta) \geq c \sum_{\Delta'\in(\mathcal{T}_\Delta\cap T)} e_1(\Delta')\,, \tag{1.4}$$

where $c > 0$ is a fixed constant independent of the choice of Δ or the full binary 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 (1.4) holds with $c = 1$ in case (1.2) is true. While the proofs presented below can be modified to use (1.4) instead of (1.2), this will result in additional complications and less favorable constants. To keep the presentation simple, we choose to use the property (1.2) 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 $p(\Delta)$ at all the elements of the partition

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

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

$$\sigma_N := \inf_{T} \inf_{\#\mathcal{P}(T)\le N} \mathcal{E}(T,\mathcal{P}(T))\,. \tag{1.5}$$

The aim of this paper is to define an *incremental* algorithm, similar to the one in [4], that for given error assignments satisfying (1.2) and (1.3) produces for each N a pair $(T_N,\mathcal{P}(T_N))$ of a tree T_N and the corresponding dimensions $\mathcal{P}(T_N)$ of local polynomial spaces with $\#\mathcal{P}(T_N) = N$ such that it provides a near-best hp-adaptive approximation, namely

$$\mathcal{E}(T_N,\mathcal{P}(T_N)) \le C_1\sigma_{c_2N}$$

with some fixed constants $C_1 \ge 1$ and $c_2 \in (0,1]$.

First, we have to define a framework that allows us 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. To receive $\mathcal{T}$ from T_N, at each leaf $\Delta\in\mathcal{L}(T_N)$ we add a "ghost" tree Υ_Δ rooted at Δ that has exactly $p(\Delta)$ leaves (Υ_Δ is just Δ in the case $p(\Delta) = 1$). After attaching all such trees to the leaves of T_N we compose the tree $\mathcal{T}$. Of course, the tree $\mathcal{T}$ is not uniquely defined since there are different ways of choosing the ghost trees Υ_Δ, in general. However, a specific choice of Υ_Δ can carry information on what the approximation would look like if we decide to use polynomial spaces of lower dimensions at some locations while keeping the same complexity.

Given $\mathcal{T}$, for each of the nodes $\Delta\in\mathcal{T}$ we define its order by

$$\mathcal{P}(\Delta) := \mathcal{P}(\Delta,\mathcal{T}) := \#\Big(\mathcal{L}(\mathcal{T})\cap\mathcal{T}_\Delta\Big)\,. \tag{1.6}$$

By the definition of $\mathcal{T}$ it follows that $\mathcal{P}(\Delta) = p(\Delta)$ for $\Delta\in\mathcal{L}(T_N)$, but we also can associate $\mathcal{T}$ with any other full subtree $T\subset\mathcal{T}$ and obtain the pair $(T,\mathcal{P}(T))$ from (1.6). Now, the hp-approximation can be identified with the pair $(\mathcal{T},T)$ instead of the pair $(T,\mathcal{P}(T))$, and we set $\mathcal{E}(\mathcal{T},T) = \mathcal{E}(T,\mathcal{P}(T))$ as defined in (1.1). It is easy to see that the error of the best approximation from (1.5) can be expressed as

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

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 thereby minimize the computational cost.

Remark 1.1. Note that the inf in the definitions (1.5) and (1.7) 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, 3]). 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 (see [2]). We describe our approach in section 3 and prove the following result.

THEOREM 1.2. *Let the local errors e_p satisfy the conditions* (1.2) *and* (1.3). *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 child 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 of complexity N provided by the pair $(\mathcal{T}_N, T_N)$ is near-best, namely,*

$$\mathcal{E}(\mathcal{T}_N, T_N) \leq \frac{2N-1}{N-n+1}\ \sigma_n \tag{1.8}$$

for any integer $n \leq N$. The complexity of the algorithm for obtaining $(\mathcal{T}_N, T_N)$ is bounded by $\mathcal{O}\big(\sum_{\Delta\in\mathcal{T}_N} \mathcal{P}(\Delta, \mathcal{T}_N)\big)$, where $\mathcal{P}(\Delta, \mathcal{T}_N)$ is defined by (1.6).

Remark 1.3. The sum $\sum_{\Delta\in\mathcal{T}_N} \mathcal{P}(\Delta, \mathcal{T}_N)$ 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. The complexity of a single calculation of e_p, for each consecutive p, is considered to be a constant independent of p. This could happen, e.g., in the case when e_p is the square of the L_2-error and the local polynomial spaces are defined via given orthogonal bases.

2. Near-best results for an h-refinement strategy. The setup in the case of h-adaptive approximation is much simpler. In particular, the local approximations use 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 (1.2):

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

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

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

and the best N-term h-adaptive approximation error is

$$\sigma_N^{\mathrm{h}} := \min_{\mathcal{T}\,:\,\#\mathcal{L}(\mathcal{T})\leq N} \mathcal{E}^{\mathrm{h}}(\mathcal{T})\,. \tag{2.2}$$

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

$$\tilde{e}(\mathcal{R}) := e(\mathcal{R}) \quad \text{for the root } \mathcal{R} \text{ and} \qquad \tilde{e}(\Delta) := \frac{e(\Delta)\tilde{e}(\Delta^\star)}{e(\Delta) + \tilde{e}(\Delta^\star)}\,, \tag{2.3}$$

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 $\tilde{e}(\Delta)$:

$$\frac{1}{\tilde{e}(\Delta)} = \frac{1}{e(\Delta)} + \frac{1}{\tilde{e}(\Delta^\star)}\,. \tag{2.4}$$

The modified error depends only on the set of ancestors 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 improves the constants from a similar result in [4] based on a different approximation algorithm. Its proof illustrates some of the ideas used in the hp-adaptive case and establishes some properties of the modified error functionals.

THEOREM 2.1. *Let the local errors* $e(\Delta)$ *satisfy the condition* (2.1) *and let the tree* $\mathcal{T}_N$ *be received by applying a greedy refinement strategy with respect to the quantities* $\tilde{e}(\Delta)$ *defined by* (2.3). *Then the tree* $\mathcal{T}_N$ *provides a near-best h-adaptive approximation error*

$$\mathcal{E}^{\mathrm{h}}(\mathcal{T}_N) \le \frac{N}{N-n+1}\ \sigma_n^{\mathrm{h}} \tag{2.5}$$

for any integer $n \le N$. *The complexity of the algorithm for obtaining* $\mathcal{T}_N$ *is* $\mathcal{O}(N)$, *omitting 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 nonempty bin accumulating the largest values available. This will only increase the constant in (2.5) 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 nonincreasing with respect to N:

$$t_N := \max_{\Delta \in \mathcal{L}(\mathcal{T}_N)} \tilde{e}(\Delta)\,. \tag{2.6}$$

Indeed, from (2.4) it follows that 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 $\max_{\Delta\in\mathcal{L}(\mathcal{T}_N)} \tilde{e}(\Delta) \ge \max_{\Delta\in\mathcal{L}(\mathcal{T}_{N+1})} \tilde{e}(\Delta)$ since in the set $\mathcal{L}(\mathcal{T}_{N+1})$ two child 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 $\Delta \in \mathcal{L}(T)$ we have $\tilde{e}(\Delta) \le t$ and for all of its internal nodes $\Delta \in (T \backslash \mathcal{L}(T))$, we have $\tilde{e}(\Delta) \ge t$. The following two results hold.

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

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

Proof. Given a leaf Δ we set $\Delta = \Delta^{(0)}$ and denote by $\Delta^{(j+1)}$ the parent of $\Delta^{(j)}$, $j = 0, 1, \ldots, \ell - 1$, where ℓ is such that $\Delta^{(\ell)} = \mathcal{R}$. Then by (2.4) and $\tilde{e}(\mathcal{R}) = e(\mathcal{R})$ we obtain

$$\begin{aligned}\frac{1}{\tilde{e}(\Delta^{(0)})} &= \frac{1}{e(\Delta^{(0)})} + \frac{1}{\tilde{e}(\Delta^{(1)})} = \frac{1}{e(\Delta^{(0)})} + \frac{1}{e(\Delta^{(1)})} + \frac{1}{\tilde{e}(\Delta^{(2)})} \\ &= \cdots = \sum_{j=0}^{\ell} \frac{1}{e(\Delta^{(j)})}\,.\end{aligned} \tag{2.7}$$

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

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

Now denoting by $\mathcal{A}(\Delta) = \{\Delta^{(1)}, \Delta^{(2)}, \ldots, \Delta^{(\ell)}\}$ the set of ancestors of $\Delta = \Delta^{(0)}$ in the tree and using that $\tilde{e}(\Delta) \le t$, we have

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

and therefore

$$\begin{aligned} \sum_{\Delta \in \mathcal{L}(T)} e(\Delta) &\le t \sum_{\Delta \in \mathcal{L}(T)} \left(1 + \sum_{\Delta' \in \mathcal{A}(\Delta)} \frac{e(\Delta)}{e(\Delta')} \right) \\ &= t \left(\#\mathcal{L}(T) + \sum_{\Delta' \in (T \setminus \mathcal{L}(T))} \frac{\sum_{\Delta \in (\mathcal{T}_{\Delta'} \cap \mathcal{L}(T))} e(\Delta)}{e(\Delta')} \right), \end{aligned} \tag{2.8}$$

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 (1.2) yields (1.4) 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 $t > 0$, let ∇ be a node in a tree $\mathcal{T}$, and let T be a general binary subtree of $\mathcal{T}$ rooted at ∇ such that $\tilde{e}(\Delta) \ge t$ for all nodes $\Delta \in T$. Then*

$$e(\nabla) \ge (\#T)t\,. \tag{2.9}$$

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 (2.4) we get

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

We are going to prove by induction on $k = k'$ that if $\#(\mathcal{T}_{\Delta'} \cap T) = k'$ for a node $\Delta' \in T$, then

$$e(\Delta') \ge tk' \tag{2.10}$$

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

$$e(\Delta') \ge t \left(k' + \frac{e(\Delta')}{\tilde{e}(\Delta)} \right). \tag{2.11}$$

The above inequalities have already been established for $k = 1$. Assume that (2.10) and (2.11) have been established for all $k' < k$, and let $\Delta \in T$ be such that the tree $\mathcal{T}_\Delta$ rooted at Δ has $k > 1$ nodes in T. Since $k > 1$, Δ has one or two 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, $k = k' + k'' + 1$. In case $\Delta \in T$ has just one child $\Delta' \in T$, we set $k'' = 0$ and $e(\Delta'') = 0$ in the considerations below. By the induction hypothesis, (2.11) holds for both Δ' and Δ''. Adding the corresponding inequalities together gives

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

Multiplying both sides by $\frac{e(\Delta)}{e(\Delta')+e(\Delta'')} \geq 1$, we obtain

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

to establish (2.10) for Δ and k. If Δ has a parent $\Delta^\star$, then we use (2.4) to get

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

to establish (2.11) for Δ and k. The induction argument completes the proof. ∎

Proof of Theorem 2.1. Let $\mathcal{T}_n^\star$ be a tree of best approximation for n, and thus $\mathcal{E}^{\rm h}(\mathcal{T}_n^\star) = \sigma_n^{\rm h}$. 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 $\mathcal{E}^{\rm h}(\mathcal{T}_N) \leq \mathcal{E}^{\rm h}(\mathcal{T}_n^\star)$. 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 $\mathcal{E}^{\rm h}(\mathcal{T}_n^\star) = \sigma_n^{\rm h}$ from below in terms of t_N from (2.6). To this end we consider the set $F := (\mathcal{T}_N \backslash \mathcal{L}(\mathcal{T}_N)) \backslash (\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))$. The total number of nodes of F is $\#F \geq (N-1) - (n-2) = N - n + 1$ since at least 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 set F can be considered as the union of the trees $T_\nabla = \mathcal{T}_\nabla \cap (\mathcal{T}_N \backslash \mathcal{L}(\mathcal{T}_N))$ for $\nabla \in \mathcal{L}(\mathcal{T}_n^\star)$. The tree T_∇ is empty if ∇ is not an internal node of $\mathcal{T}_N$; otherwise it consists of ∇ itself and all the internal nodes of $\mathcal{T}_N$ that are its descendants. The application of (2.9) gives

$$\begin{aligned} \sigma_n^{\rm h} = \mathcal{E}^{\rm h}(\mathcal{T}_n^\star) &= \sum_{\nabla \in \mathcal{L}(\mathcal{T}_n^\star)} e(\nabla) \\ &\geq \sum_{\nabla \in \mathcal{L}(\mathcal{T}_n^\star)} (\#T_\nabla) t_N = (\#F) t_N \geq (N - n + 1) t_N \, . \end{aligned} \tag{2.12}$$

To estimate $\mathcal{E}^{\rm h}(\mathcal{T}_N)$ 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, whose combined errors can be estimated by $\mathcal{E}^{\rm h}(\mathcal{T}_n^\star) = \sigma_n^{\rm h}$, namely

$$\mathcal{E}^{\rm h}(\mathcal{T}_N) = \sum_{\Delta \in \mathcal{L}(\mathcal{T}_N)} e(\Delta) \leq \sigma_n^{\rm h} + \sum_{\Delta \in [\mathcal{L}(\mathcal{T}_N) \cap (\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))]} e(\Delta) \, . \tag{2.13}$$

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 obtain

$$\sum_{\Delta \in [\mathcal{L}(\mathcal{T}_N) \cap (\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))]} e(\Delta) \leq \#(\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star)) \; t_N \leq \frac{n-1}{N-n+1} \; \sigma_n^{\rm h} \, , \tag{2.14}$$

where we have used the fact that $T \subset (\mathcal{T}_n^\star \backslash \mathcal{L}(\mathcal{T}_n^\star))$ and (2.12). Finally, the combination of (2.13) and (2.14) completes the proof of (2.5). ∎

3. Adaptive strategy for hp-refinements. Before formulating our variant of 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 (1.6). 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 quantities $E(\Delta) = E(\Delta, \mathcal{T}_N)$ is that they depend on $\mathcal{T}_N$ only through the 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 $\mathcal{T}_N$ by setting $E_j(\Delta) := E(\Delta, \mathcal{T}_*)$, where $\mathcal{T}_*$ is any of the trees in the sequence $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \ldots$ such that $\mathcal{P}(\Delta, \mathcal{T}_*) = j$.

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 make the analysis of the hp-adaptive algorithm more complicated than the analysis from section 2.

As in the h-refinement case, we define via (2.3) 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 when 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 $\tilde{E}_j(\Delta)$ 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 $\tilde{E}(\Delta) := \tilde{E}(\Delta, \mathcal{T}_N) := \tilde{E}_{\mathcal{P}(\Delta, \mathcal{T}_N)}(\Delta)$.

The definition (3.2) of $\tilde{E}_j(\Delta)$ and (2.7) used with the set of ancestors $\mathcal{A}(\Delta)$ give

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

Next, we introduce two functions $q : \mathcal{T}_N \to [0, \infty)$ and $s : \mathcal{T}_N \to \mathcal{L}(\mathcal{T}_N)$ that are critical for defining the hp-adaptive algorithm. The quantity $q(\nabla)$ is related to the maximal modified hp-error of the subtree rooted at ∇, while $s(\nabla)$ points to the leaf with the largest contribution for this error. For the leaves $\Delta \in \mathcal{L}(\mathcal{T}_N)$ we 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, we define

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

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

- set $N = 1$ and $\mathcal{T}_1 := \{\mathcal{R}\}$;
- while $N < N_{\max}$: given 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 detailed description of the algorithm to list all the necessary computations and comparisons in growing the tree in order to estimate its complexity.

hp-algorithm:

(i) set $\mathcal{T}_1 := \{\mathcal{R}\}$, $\tilde{e}(\mathcal{R}) := e(\mathcal{R})$, $E_1(\mathcal{R}) := e(\mathcal{R})$, $\tilde{E}_1(\mathcal{R}) := \tilde{e}(\mathcal{R})$, $q(\mathcal{R}) := \tilde{e}(\mathcal{R})$, $s(\mathcal{R}) := \mathcal{R}$, $\mathcal{P}(\mathcal{R}) := 1$;
(ii) **for** $N = 1$ **to** $N_{\max} - 1$
(iii) expand the current tree $\mathcal{T}_N$ to $\mathcal{T}_{N+1}$ by subdividing $\Delta_N := s(\mathcal{R})$ and adding two child nodes Δ'_N and Δ''_N to it;
(iv) for $\nabla = \Delta'_N$ and $\nabla = \Delta''_N$ calculate the quantities: $\tilde{e}(\nabla) := \frac{e(\nabla)\tilde{e}(\Delta)}{e(\nabla)+\tilde{e}(\Delta)}$, $E_1(\nabla) := e(\nabla)$, $\tilde{E}_1(\nabla) := \tilde{e}(\nabla)$, $q(\nabla) := \tilde{e}(\nabla)$, $s(\nabla) := \nabla$, $\mathcal{P}(\nabla) := 1$;
(v) set $\Delta := \Delta_N$;
(vi) **while** $\Delta \neq \emptyset$
(vii) set $\mathcal{P}(\Delta) := \mathcal{P}(\Delta) + 1$ and calculate $e_{\mathcal{P}(\Delta)}(\Delta)$;
(viii) set Δ' and Δ'' to be the children of Δ;
(ix) set $E_{\mathcal{P}(\Delta)}(\Delta) := \min\{E_{\mathcal{P}(\Delta')}(\Delta') + E_{\mathcal{P}(\Delta'')}(\Delta''),\ e_{\mathcal{P}(\Delta)}(\Delta)\}$;
(x) set $\tilde{E}_{\mathcal{P}(\Delta)}(\Delta) := \frac{E_{\mathcal{P}(\Delta)}(\Delta)\tilde{E}_{\mathcal{P}(\Delta)-1}(\Delta)}{E_{\mathcal{P}(\Delta)}(\Delta)+\tilde{E}_{\mathcal{P}(\Delta)-1}(\Delta)}$;
(xi) set $\mathcal{D} := \operatorname{argmax}\{q(\Delta'), q(\Delta'')\}$ and update $q(\Delta) := \min\big\{q(\mathcal{D}), \tilde{E}_{\mathcal{P}(\Delta)}(\Delta)\big\}$, $s(\Delta) := s(\mathcal{D})$;
(xii) replace Δ with its parent (or $\emptyset$ if $\Delta = \mathcal{R}$);
(xiii) **end while**
(xiv) **end for**

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. The algorithm has two loops: an outer loop including all instructions in (ii)–(xiv), and an inner loop (vi)–(xiii). The outer loop runs for each consecutive increment of N from 1 to $N_{\max} - 1$, and the inner loop performs the calculations at the nodes of the tree starting from the newly subdivided node Δ and then proceeding with all elements of its ancestry line $\mathcal{A}(\Delta)$ of Δ. The quantities $\mathcal{P}(\Delta)$ are initialized as 1 at (i) or (iv) and then increased in the inner loop by 1 assigning $\mathcal{P}(\Delta, \mathcal{T}_{N+1}) = \mathcal{P}(\Delta, \mathcal{T}_N) + 1$ for all nodes Δ for which the quantities E, $\tilde{E}$, q, and s are updated. Therefore $\mathcal{P}(\Delta, \mathcal{T}_N)$ keeps the count of the calculations performed at Δ. ∎

THEOREM 3.3. *The pair* $(\mathcal{T}_N, T_N)$ *produced by the* hp-algorithm *provides near-best approximation and satisfies* (1.8) *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 (1.5) 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 $\mathcal{P}(\nabla, \mathcal{T}_N) \le \mathcal{P}^\star(\nabla)$ 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 $q_k \ge q_N$ at the stage $\mathcal{T}_k$ of growing the tree $\mathcal{T}_N$ at the last update of $\mathcal{P}(\nabla)$. From the definitions of q and s it follows that at this stage $q(\nabla) \ge q_k$ and therefore $\tilde{E}_j(\nabla) \ge q(\nabla) \ge q_N$ 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 the quantities $E_i(\nabla)$ are nonincreasing with i and that $\tilde{E}_i(\nabla) \le E_i(\nabla)$, we obtain

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

and therefore

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

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 $\mathcal{P}(\nabla, \mathcal{T}_N) \ge \mathcal{P}^\star(\nabla)$ for all $\nabla \in \mathcal{L}(T_n^\star)$ since then $E_{\mathcal{P}(\nabla, \mathcal{T}_N)}(\nabla) \le E_{\mathcal{P}^\star(\nabla)}(\nabla) \le e_{\mathcal{P}^\star(\nabla)}(\nabla)$ and therefore $\mathcal{E}(\mathcal{T}_N, T_N) \le \mathcal{E}(T_n^\star, \mathcal{P}^\star) = \sigma_n$ which gives (1.8). This ensures the sign $>$ in (3.7) below. 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

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

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

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

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

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

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) and (3.4). Let $\mathcal{Q}$ be the maximal subtree of $\mathcal{T}_N$ for which $L \cap \mathcal{Q} = \mathcal{L}(\mathcal{Q})$. The tree

$\mathcal{Q}$ has to be a full tree since $q(\Delta) = \tilde{E}_1(\Delta)$ for all leaves $\Delta \in \mathcal{L}(\mathcal{T}_N) \subset L$. 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 $q(\Delta) \le q(\mathcal{R}) = q_N$. 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 obtain for $j = \mathcal{P}(\Delta, \mathcal{T}_N)$

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

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

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

Applying the same estimate as in (2.8) in the proof of Lemma 2.3 to the double sum gives

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

using that $\#\mathcal{Q} - \#\mathcal{L}(\mathcal{Q}) \le N - 1$ since the internal nodes of Q are internal nodes of $\mathcal{T}_N$. This concludes the proof of (1.8). ∎

Proof of Theorem 1.2. The proof follows directly 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 the fact that the p-option was never taken in the calculation of the quantities q and applying an argument similar to the one for (2.13). 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 (1.8) but would significantly complicate the proof. Since the advances are marginal, we have chosen clarity.

Acknowledgment. The author wishes to thank Ricardo Nochetto for several fruitful discussions and various valuable suggestions.

REFERENCES

[1] P. Binev, *Adaptive methods and near-best tree approximation*, Oberwolfach Rep., 29 (2007), pp. 1669–1673.

[2] P. Binev, *Instance optimality for hp-type approximation*, Oberwolfach Rep., 39 (2013), pp. 2192–2194.

[3] P. Binev, W. Dahmen, and R. DeVore, *Adaptive finite element methods with convergence rates*, Numer. Math., 97 (2004), pp. 219–268.

[4] P. Binev and R. DeVore, *Fast computation in adaptive tree approximation*, Numer. Math., 97 (2004), pp. 193–217.

[5] C. Canuto, R. Nochetto, R. Stevenson, and M. Verani, *High-order adaptive Galerkin methods*, in Spectral and High Order Methods for Partial Differential Equations: ICOSAHOM 2014, R. M. Kirby, M. Berzins, and J. S. Hesthaven, eds., Springer, Cham, 2015, pp. 51–72.

[6] C. Canuto, R. Nochetto, R. Stevenson, and M. Verani, *Convergence and optimality of hp-AFEM*, Numer. Math., 135 (2017), pp. 1073–1119.
[7] C. Canuto, R. Nochetto, R. Stevenson, and M. Verani, *On p-robust saturation for hp-AFEM*, Comput. Math. Appl., 73 (2017), pp. 2004–2022.
[8] P. Daniel, A. Ern, I. Smears, and M. Vohralík, *An adaptive hp-refinement strategy with computable guaranteed bound on the error reduction factor*, Comput. Math. Appl., 76 (2018), pp. 967–983.