Numer. Math. (2004) 97: 193–217
Digital Object Identifier (DOI) 10.1007/s00211-003-0493-6

Numerische
Mathematik

Fast computation in adaptive tree approximation

Peter Binev, Ronald DeVore

Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA;
e-mail: {binev,devore}@math.sc.edu

Received June 10, 2002 / Revised version received April 29, 2003 /
Published online January 8, 2004 – © Springer-Verlag 2004

Summary. Adaptive methods of approximation arise in many settings including numerical methods for PDEs and image processing. They can usually be described by a tree which records the adaptive decisions. This paper is concerned with the fast computation of near optimal trees based on n adaptive decisions. The best tree based on n adaptive decisions could be found by examining all such possibilities. However, this is exponential in n and could be numerically prohibitive. The main result of this paper is to show that it is possible to find near optimal trees using computations linear in n.

Mathematics Subject Classification (2000): 65Y20, 65N50, 41A63, 41A15, 68W40, 68W25

1 Introduction

Tree approximation is a form of nonlinear approximation in which the approximants are required to have a certain structure described by trees. It was introduced and studied in [2] in the context of n-term wavelet approximation and the application of this form of approximation to image compression. Wavelets are naturally indexed on the set $\mathcal{D}$ of dyadic cubes which has a tree structure determined by set inclusion. In this setting, tree approximation with n terms seeks to approximate a given target function f by a linear combination of n wavelets whose indices from $\mathcal{D}$ are required to align themselves on

This work has been supported in part by the Office of Naval Research contracts 03-10051, (N00014-00-1-0470), the Army Research Office contract DAAD 19-02-1-0028, and the National Science Foundation grants DMS 9872890, DMS 0221642.
Correspondence to: R. DeVore

a tree. The tree structure gives an efficient way to encode the positions of the wavelets appearing in the sum. This form of approximation is more restrictive than the usual n-term approximation by wavelets where the n wavelets used in forming the approximation are completely arbitrary. Both of these forms of approximation are nonlinear since the n-terms may depend on f. The paper [2] gives results on the error of n-term tree approximation and shows how these can be used to build efficient encoders for image compression.

Tree approximation also occurs naturally in the construction of adaptive methods for approximation. Given a domain Ω and a target function f defined on Ω, we approximate f by dividing Ω into a partition P of n cells and using piecewise polynomials (of some fixed degree) subordinate to this partition. These cells are obtained by some adaptive process which decides when and how a cell should be divided. Thus, the partition P can be described by a tree which indicates the adaptive divisions that have occurred.

The usual application of this second form of tree approximation is to adaptive numerical methods for PDEs. This application is quite different from image processing. Indeed, in image processing based on wavelet expansions, the target function and all of its wavelet coefficients can be considered to be known and the only problem is how to organize a good approximation with the required tree structure. In the PDE setting however, the target function is unknown and information about it can only be obtained through certain numerical queries. This information is usually given in the form of bounds on the local approximation error through the residual. In Adaptive Finite Element Methods (AFEM) the local error bounds are called a posteriori error estimators and are a central issue in building good adaptive solvers.

However, even in AFEM, the adaptive approximation of known target functions arises and must be numerically handled in an efficient manner. For example, in the numerical resolution of elliptic equations, the right hand side of such an equation is known but must be efficiently approximated numerically. Also methods using coarsening of partitions will encounter a similar problem. In fact, our motivation for the present paper occurred in our study of AFEM and the results of this paper are used to build a numerical AFEM with proven convergence rates in [1].

We could find a best tree approximation to f with n interior nodes by forming all possible trees with n interior nodes and examining the global error associated to each tree. We would choose the tree which gives the smallest error bound. However, there are exponential in n such trees so this method is computationally prohibitive when n is large. The main point of the present paper is to show that under appropriate settings, it is possible to create near best adaptive n term approximations using only $O(n)$ computations.

In §3 we shall give a precise description of the setting for this paper and formulate our main results. In the sections that follow, we shall introduce two algorithms for attaining the performance we want. The first algorithm works under more restrictive assumptions than the second but may be more intuitive in its structure. In the last two sections we give examples of how to apply these algorithms in settings such as those mentioned above for AFEM.

2 Adaptive partitioning

In this section, we shall describe the setting of adaptive refinement that arises in adaptive numerical methods for PDEs. We shall see that such adaptive partitioning can be described by trees.

Let Ω be a polygonal domain in $I\!R^d$. By a partition P for Ω we shall mean a collection of simplices Δ such that

$$\bigcup_{\Delta\in P} \Delta = \Omega, \tag{2.1}$$

and

$$\text{meas}(\Delta \cap \Delta') = 0 \text{ for any pair of different simplices } \Delta, \Delta' \in P, \tag{2.2}$$

where meas denotes Lebesgue measure in $I\!R^d$.

The adaptive algorithms of interest to us are based on subdividing certain of the simplices in P. The choice of which simplices to subdivide is adaptive and based on data dependent criteria. However the rule for subdividing the simplices is fixed in advance. Namely, we assume that given any simplex Δ, this rule gives a collection $\Delta_j \subset \Delta$, $j = 1, \dots, m(\Delta)$, with $m(\Delta) \geq 2$, of simplices which partition Δ. The simplices $\Delta_1, \dots, \Delta_{m(\Delta)}$ are called the children of Δ and Δ is their parent. For example, in the case $d = 2$, one such rule might be to subdivide a given triangle into four triangles by using the bisectors to each side. There are many other possibilities. However, we stress the important fact that the decision on which cells will be subdivided is data dependent but how they are subdivided is not allowed to be data dependent in the setting of this paper.

Let P be a partition of Ω. An *elementary refinement* P' of P is the replacement of one simplex $\Delta \in P$ by the K new simplices Δ_j, $j = 1, ..., K$, given by the subdivision rule. All other simplices in P remain untouched.

An *adaptive partitioning* consists of a series of elementary refinements. Starting with an initial partition P_0, a set of simplices M_0 are marked for elementary refinement. The new partition P_1 is obtained by replacing each marked cell by its elementary refinement but leaving all other cells untouched. We continue with this process of marking cells and refining and thereby generate the partitions $P_1, \dots, P_m$ such that each P_j is the refinement of P_{j-1}, $j = 1, \dots, m$. The complexity of the final partition P_m is determined by the number of marked cells, i.e. the cardinality of $n := \#(\cup_{j=0}^{m-1} M_j)$, which is the same as the number of elementary refinements that have taken place.

Each adaptively generated partition P can be represented by a tree $T = T(P)$. The nodes in this tree T are the simplices of P_j, $j = 0, \dots, m$. The *roots* of the tree are the simplices Δ in P_0. The set $\mathcal{L}(T)$ of leaves of the tree T are the simplices in the final partition P_m. Recall that the leaves of a tree T are the final nodes, i.e. the nodes in T which have no children in T.

The trees $T(P)$ which arise in adaptive partitioning are all finite subtrees of an infinite tree T_* which we call the master tree. The tree T_* has as its nodes all of the possible simplices Δ that can arise in the refinement process. This idea of a master tree which delineates the finite subtrees that can arise in an adaptive algorithm occurs in more general contexts such as trees that arise in numerically based wavelet methods. Therefore, in going further in this paper, we shall consider adaptive approximation in this more general tree setting as described in the following sections.

3 Adaptive tree approximation

In this section, we shall describe the general setting of tree approximation. We assume that we have in hand an infinite tree T_* (called the *master tree*) with a finite number of root nodes and with the property that each node Δ of T_* has $m(\Delta)$ children with $2 \le m(\Delta) \le K$. We allow the number of children to vary from node to node. Although in most applications $m(\Delta)$ does not depend on Δ.

A subtree T of T_* is a collection of nodes $\Delta \in T_*$ such that whenever Δ, Δ' are in T and $\Delta' \neq \Delta$ is a descendent of Δ then all of the children of Δ are also in T. Notice that this requires that whenever Δ is in T then all of the siblings of Δ are also in T. This condition, not always assumed for a tree, is compatible with the trees that arise in adaptive methods.

A subtree T is said to be *proper* if T contains all of the root nodes of T_*. The set $\mathcal{L}(T)$ of leaves of T are all of the nodes in T for which none of their children are in T. All other nodes of T are called *interior nodes*. The number of interior nodes of T will be denoted by $N(T)$. This is also the number of subdivisions used to create T. We shall frequently use the following remark:

$$\#(\mathcal{L}(T)) \le \#(T) \le 2\#(\mathcal{L}(T)). \tag{3.1}$$

Similarly, in the case that all root nodes of T_* are interior nodes of T, we have

$$N(T) \le \#(T) \le (K+1)N(T) \tag{3.2}$$

and

$$\#(\mathcal{L}(T)) \le K\ N(T). \tag{3.3}$$

The left inequality in (3.1) is obvious. The right inequality follows by induction on $\#(T)$. Indeed, if we grow a tree T by adding the children of one of its leaves Δ, then the number of nodes in T increases by $m(\Delta)$ and the number of leaves grows by $m(\Delta) - 1 \geq 1$. Since $m(\Delta) \leq 2(m(\Delta) - 1)$ we arrive at (3.1). A similar reasoning gives (3.2) and (3.3). In particular, one can use that for any tree $T \subset T_*$

$$\#(T) \leq N_0 + KN(T), \tag{3.4}$$

where N_0 is the number of root nodes of T_*.

We assume that we are given a functional e which associates to each node Δ of T_* a nonnegative real number $e(\Delta)$. Given any proper subtree T of T_* we define

$$E(T) := \sum_{\Delta \in \mathcal{L}(T)} e(\Delta). \tag{3.5}$$

The reader should think of e as a bound for the local approximation error and E a bound for the global approximation error to a given function f . For example, in the case of piecewise polynomial approximation on a simplicial decomposition, given a function $f \in L_p(\Omega)$, $1 \leq p < \infty$, we could define $e(\Delta)$ to be the p-th power of the error of best $L_p(\Delta)$-approximation to f by polynomials of some fixed degree r. Then $E(T)$ is the p-th power of the global error in approximating f in $L_p(\Omega)$ by piecewise polynomials of degree r with no continuity assumptions on the piecewise polynomials across the faces of the simplices of $\mathcal{L}(T)$. Another setting that occurs when considering piecewise polynomial approximation with continuity conditions across the boundaries of the simplices is that $e(\Delta)$ is the p-th power of the error of best $L_p(\hat{\Delta})$-approximation to f by polynomials of some fixed degree r with $\hat{\Delta}$ a local domain which contains Δ. The overlapping of the cells $\hat{\Delta}$ causes some difficulties which will be overcome in the theory that we develop.

Given, the functionals e and E, and an arbitrary integer $n > 0$, consider the class $\mathcal{T}_n$ of all proper trees T with $N(T) = n$ and define

$$E_n := \min_{T \in \mathcal{T}_n} E(T). \tag{3.6}$$

Thus, E_n measures the best error we could obtain by utilizing trees generated by using n subdivisions. The main question we wish to study in this paper is to what extent is it possible to find adaptive algorithms which achieve approximation order like $E_n(P)$. Since the number of trees in $\mathcal{T}_n$ is exponential with respect to n, finding the best tree T_n^* which achieves the error E_n could be a very costly operation. We shall show that there are constants[1] $C_1, C_2 > 0$

[1] We shall denote constants by the generic C and they may vary at each occurrence. Constants that are important in going further in the paper will be denoted by $C_1, c_1, C_2, \ldots$.

and numerically effective algorithms to adaptively find trees $T \in \mathcal{T}_{C_1 n}$ such that

$$E(T) \le C_2 E_n, \tag{3.7}$$

where the constants C_1 and C_2 depend only on the integer K. We call such an algorithm *near optimal.*

We shall measure the numerical *cost* of an adaptive algorithm by the number of computations used in the algorithm where the cost of computing $e(\Delta)$ for any given Δ is taken to be one. In the next two sections, we propose two algorithms to find a near best approximant for each n in a numerically effective way. Namely, the cost of each of these algorithms is linear in n. The first algorithm, given in the next section, is more restrictive in the assumptions it makes on e. Also, the first algorithm depends on a thresholding parameter which is eliminated in the second algorithm.

4 The first algorithm

In this section, we shall assume that the functional e satisfies the following

Refinement property *If Δ is any node of T_* and $\Delta_1, \dots, \Delta_{m(\Delta)}$ are its children then*

$$\sum_{i=1}^{m(\Delta)} e(\Delta_i) \le e(\Delta). \tag{4.1}$$

Our first algorithm will be based on a threshold parameter $t > 0$. Let T_* be a master tree and let e be a functional defined on the nodes of T_* satisfying (4.1) and let E be defined by (3.5). The basic idea of the first algorithm is to subdivide those nodes Δ for which $e(\Delta)$ exceeds the threshold t. But this procedure may fail since we are not guaranteed that such subdivisions actually effectively decrease the error. Thus, we shall modify the adaptive criteria to include a penalty term.

The intuitive idea behind the first algorithm can be explained by considering the functional e as specifying a certain energy at each node. The inequality (4.1) says that the energy decreases (strictly speaking does not increase) when a node is divided into its children. We shall modify e to obtain a new functional $\tilde{e}$ in order to guarantee that the modified energy $\tilde{e}$ lost in such a subdivision is at least t.

Given $\Delta \in T_*$, we let $S(\Delta)$ denote the set of its siblings (i.e. the collection of all $\Delta' \in T$ which have the same parent as Δ). The quantity

$$\lambda(\Delta) := \frac{e(\Delta)}{\sum_{\Delta' \in S(\Delta)} e(\Delta')} \tag{4.2}$$

will be used in the case the denominator is positive. It measures the portion of the energy in $S(\Delta)$ that is carried by Δ. Note that $\sum_{\Delta' \in S(\Delta)} \lambda(\Delta') = 1$ for each Δ.

Let $t > 0$ be the thresholding parameter. For each $\Delta \in T_*$ with children $\Delta_1, \dots, \Delta_{m(\Delta)}$, we define

$$d(\Delta) := e(\Delta) - \sum_{i=1}^{m(\Delta)} e(\Delta_i) =: e(\Delta) - \sigma(\Delta) \tag{4.3}$$

and

$$\delta(\Delta) := \begin{cases} 0, & \text{if } d(\Delta) \geq t, \\ t - d(\Delta), & \text{if } d(\Delta) < t. \end{cases} \tag{4.4}$$

The quantity $\delta(\Delta)$ is the amount of artificial energy we will take away at each node subdivision.

With the above notation in hand, we now introduce a modified functional $\tilde{e}$ defined on the nodes of T_* by an expression of the form

$$\tilde{e}(\Delta) = e(\Delta) - \alpha(\Delta), \tag{4.5}$$

We define $\alpha(\Delta) = 0$ for all root nodes and whenever $\alpha(\Delta)$ has been defined, then we define $\alpha(\Delta')$ for each child Δ' of Δ by:

$$\alpha(\Delta') := \lambda(\Delta')\alpha(\Delta) + \lambda(\Delta')\delta(\Delta). \tag{4.6}$$

The first term in the definition of $\alpha(\Delta)$ represents the portion of the artificial energy already removed that needs to be attributed to this node and the second term is the new energy to be removed because of the subdivision of Δ. Let us observe for later use:

Remark 4.1 If Δ' is a child of Δ then

$$\tilde{e}(\Delta') = e(\Delta') - \alpha(\Delta') = e(\Delta')\left(1 - \frac{\alpha(\Delta) + \delta(\Delta)}{\sigma(\Delta)}\right), \tag{4.7}$$

so that if one of the children Δ' of Δ has $\tilde{e}(\Delta')$ positive then all other children Δ'' will have $\tilde{e}(\Delta'')$ nonnegative.

The first algorithm will recursively generate trees $T_0, \dots, T_k$ by growing. The tree T_0 consists of the set of root nodes of T_*. The remaining T_{k+1} are generated from the previous T_k by using the following criteria.

First Algorithm *For each $\Delta \in \mathcal{L}(T_k)$, compute $\tilde{e}(\Delta)$. Whenever $\tilde{e}(\Delta) > t$, we add all of the children of Δ to T_k. Then T_{k+1} consists of T_k together with all of these children. The algorithm terminates when $T_{k+1} = T_k$, i.e. when $\tilde{e}(\Delta) \leq t$ for all $\Delta \in \mathcal{L}(T_k)$. The final tree is denoted by T.*

The most important property of this algorithm is that the sum of modified errors $\tilde{e}$ looses *locally* a quantity of t each time a node is subdivided (see (4.11) below). To analyze **First Algorithm**, we shall need the following lemmas.

Lemma 4.2 *Let $t > 0$ and T be the tree associated to the* **First Algorithm**. *If Δ is a node in the interior of T and T_Δ is the subtree consisting of all $\Delta' \in T$ such that Δ' is a descendent of Δ, then*

$$\frac{\#(T_\Delta)t}{K+1} \le N(T_\Delta)t \le \tilde{e}(\Delta) \le e(\Delta). \tag{4.8}$$

Proof. The left inequality in (4.8) is (3.2) and the right inequality is obvious since $\alpha(\Delta)$ is nonnegative. Therefore, we need only prove the second inequality. From (4.5) and (4.6), it follows that for each $\Delta' \in T_*$ with the children $\Delta'_1, \dots, \Delta'_{m(\Delta')}$, we have

$$\sum_{i=1}^{m(\Delta')} \tilde{e}(\Delta'_i) = \sum_{i=1}^{m(\Delta')} e(\Delta'_i) - \alpha(\Delta') - \delta(\Delta') \tag{4.9}$$

and therefore

$$\begin{aligned} \tilde{e}(\Delta') - \sum_{i=1}^{m(\Delta')} \tilde{e}(\Delta'_i) &= e(\Delta') - \alpha(\Delta') - \sum_{i=1}^{m(\Delta')} e(\Delta'_i) + \alpha(\Delta') + \delta(\Delta') \\ &= d(\Delta') + \delta(\Delta'). \end{aligned} \tag{4.10}$$

Since $d(\Delta') + \delta(\Delta') \ge t$, we obtain

$$\sum_{i=1}^{m(\Delta')} \tilde{e}(\Delta'_i) \le \tilde{e}(\Delta') - t. \tag{4.11}$$

In other words, each subdivision reduces the energy $\tilde{e}$ by at least an amount t.

We define Υ_0 to be the subtree of T_Δ obtained by deleting from T_Δ all leaves $\Delta' \in \mathcal{L}(T_\Delta)$ for which it and all of its siblings are in $\mathcal{L}(T_\Delta)$.

If we apply (4.11) starting at the bottom of Υ_0 and moving to the top root Δ, we obtain

$$\sum_{\Delta' \in \mathcal{L}(\Upsilon_0)} \tilde{e}(\Delta') + tN(\Upsilon_0) \le \tilde{e}(\Delta). \tag{4.12}$$

The sum on the left side of (4.12) is over two types of Δ'. The first are those Δ' which are interior nodes in T_Δ. Since these are further subdivided by the algorithm, they satisfy $\tilde{e}(\Delta') \ge t$. There are precisely $N(T_\Delta) - N(\Upsilon_0)$ nodes of this type. The second type of node satisfies $\tilde{e}(\Delta') \ge 0$ because of Remark 4.1. It follows that the sum on the left side of (4.12) is $\ge t(N(T_\Delta) - N(\Upsilon_0))$. When this is used in (4.12), we arrive at the second inequality in (4.8). □

Lemma 4.3 *Let $t > 0$ and let T be the tree associated to the* **First Algorithm**. *If Δ is a root node of T and T_Δ is any subtree of T with single root node Δ, then*

$$\sum_{\Delta' \in \mathcal{L}(T_\Delta)} e(\Delta') \le \sum_{\Delta' \in \mathcal{L}(T_\Delta)} \tilde{e}(\Delta') + N(T_\Delta)t. \tag{4.13}$$

Proof. If $N(T_\Delta) = 0$, that is $T_\Delta = \{\Delta\}$ then (4.13) is obvious because $e(\Delta) = \tilde{e}(\Delta)$. Therefore, in going further, we shall only consider the case where $N(T_\Delta) \ge 1$. From the definition of $\tilde{e}$ in (4.5), we have

$$\sum_{\Delta' \in \mathcal{L}(T_\Delta)} e(\Delta') = \sum_{\Delta' \in \mathcal{L}(T_\Delta)} \tilde{e}(\Delta') + \sum_{\Delta' \in \mathcal{L}(T_\Delta)} \alpha(\Delta'). \tag{4.14}$$

We complete the proof by showing that

$$\sum_{\Delta' \in \mathcal{L}(T_\Delta)} \alpha(\Delta') \le N(T_\Delta)t \tag{4.15}$$

by using induction on $N(T_\Delta)$. In the trivial case $N(T_\Delta) = 0$ we have $\mathcal{L}(T_\Delta) = \{\Delta\}$ and $e(\Delta) = \tilde{e}(\Delta)$. When $N(T_\Delta) = 1$, T_Δ consists solely of Δ and its children and each of these children is a leaf of T_Δ. Since $\alpha(\Delta) = 0$ and for each of the children Δ' of Δ, we have

$$\alpha(\Delta') = \lambda(\Delta')(\alpha(\Delta) + \delta(\Delta)) \le t\lambda(\Delta')$$

because $\delta(\Delta) \le t$. Summing over Δ' and using that $\sum_{\Delta'} \lambda(\Delta') = 1$, we arrive at (4.15) in this case.

Suppose now that (4.15) holds for all trees T_Δ with $N(T_\Delta) = k$ and consider a tree T_Δ with $N(T_\Delta) = k + 1$. Let Δ' be a node whose children $\Delta'_1, \dots, \Delta'_{m(\Delta')}$ are leaves of T_Δ. The tree T'_Δ which is obtained from T_Δ by deleting all of the children of Δ' satisfies $N(T'_\Delta) = k$ and hence (4.15) holds for T'_Δ. We need only show that

$$\sum_{i=1}^{m(\Delta')} \alpha(\Delta'_i) \le \alpha(\Delta') + t. \tag{4.16}$$

But $\alpha(\Delta'_i) = \lambda(\Delta'_i)(\alpha(\Delta')+\delta(\Delta'))$ and $\delta(\Delta') \le t$. Since $\sum_{i=1}^{m(\Delta')} \lambda(\Delta'_i) = 1$, we have (4.16) and have completed the proof of the lemma. □

Theorem 4.4 *Let $t > 0$ be any threshold parameter for which the* **First Algorithm** *gives a final tree T. Then,*

$$E(T) \le 2(K + 1)E_m \tag{4.17}$$

where $m := [N(T)/2]$. The algorithm uses at most $\#(T) \le N_0 + K\,N(T)$ computations of e in generating T.

Proof. The statement concerning the number of computations of e is clear from (3.4). To prove (4.17), we let T^* be a tree of order m which satisfies $E(T^*) = E_m$. We consider three disjoint sets Λ_0, Λ_1, Λ_2 of root nodes. The first set Λ_0 consists of all root nodes that are not subdivided in either T or T^*. The second set Λ_1 consists of all root nodes which are interior nodes in T^* but are not subdivided in T. The last set Λ_2 consists of all remaining root nodes; each of these root nodes is an interior node in T. For each node Δ, we let T_Δ be the set of all $\Delta' \in T$ that are descendants of Δ (such a tree may be empty, if $\Delta \notin T$). We have that

$$E(T) = \sum_{\Delta\in\Lambda_0} e(\Delta) + \sum_{\Delta\in\Lambda_1} e(\Delta) + \sum_{\Delta\in\Lambda_2} \sum_{\Delta'\in\mathcal{L}(T_\Delta)} e(\Delta'). \tag{4.18}$$

To estimate $E(T)$, we set $E(\Lambda_0) := \sum_{\Delta\in\Lambda_0} e(\Delta)$ for the first sum in (4.18) and use Lemma 4.3 to each T_Δ in the other two sums to find

$$\begin{aligned} E(T) &\le E(\Lambda_0) + \sum_{\Delta'\in\Lambda_1} \tilde{e}(\Delta') + \sum_{\Delta\in\Lambda_2} \sum_{\Delta'\in\mathcal{L}(T_\Delta)} \tilde{e}(\Delta') + N(T)t \\ &\le E(\Lambda_0) + \#(\Lambda_1)t + (K+1)N(T)t \end{aligned} \tag{4.19}$$

where the last inequality uses the fact that $\tilde{e}(\Delta') \le t$ for each $\Delta' \in \mathcal{L}(T)$ and that the total number of elements in $\cup_{\Delta\in\Lambda_2}\mathcal{L}(T_\Delta)$ can not exceed $K\ N(T)$ because each such Δ' is a child of an interior node of T.

To estimate E_m from below we consider subtrees T_Δ for nodes $\Delta \in \mathcal{L}(T^*)\backslash\Lambda_0$ which are not descendants of a root node from Λ_1. We apply Lemma 4.2 to each such T_Δ and then sum over Δ to obtain (note that $N(T_\Delta) = 0$ for all other $\Delta \in \mathcal{L}(T^*)$)

$$E_m = \sum_{\Delta\in\mathcal{L}(T^*)} e(\Delta) \ge E(\Lambda_0) + \sum_{\Delta\in\mathcal{L}(T^*)} N(T_\Delta)\, t \tag{4.20}$$

The fact that any interior node of T, which is not an interior node for T^*, will be in one of the T_Δ gives that the sum in (4.20) is at least $N(T) - (N(T^*) - \#(\Lambda_1))$. From the definition of m we have $N(T) \ge 2N(T^*)$, and therefore

$$\begin{aligned} &E(\Lambda_0) + \#(\Lambda_1)t + N(T)t/2 \\ &\quad\le E(\Lambda_0) + \big(N(T) - (N(T^*) - \#(\Lambda_1))\big)\, t \le E_m\,. \end{aligned} \tag{4.21}$$

A combination of (4.19) and (4.21) gives (4.17). □

5 The second algorithm

In this section, we present a second algorithm for adaptive tree approximation. Its main advantage is that it weakens the subadditivity condition (4.1) by replacing it with the following subadditivity assumption:

Subadditivity *For each $\Delta \in T_*$ and any finite tree $T_\Delta \subset T_*$ with single root node Δ, we have*

$$\sum_{\Delta' \in \mathcal{L}(T_\Delta)} e(\Delta') \le C_0 e(\Delta) \tag{5.1}$$

with $C_0 \ge 1$ an absolute constant.

This will be important in Finite Element applications where the error functional e corresponds to local approximation on sets $\hat{\Delta}$ which overlap. We shall actually show in §7 that this subadditivity condition can be weakened even further so as to be applicable in general finite element settings.

As was the case in the first algorithm, we shall introduce a modified error functional $\tilde{e}$ which will drive the algorithm. Initially, for all of the root nodes of T_*, we define $\tilde{e}(\Delta) = e(\Delta)$. Now, assuming that $\tilde{e}(\Delta)$ has been defined for a $\Delta \in T_*$, we define $\tilde{e}$ for each child Δ_j, $j = 1, \dots, m(\Delta)$, of Δ as

$$\tilde{e}(\Delta_j) := q(\Delta) \tag{5.2}$$

where

$$q(\Delta) := \frac{\sum_{j=1}^{m(\Delta)} e(\Delta_j)}{e(\Delta) + \tilde{e}(\Delta)} \tilde{e}(\Delta). \tag{5.3}$$

In other words, $\tilde{e}$ is constant on the children of Δ. It is useful to define the *penalty* terms

$$p(\Delta_j) := \frac{e(\Delta_j)}{\tilde{e}(\Delta_j)} \tag{5.4}$$

which describes how $\tilde{e}$ differs from e. Note that the $p(\Delta_j)$ satisfy the equality

$$\sum_{j=1}^{m(\Delta)} p(\Delta_j) = p(\Delta) + 1\,. \tag{5.5}$$

which is the main property we shall need for $\tilde{e}$.

Let us note that it follows from (5.5) that for any proper subtree $T \subset T_*$ we have

$$\sum_{\Delta \in \mathcal{L}(T)} p(\Delta) = N_0 + N(T) \tag{5.6}$$

where N_0 is the number of root nodes of T_* and $N(T)$ is as before the number of subdivisions used to create T. This follows by induction on $N(T)$. Similarly if T_{Δ^*} is a tree with a single root node Δ^* then

$$\sum_{\Delta \in \mathcal{L}(T_{\Delta^*})} p(\Delta) = p(\Delta^*) + N(T_{\Delta^*}). \tag{5.7}$$

Second Algorithm *This algorithm creates a sequence of trees $T = T_j$, $j = 1, 2, \dots$ as follows. For $j = 0$, T_0 is the collection of root nodes of T_*. If T_{j-1} has been defined, we examine all leaves of T_{j-1} and subdivide the leaves Δ with largest $\tilde{e}(\Delta)$ to produce T_j. (In the case there are several largest, we subdivide all of them.)*

Remark 5.1 Note that $\#(T_j) \le K\#(T_{j-1})$ for each $j = 1, 2, \dots$

The following theorem which is the analogue of Theorem 4.4, analyzes the performance of the second algorithm.

Theorem 5.2 *There is an absolute constant $C > 0$ such that for each $j = 0, 1, \dots,$ the output tree $T = T_j$ of the* **Second Algorithm** *satisfies*

$$E(T) \le CE_m \tag{5.8}$$

whenever $m \le n/(2K + 2)$ and $n := N(T)$. To create T, the algorithm uses $\le C(n + N_0)$ arithmetic operations and computations of e, where N_0 is the number of root nodes of T_*.

Proof. The proof of this theorem has many aspects in common with Theorem 4.4. We consider any $j = 0, 1, \dots$ such that $T = T_j$ satisfies $N(T) =: n \ge 2(K+1)$ and we let T^* be the best tree with $N(T^*) = m$ with $m \le n/(2K+2)$. We shall also assume that each root node has been subdivided in the creation of either T or T^*. If this is not the case, one would derive the theorem for the new master tree obtained by deleting the root nodes not divided and then derive the result for the original tree from this.

Let t be the smallest value attained by $\tilde{e}$ on the interior nodes of T. We let $\Lambda := \{\Delta : \tilde{e}(\Delta) = t\}$ be the collection of interior nodes of T where $\tilde{e}$ takes on the value t. We derive an upper bound for $E(T) = \sum_{\Delta \in \mathcal{L}(T)} e(\Delta)$ by breaking T into $T = \Upsilon_0 \cup \Upsilon_1$ where $\Upsilon_1 := \cup_{\Delta \in \Lambda} T_\Delta$ and for each Δ, T_Δ is the subtree of T consisting of Δ and all of its descendants in T. The tree Υ_0 is obtained by deleting all the proper descendants of all of the $\Delta \in \Lambda$ (i.e. we leave each of the $\Delta \in \Lambda$ in Υ_0 but remove all of their descendants). It follows that $\mathcal{L}(T) = \mathcal{L}_0 \cup \mathcal{L}_1$ where $\mathcal{L}_1 = \mathcal{L}(\Upsilon_1)$ and $\mathcal{L}_0 = \mathcal{L}(\Upsilon_0) \setminus \Lambda$. This gives

$$E(T) = \sum_{\Delta \in \mathcal{L}_0} e(\Delta) + \sum_{\Delta \in \mathcal{L}_1} e(\Delta) = \Sigma_0 + \Sigma_1. \tag{5.9}$$

We will bound each of the two sums appearing on the right side of (5.9). First, we note that from the subadditivity (5.1), it follows that for each $\Delta \in \Lambda$, we have

$$\sum_{\Delta' \in \mathcal{L}(T_\Delta)} e(\Delta') \le C_0 e(\Delta) = C_0 \tilde{e}(\Delta) p(\Delta). \tag{5.10}$$

Therefore,

$$\Sigma_1 = \sum_{\Delta\in\Lambda}\sum_{\Delta'\in\mathcal{L}(T_\Delta)} e(\Delta') \le C_0 \sum_{\Delta\in\Lambda} \tilde{e}(\Delta)p(\Delta). \tag{5.11}$$

Since $\Sigma_0 = \sum_{\Delta\in\mathcal{L}_0} \tilde{e}(\Delta)p(\Delta)$, we have

$$\Sigma_0 + \Sigma_1 \le C_0 \sum_{\Delta\in\mathcal{L}(\Upsilon_0)} \tilde{e}(\Delta)p(\Delta). \tag{5.12}$$

We next want to note that

$$\tilde{e}(\Delta) \le t, \quad \Delta \in \mathcal{L}(\Upsilon_0). \tag{5.13}$$

This is the case for any $\Delta \in \Lambda$ by the very definition of t (note that any $\Delta \in \Lambda$ is a leaf of Υ_0). To see (5.13) for any other leaf of Υ_0 (i.e. those not in Λ) consider the state of affairs when all of the $\Delta \in \Lambda$ have already been generated by the algorithm and the $\Delta \in \Lambda$ are to be subdivided. Let $T' \subset T$ be the tree representing the state of the algorithm at this point. We claim that any $\Delta \in \mathcal{L}(\Upsilon_0)$ that is not in Λ is already a leaf of T' and satisfies $\tilde{e}(\Delta) < t$. In fact, if $\tilde{e}(\Delta) > t$, then we would be subdividing Δ and not the elements in Λ. So $\tilde{e}(\Delta) < t$ (it cannot be equal to t since otherwise Δ would be in Λ) and by the definition of t, Δ cannot be an interior node to T and therefore must be a leaf of T. Thus, we have verified (5.13). Using this in (5.12) we obtain

$$\begin{aligned} E(T) = \Sigma_0 + \Sigma_1 &\le C_0 \sum_{\Delta\in\mathcal{L}(\Upsilon_0)} \tilde{e}(\Delta)p(\Delta) \le C_0 t(N_0 + N(\Upsilon_0)) \\ &\le C_0 t(N_0 + N(T)), \end{aligned} \tag{5.14}$$

where we have used (5.6) in the next to last inequality.

We shall now prove that $t(N_0 + N(T)) \le CE_m$. Let $\mathcal{L}^*$ be the collection of all leaves $\Delta^* \in \mathcal{L}(T^*)$ which are in the interior of T and for each of these Δ^* let T_{Δ^*} be the tree consisting of all $\Delta \in T \setminus \mathcal{L}(T)$ such that Δ is a descendant of Δ^*. For each leaf $\Delta \in \mathcal{L}(T_{\Delta^*})$, we have Δ is in the interior of T and therefore $\tilde{e}(\Delta) \ge t$ by the very definition of t. So we can apply (5.1) to find

$$\begin{aligned} t \sum_{\Delta\in\mathcal{L}(T_{\Delta^*})} p(\Delta) &\le \sum_{\Delta\in\mathcal{L}(T_{\Delta^*})} p(\Delta)\tilde{e}(\Delta) \\ &= \sum_{\Delta\in\mathcal{L}(T_{\Delta^*})} e(\Delta) \le C_0 e(\Delta^*). \end{aligned} \tag{5.15}$$

We now sum (5.15) over all $\Delta^* \in \mathcal{L}^*$ and use (5.7) to obtain

$$tM \le t \sum_{\Delta^*\in\mathcal{L}^*}\sum_{\Delta\in\mathcal{L}(T_{\Delta^*})} p(\Delta) \le C_0 \sum_{\Delta^*\in\mathcal{L}^*} e(\Delta^*) \le C_0 E_m, \tag{5.16}$$

where

$$M := \sum_{\Delta^* \in \mathcal{L}^*} N(T_{\Delta^*}). \tag{5.17}$$

To conclude the proof, we shall show that

$$N_0 + N(T) \le 2(K+2)M\ . \tag{5.18}$$

First, let T' be the tree obtained from T by deleting all of the leaves of T. Then, T' contains T_{Δ^*} for all $\Delta^* \in \mathcal{L}^*$.

In the trivial case $N(T') < N_0$ we use (3.4) to receive

$$n = N(T) = \#(T') \le N_0 + K\ N(T') \le (K+1)N_0 \tag{5.19}$$

and therefore $m \le n/(2K+2) \le N_0/2$. Hence there are at least $N_0 - m \ge N_0/2$ root nodes of T_* that are not subdivided in T^* and according to our assumption they are subdivided in T. Thus $M \ge N_0/2$ and returning to (5.19), we obtain (5.18).

In the case $N_0 \le N(T')$ from (3.4) we know that

$$\begin{aligned} N_0 + N(T) &= N_0 + \#(T') \\ &\le N_0 + N_0 + K\ N(T') \le (K+2)N(T'). \end{aligned} \tag{5.20}$$

Finally, we claim that

$$N(T') \le 2M, \tag{5.21}$$

which will complete the proof of (5.18) in this case. To see (5.21), we note that the only interior nodes of T' that are not interior nodes in one of the T_{Δ^*} are those which are interior nodes in T^*. This means that there are at most $N(T^*) \le m$ of them. Hence,

$$\begin{aligned} N(T') \le M + m &\le M + \frac{N(T)}{2K+2} \\ &\le M + \frac{N_0 + K\ N(T')}{2K+2} \le M + N(T')/2, \end{aligned} \tag{5.22}$$

where we have used the property $N(T) = n \ge (2K+2)m$ of the theorem, and (3.4). □

Remark 5.3 In the **Second Algorithm** the use of the largest $\tilde{e}(\Delta)$ may require a sorting procedure with complexity $O(N \log N)$. To keep the number of operations of order $O(N)$ we can use *binary bins* instead of sorting. Then after $\tilde{e}(\Delta)$ is calculated, we determine an integer κ such that $2^\kappa \le \tilde{e}(\Delta) < 2^{\kappa+1}$, and place Δ into a bin with index κ. We now choose for subdivision the leaves from the bin with the largest possible index and proceed as in the **Second Algorithm**. Theorem 5.2 remains valid for this variant of the algorithm. The proof is the same with the following modifications. We define $t = 2^\kappa$ where κ is the smallest index of a bin which contains an interior node of T. The set Λ is now the collection of interior nodes of T which belong to that bin. Then $t \le \tilde{e}(\Delta) < 2t$ for $\Delta \in \Lambda$, and subsequently $\tilde{e}(\Delta') < 2t$ for $\Delta' \in T'$. The only other changes are to replace t with $2t$ in the upper estimates (5.13) and (5.14).

Most numerical implementations of adaptive algorithms such as those that we have given would have the goal of producing a tree T such that $E(T) \le \mu$ where $\mu > 0$ is some prescribed tolerance. In the case of **Second Algorithm** this can be accomplished simply by introducing an error check after each step of the algorithm.

Thresholding Second Algorithm *Given an error tolerance $\mu > 0$, this algorithm produces a tree T_μ such that $E(T_\mu) \le \mu$ as follows.*

(i) Compute $e(\Delta) = \tilde{e}(\Delta)$, $\Delta \in T_0$, and then $E(T_0)$ for T_0 the set of root nodes of T_*. Define $T = T_0$ and proceed to step (ii).

(ii) If $E(T) \le \mu$, then define $T_\mu := T$ and stop the algorithm. If $E(T) > \mu$, then proceed to step (iii).

(iii) Given a proper subtree T of T_*, compute $\rho := \max_{\Delta' \in \mathcal{L}(T)} \tilde{e}(\Delta')$ and subdivide all $\Delta \in \mathcal{L}(T)$ for which $\tilde{e}(\Delta) = \rho$ thereby obtaining the new tree T'. Redefine T to be T' and proceed to step (ii).

Corollary 5.4 *There are absolute constants $C_1, c_1 > 0$ such that for any error tolerance $\mu > 0$, the tree T_μ produced by the* **Thresholding Second Algorithm** *has the properties:*

(i) If $\tilde{T} \subset T_*$ is any tree satisfying $E(\tilde{T}) \le c_1\mu$ then

$$N(T_\mu) \le C_1 N(\tilde{T}). \tag{5.23}$$

(ii) The number of evaluations of e and the number of arithmetic operations in producing T_μ is $\le C_1(N_0 + N(T_\mu))$.

Proof. Statement (ii) is clear. To prove (i), we take $c_1 = \lambda(C+1)^{-1}$ where C is the constant in Theorem 5.2 and $\lambda \in (0, 1/4)$ is a constant which depends only on K and will be prescribed in the course of the proof. Let T' be the tree before the last subdivisions are made to produce $T := T_\mu$. Then, $E(T') > \mu$. We define T^* as the best tree approximation with $m := N(T^*) = N(\tilde{T})$ subdivisions. We shall consider three cases:

Case 1 $E(T) \ge \lambda\mu$. We use Theorem 5.2 for T. Since

$$E_m = E(T^*) \le E(\tilde{T}) \le c_1\mu \le (C+1)^{-1}E(T),$$

this theorem says that $m > N(T)/(2K+2)$ which proves (i) in this case.

Case 2 $N(T') \geq \lambda N(T)$. In this case, we start with the inequality

$$E(T^*) \leq E(\tilde{T}) \leq c_1\mu \leq c_1 E(T') \leq (C+1)^{-1}E(T').$$

We now use Theorem 5.2 for T' and deduce that $m > N(T')/(2K+2) \geq \lambda N(T)/(2K+2)$ which proves (i) in this case.

Case 3 $E(T) < \lambda\mu$ and $N(T') < \lambda N(T)$. This case is the most complicated. Let A be the set of nodes in T_0 which were subdivided in producing T from T' and let B be the set of all other nodes in T' which were subdivided in producing T. We further set $a := \#(A)$, $b := \#(B)$. For each $\Delta \in A \cup B$ we have $\tilde{e}(\Delta) = \rho := \max_{\Delta\in\mathcal{L}(T')} \tilde{e}(\Delta)$. We also have $N(T') \geq K^{-1}b$ and $N(T') < \lambda N(T) = \lambda(b + a + N(T'))$ which implies that

$$b \leq 2\lambda K a \tag{5.24}$$

provided $\frac{(1-\lambda)}{K} - \lambda \geq \frac{1}{2K}$, which is true whenever λ is sufficiently small (depending only on K).

We next want to show that $\Sigma_B := \sum_{\Delta\in B} e(\Delta)$ is small compared with $\Sigma_A := \sum_{\Delta\in A} e(\Delta)$. For this, we introduce T_B which is the subtree of T' consisting of B and all of its ancestors and N_B which is the number of root nodes of T_B. We use (5.7) to find

$$\Sigma_B = \sum_{\Delta\in B} p(\Delta)\tilde{e}(\Delta) = \rho(N_B + N(T_B)) \leq 2N(T_B)\rho \leq 4b\rho, \tag{5.25}$$

where the last inequality used (3.1). On the other hand,

$$\Sigma_A = \sum_{\Delta\in A} e(\Delta) = \sum_{\Delta\in A} \tilde{e}(\Delta) = a\rho, \tag{5.26}$$

because $\tilde{e}(\Delta) = e(\Delta)$ for root nodes. From (5.24), we obtain

$$\Sigma_B \leq 4b\rho \leq 8\lambda K a\rho \leq 8\lambda K \Sigma_A. \tag{5.27}$$

Since

$$\Sigma_A + \Sigma_B \geq E(T') - E(T) \geq (1-\lambda)\mu, \tag{5.28}$$

(5.27) implies that

$$\Sigma_A \geq \mu/2 \tag{5.29}$$

provided we choose λ small enough (depending only on K). We now fix $\lambda = 1/(8K+2)$ to satisfy the above restrictions. Since $E(T^*) \leq c_1\mu \leq \lambda\mu \leq \mu/4$, at least one half of the nodes in A cannot be in $\mathcal{L}(T^*)$ and therefore were subdivided in producing T^*. That is $a/2 \leq N(T^*) = N(\tilde{T})$. On the other hand,

$$\begin{aligned}(1-\lambda)N(T) &\leq N(T) - N(T') = a + b\\ &\leq a(1+2\lambda K) \leq 2(1+2\lambda K)N(\tilde{T}).\end{aligned} \tag{5.30}$$

This proves (i) in this final case and completes the proof of the corollary. □

6 Piecewise polynomial approximation

The remainder of this paper will be devoted to giving examples of settings where the above algorithms can be employed. In this section, we give our first example, which is quite simple. We suppose that P_0 is an initial triangulation of a polygonal domain $\Omega \subset I\!R^d$ and we have in hand a specific subdivision rule that tells us how to subdivide any given triangle that arises during a subdivision process (i.e. we have the master tree T_* for P_0 and this subdivision rule). We consider the problem of approximating a given function $f \in L_p(\Omega)$, $1 \le p < \infty$, by piecewise polynomials of degree $r \ge 0$ on partitions P which correspond to subtrees of T_*. Let Π_r denote the space of polynomial of degree r. We shall assume there is no continuity restrictions on these piecewise polynomials across the boundaries of the simplicial cells which make up P.

We fix $1 \le p < \infty$ and $f \in L_p(\Omega)$ and define

$$e(\Delta) := \inf_{P \in \Pi_r} \|f - P\|^p_{L_p(\Delta)}, \qquad \Delta \in T_*. \tag{6.1}$$

Corresponding to this e, for any tree $T \subset T_*$, we define

$$E(T) := \sum_{\Delta \in \mathcal{L}(T)} e(\Delta). \tag{6.2}$$

We recall that the trees $T \subset T_*$ are in one to one correspondence with adaptively generated partitions P. Namely, the leaves $\mathcal{L}(T)$ form a partition $P = P(T)$ which is obtained by applying $N(T)$ subdivisions. Let us denote by

$$E(P) := E(T(P)) \tag{6.3}$$

which is the p-th power of the error in approximating f in the $L_p(\Omega)$ norm by piecewise polynomials of degree r on the partition P. Let us further denote by $\mathcal{P}_n$ the set of all partitions P which can be generated by at most n subdivisions of P_0.

Because of the disjoint supports of the triangular cells Δ in $\mathcal{L}(T)$ and the set subadditivity of $\|\cdot\|^p_{L_p}$, we have that condition (4.1) holds. This means we can employ the first algorithm of §4. Thus, we can apply either the first or second algorithm in this setting. Either of Theorem 4.4 or Theorem 5.2 gives

Theorem 6.1 *If $f \in L_p(\Omega)$, then the* **First Algorithm** *of §4 and the* **Second Algorithm** *of §5 when applied with e of* (6.1) *both generate a tree T whose leaves $\mathcal{L}(T)$ form a partition $P = P(T)$ such that*

$$E(P) \le C_1 \inf_{P' \in \mathcal{P}_m} E(P'), \tag{6.4}$$

whenever $m \le c_1 N(T)$, where $C_1, c_1 > 0$ are absolute constants.

7 Piecewise linear approximation with continuity across boundary edges

In this section, we want to show how to apply the principles of the second algorithm in a typical Finite Element Application. We suppose that P_0 is an initial triangulation of a polygonal domain $\Omega \subset I\!R^2$ and we have in hand a specific subdivision rule that tells us how to subdivide any given triangle that arises during a subdivision process (i.e. we have the master tree T_* for P_0 and this subdivision rule). There are two new ingredients that we wish to incorporate into our analysis. The first is the appearance of *hanging nodes*. Given a partition P that is a refinement of P_0, we let $\mathcal{V}_P$ denote the collection of all vertices of the triangles which make up P. Such a vertex v is called a hanging node for $\Delta \in P$ if it appears in the interior of one of the edges of Δ. We shall say that a partition P is admissible if it has no hanging nodes. A completion $\bar{P}$ of a partition P is an admissible partition which is a refinement of P (using the specified subdivision rule).

The second ingredient which will cause us some difficulty is that the typical error functionals e which arise in Finite Element Applications will typically not satisfy our condition (5.1). The reason for this is that the Galerkin method requires that the approximants have some smoothness which usually implies at least continuity of the piecewise polynomials. Therefore the piecing together of local approximants to obtain a good global approximation prevents $e(\Delta)$ from being completely localized to Δ and as a result (5.1) will typically fail to hold.

We shall show how to circumvent these difficulties in this section. We shall limit our discussion to one specific subdivision rule (newest vertex bisection) which is discussed and utilized in the paper [1]. In principle, the results of this section could be extended to other subdivision rules but this would require the development of specific properties of the completion process for these rules. These properties were developed in [1] for newest vertex bisection and required a nontrivial analysis to do so.

We first introduce and discuss the properties of newest vertex bisection that we shall need. The proofs of these properties not given here will all be found in [1]. Given an initial admissible partition P_0 of Ω, we give each edge of this partition a label of 0 or 1 in such a way that for each triangular cell exactly one of its edges has the label 0. That such a labelling is possible is proved in Lemma 2.1 of [1]. The edge in Δ given the value 0 will be denoted by $E(\Delta)$. The vertex opposite this edge is called the *newest vertex* for Δ and is denoted by $v(\Delta)$. The rule for newest vertex subdivision of a cell Δ is to insert a new edge connecting $v(\Delta)$ to the midpoint of $E(\Delta)$ thus producing two new cells Δ_1, Δ_2 (the children of Δ). These two new cells are given the new vertex which is the midpoint of $E(\Delta)$. The three new sides produced (i.e. the new diagonal and the two new sides arising from the bisection of $E(\Delta)$)

are given the label 2. Thus at this stage each cell will have the label $(1, 1, 0)$ (if it has not been subdivided) or $(2, 2, 1)$ (if it is a new cell obtained by subdivision) and the newest vertex is always opposite the side with smallest label. In general, any cell appearing in newest vertex bisection has the label $(k + 1, k + 1, k)$ with newest vertex the one opposite the smallest labelled side. When this cell is subdivided it produces two children (using the newest vertex bisection rule) which have the labels $(k + 2, k + 2, k + 1)$ and we have the same property that the side opposite the smallest label is the newest vertex.

In going further in this section, T_* will denote the master tree for newest vertex subdivision as described above. So T_* is a binary tree. We denote as before $\mathcal{T}_n$ the class of proper trees with $N(T) = n$ and by $\mathcal{T}_n^a$ the class of admissible trees. Thus $T \in \mathcal{T}_n^a$ means that $T \in \mathcal{T}_n$ and that its leaves form an admissible partition P (i.e. P has no hanging nodes). We have already alluded to the following fact:

Remark 7.1 Any tree $T \in \mathcal{T}_n$ can be refined to find a tree $\bar{T}$ which is admissible and satisfies

$$N(\bar{T}) \leq C_2 N(T) \tag{7.1}$$

with C_2 an absolute constant.

Proof. In essence, this is Theorem 2.4 of [1]. However that theorem is not stated in the form of (7.1) so we must say a few words on how to obtain (7.1). Let $\mathcal{M}_0$ denote the set of all root nodes in T_* which are interior nodes of T (i.e. they are subdivided in the creation of T). We let T_1' denote the tree obtained by subdividing exactly the cells in $\mathcal{M}_0$. We then denote by T_1 the completion of T_1'. Next, we let $\mathcal{M}_1$ denote the set of all leaves of T_1 which are interior nodes of T. We let T_2' denote the tree obtained from T_1 by subdividing precisely the nodes in $\mathcal{M}_1$ and let T_2 denote the completion of T_2'. We continue in this way until the smallest value of k where $\mathcal{M}_k = \emptyset$. We have thus reached a tree $\bar{T} := T_k$ which is admissible and is a refinement of T. Theorem 2.4 of [1] proves that

$$\begin{aligned} \#(P_0) + N(\bar{T}) &\leq \#(P_0) + C(\#(\mathcal{M}_0) + \cdots + \#(\mathcal{M}_{k-1})) \\ &\leq \#(P_0) + C\, N(T) \end{aligned} \tag{7.2}$$

with C an absolute constant. Here, in the last inequality we used the fact that each node in $\cup_{j=0}^{k-1}\mathcal{M}_j$ is an interior node of T by the very definition of the set $\mathcal{M}_j$. □

We want next to define the types of error functional e we shall associate to newest vertex bisection. To do this, we introduce the *minimal ring* associated to a triangular cell $\Delta \in T_*$. Given any admissible partition P and $\Delta \in P$, we define

$$R(\Delta, P) := \cup_{\Delta' \in P, \Delta \cap \Delta' \neq \emptyset} \Delta' \tag{7.3}$$

which is the first ring about Δ. This ring depends on P. However, we can find a minimal ring about Δ which does not depend on P. Namely, we define

$$R_-(\Delta) := \cap_P R(\Delta, P) = \cup_{\Delta' \in P_-(\Delta)} \Delta' \tag{7.4}$$

where the intersection is taken over all **admissible** partitions P. Here the set $P_-(\Delta)$ is the collection of cells from T_* which touch Δ and make up $R_-(\Delta)$. The structure of the set $P_-(\Delta)$ has been given in Lemma 4.3 of [1].

We can now define functionals e on T_* for which we can develop an analogue to the results of §5. These functionals start with a function norm $\|\cdot\|$ which has a power which is set subadditive. Examples are any of the L_p, $1 \le p < \infty$, norms, whose p-th power has this property, or any of the Sobolev $H^s = W^s(L_2(\Omega))$ norms, $s \ge 0$, whose square has this property. We shall limit our discussion to the latter case since it is of most interest in Finite Element Applications. So, fix $0 \le s \le 1$, and let $\|\cdot\|$ denote the $H^s(\Omega)$ norm with Ω the polygonal domain corresponding to the partition P_0. We fix a function $f \in H^s(\Omega)$ and for each $\Delta \in T_*$, we define

$$e(\Delta) := \inf_S \|f - S\|^2_{H^s(R_-(\Delta))} \tag{7.5}$$

where the infimum is taken over all continuous piecewise linear functions S defined on $R_-(\Delta)$ which are subordinate to $P_-(\Delta)$. We define $E(T)$ for this e as we have before. We also define the best error E_m but now the competition is restricted to admissible partitions:

$$E_m := \inf_{T \in \mathcal{T}^a_m} E(T). \tag{7.6}$$

We shall now modify the second algorithm so as to generate admissible partitions. Some modifications are also made to make the proof of the following theorem proceed without difficulty.

While we view our goal as to create partitions based on newest vertex, we shall actually use the following subdivision rule:

New Subdivision Rule *If Δ is a triangular cell, then it is subdivided into* 32 *children which are obtained by applying newest vertex bisection* 5 *times uniformly on Δ. In other words, the children of this subdivision rule are all fifth generation offspring of Δ obtained by the newest vertex bisection rule.*

Our reason for applying so many newest vertex bisections in just one primary subdivision of the **New Subdivision Rule** is simply that it guarantees the following property:

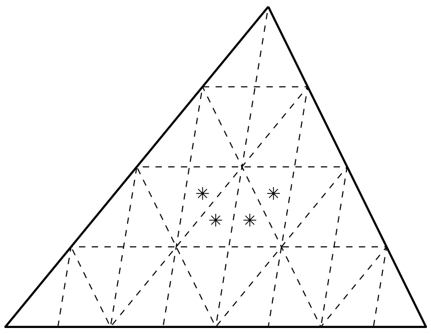

Fig. 7.1. Application of the New Subdivision Rule to a triangle

Property I *When applying the* **New Subdivision Rule** *to a triangular cell* Δ*, among the* 32 *children of* Δ *there is one child* Δ' *whose minimal ring* $R_-(\Delta')$ *of* (7.5) *(still defined according to newest vertex bisection) is completely contained in* Δ*.*

Indeed, take for Δ' any child which does not touch the boundary of Δ (see marked triangles in Figure 7.1).

To define the modified second algorithm, we use the definition of $\tilde{e}$ for the **New Subdivision Rule**.

Modified Second Algorithm *Given as input a target accuracy* μ*, this algorithm produces an admissible tree* T *satisfying*

$$E(T) \le \mu \tag{7.7}$$

as follows. For $j = 0$*,* T_0 *is the collection of roots of* T_**. For each* j*, we do the following:*

(i) Compute $E(T_j)$*. If* $E(T_j) \le \mu/C_2$*, with* C_2 *the constant of* (7.1) *go to (iii). If* $E(T_j) > \mu/C_2$ *go to (ii).*

(ii) We examine all leaves of T_j *and subdivide, according to the* **New Subdivision Rule** *all of the leaves* Δ *with largest* $\tilde{e}(\Delta)$ *to produce* T_{j+1}*. We replace* j *by* $j+1$ *and return to (i).*

(iii) We define T *as the completion of* T_j*. This completion is made according to the newest vertex bisection completion and therefore satisfies* $N(T) \le C_2 N(T_j)$*.*

The following theorem describes the optimal properties of this algorithm.

Theorem 7.2 *There are absolute constants* $C_3, c_3 > 0$ *such that the output* T *of the* **Modified Second Algorithm** *satisfies*

$$E(T) \le C_3 E_m \tag{7.8}$$

whenever $m \le c_3 n$ *and* $n := N(T)$*. Here,* E_m *is defined in the competition among all admissible subtrees of the newest vertex tree* T_**. To create* T*, the* *algorithm uses $\leq C_4(n + N_0)$ arithmetic operations and computations of e, where N_0 is the number of root nodes of T_*.*

It is clear at the outset that the tree T of the **Modified Second Algorithm** is a subtree of T_* and it is admissible. The remainder of this section will be devoted to proving this theorem which is a modification of the proof of Theorem 5.2. We let T^* be an admissible tree with $N(T^*) = m$ which attains E_m. As in the proof of Theorem 5.2, we assume (without loss of generality) that all root nodes are interior nodes (i.e. have been subdivided) in either T or T^*, and resolve the trivial case $N(T') < N_0$.

We shall prove two inequalities:

(a) $E(T) \leq Ct(N_0 + N(T))$;
(b) $t(N_0 + N(T)) \leq CE_m$ if $m \leq cn$,

where the constants are absolute and $t > 0$ is the value specified as in the proof of Theorem 5.2. These two inequalities combine to give the theorem.

Proof of (a). If we had the original subadditivity property (5.1) for e, the proof would be trivial. Indeed, we would know (a) with $N(T_j)$ in place of $N(T)$ as it is in (5.14). Then, we can replace $N(T_j)$ by $N(T)$ because of (7.1). However, this subadditivity does not necessarily hold. Instead, we have the following weaker condition:

Modified subadditivity *For any $\Delta \in T_*$ and any* **admissible** *finite tree T_Δ with single root node Δ, we have*

$$\sum_{\Delta' \in \mathcal{L}(T_\Delta)} e(\Delta') \leq C_0 e(\Delta) \tag{7.9}$$

with C_0 an absolute constant.

This property follows from the set subadditivity of $\|\cdot\|^2$ and the fact that any point $x \in \Omega$ can appear in at most k_0 of the minimal rings $R_-(\Delta')$, $\Delta' \in \mathcal{L}(T_\Delta)$ where k_0 is an absolute constant (see the simple proof in [1] which uses the fact that T_Δ is admissible).

Let us now see how to use this modified subadditivity to prove (a). Let $\tilde{T} = T_j$ be the final partition obtained by applying the **Modified Second Algorithm** and let T be the completion of $\tilde{T}$ which is obtained by additional subdivisions. We know from (7.1) that

$$N(T) \leq C_2 N(\tilde{T}). \tag{7.10}$$

As in Theorem 5.2, we let t be the smallest value attained by $\tilde{e}$ on the interior nodes of $\tilde{T}$ and let $\Lambda := \{\Delta : \tilde{e}(\Delta) = t\}$ be the collection of interior nodes of $\tilde{T}$ where $\tilde{e}$ takes on the value t. We define Υ_0, Υ_1 and T_Δ, $\Delta \in \Lambda$, $\mathcal{L}_0$, $\mathcal{L}_1$, Σ_0, and Σ_1 as in the proof of Theorem 5.2. These trees and sets of leaves are defined relative to T (and not $\tilde{T}$). Thus each of the trees T_Δ is admissible. Then as before,

$$E(T) = \sum_{\Delta \in \mathcal{L}_0} e(\Delta) + \sum_{\Delta \in \mathcal{L}_1} e(\Delta) = \Sigma_0 + \Sigma_1. \tag{7.11}$$

Since T_Δ are admissible trees, it follows that for each $\Delta \in \Lambda$, we have

$$\sum_{\Delta' \in \mathcal{L}(T_\Delta)} e(\Delta') \le C_0 e(\Delta) = C_0 \tilde{e}(\Delta) p(\Delta), \tag{7.12}$$

where now we have used the modified subadditivity (7.9). We can continue then with the reasoning of Theorem 5.2 and arrive at

$$\Sigma_0 + \Sigma_1 \le C_0 \sum_{\Delta \in \mathcal{L}(\Upsilon_0)} \tilde{e}(\Delta) p(\Delta). \tag{7.13}$$

In contrast to Theorem 5.2, we do not know that $\tilde{e}(\Delta') \le t$, $\Delta' \in \mathcal{L}(\Upsilon_0)$, because of the completion process. However, we do know this inequality for all $\tilde{\Delta} \in \tilde{\Lambda}$ where $\tilde{\Lambda}$ consists of all $\tilde{\Delta}$ that are in $\mathcal{L}(\tilde{T})$ but are not in T_Δ for any $\Delta \in \Lambda$. Indeed, this is the same as in the proof of Theorem 5.2. Now, for any $\tilde{\Delta} \in \tilde{\Lambda}$, we let $T_{\tilde{\Delta}}$ be the tree with single root $\tilde{\Delta}$ which consists of all cells in T contained in $\tilde{\Delta}$. Then, $T_{\tilde{\Delta}}$ is admissible and the same derivation as (7.12) gives

$$\sum_{\Delta' \in \mathcal{L}(T_{\tilde{\Delta}})} e(\Delta') \le C_0 \tilde{e}(\tilde{\Delta}) p(\tilde{\Delta}). \tag{7.14}$$

Since every $\Delta' \in \mathcal{L}(\Upsilon_0)$ appears either in Λ or in one of the $\mathcal{L}(T_{\tilde{\Delta}})$, $\tilde{\Delta} \in \tilde{\Lambda}$, we obtain using (5.6)

$$\begin{aligned} \sum_{\Delta' \in \mathcal{L}(\Upsilon_0)} e(\Delta') &\le C_0 \sum_{\tilde{\Delta} \in \tilde{\Lambda}} \tilde{e}(\tilde{\Delta}) p(\tilde{\Delta}) + \sum_{\Delta \in \Lambda} \tilde{e}(\Delta) p(\Delta) \\ &\le C_0 t (N_0 + N(\tilde{T})). \end{aligned} \tag{7.15}$$

This gives (a) because of (7.11), (7.13) and the fact that $N(\tilde{T}) \le N(T)$. □

Proof of (b). We shall follow the same line of proof as in Theorem 5.2 until we run into a difficulty and then we shall show how to overcome that difficulty. The proof uses $\tilde{T}$ rather than T in the constructions. We let $\mathcal{L}^*$ again denote the set of leaves of $\mathcal{L}(T^*)$ which are interior nodes of $\tilde{T}$. For each $\Delta^* \in \Lambda^*$ we define T_{Δ^*} as the set of all $\Delta \in \tilde{T} \setminus \mathcal{L}(\tilde{T})$ which are contained in Δ^*. In Theorem 5.2 we showed that

$$tM \le C_0 \sum_{\Delta^* \in \mathcal{L}^*} e(\Delta^*) \le C_0 E_m \tag{7.16}$$

where

$$M := \sum_{\Delta^* \in \mathcal{L}^*} N(T_{\Delta^*}). \tag{7.17}$$

We shall show this again with a new constant on the right side of (7.16) but we need a different proof because we do not have the subadditivity used in (5.15).

Lemma 7.3 *For each $\Delta^* \in \mathcal{L}^*$, we have*

$$tN(T_{\Delta^*}) \leq C_5 e(\Delta^*) \tag{7.18}$$

with C_5 an absolute constant.

Proof. In the case that $\{\Delta^*\} = \mathcal{L}(T_{\Delta^*})$, we have $N(T_{\Delta^*}) = 0$ and (7.18) holds trivially. So in going further we assume $N(T_{\Delta^*}) \geq 1$ which means that the 32 children of Δ^* from the **New Subdivision Rule** are in the interior of $\tilde{T}$. Let $\Delta \in \mathcal{L}(T_{\Delta^*})$ and let $\bar{\Delta}$ be its parent relative to the **New Subdivision Rule** used to create $\tilde{T}$. Recall that $\tilde{e}$ is constant on each of the 32 children of $\bar{\Delta}$. From Property I, we can choose one of these children Δ' which is strictly contained in the interior of $\bar{\Delta}$ and for which $R_-(\Delta') \subset \bar{\Delta}$. Let us denote by Λ' the set of all such Δ' that we have created. We note that for each pair Δ_1, Δ_2 in Λ' the sets $R_-(\Delta_1) \cap R_-(\Delta_2)$ has measure 0. Indeed the two parents $\bar{\Delta}_1, \bar{\Delta}_2$ of these cells are disjoint (for this property one is using the fact that when $\bar{\Delta}_1$ is subdivided so are all of its siblings). Thus, from the set subadditivity of $\|\cdot\|^2$ we obtain

$$t\#(\Lambda') \leq \sum_{\Delta' \in \Lambda'} e(\Delta') \leq e(\Delta^*) \tag{7.19}$$

where we have used the fact that $\tilde{e}(\Delta') \geq t$, $\Delta' \in \Lambda'$, because these nodes are interior to $\tilde{T}$. Now since $\#(\Lambda') = \#(\mathcal{L}(T_{\Delta^*})/32$ and $N(T_{\Delta^*}) \leq 2\#(\mathcal{L}(T_{\Delta^*}))$ (see (3.1) and (3.2)), (7.18) follows from (7.19) with $C_5 = 64$. □

Having established (7.18), we can sum over all Δ^* in $\mathcal{L}^*$ and derive (7.16) with a new constant in place of C_0.

We can conclude the proof of *Proof of (b)* as in Theorem 5.2. Namely, let T' be the tree obtained from $\tilde{T}$ by deleting all of the leaves of $\tilde{T}$. Then, T' contains T_{Δ^*} for all $\Delta^* \in \mathcal{L}^*$. As in (5.20), we obtain (here we have $K = 32$)

$$N(\tilde{T}) \leq (K+1)N(T') \quad \text{and} \quad N_0 + N(\tilde{T}) \leq (K+2)N(T'). \tag{7.20}$$

Finally, we claim that

$$N(T') \leq CM \tag{7.21}$$

with C an absolute constant, which will complete the proof. To see (7.21), we note that the only interior nodes of T' that are not interior nodes in one of the T_{Δ^*} are those which are interior nodes of T^*. This means that there are at most $N(T^*) \leq m$ of them. Hence,

$$N(T') \leq M + m \leq M + N(\tilde{T})/(2K+2) \leq M + N(T')/2, \tag{7.22}$$

where we have used (7.20) and the inequality

$$N(\tilde{T}) \geq C_2^{-1} N(T) = C_2^{-1} n \geq 2(K+1)m \tag{7.23}$$

which holds provided c_3 is chosen smaller than $C_2^{-1}/(2K+2)$. This completes the *Proof of (b)* and completes the proof of Theorem 7.2. □

Finally, we remark that the properties of the **Modified Second Algorithm** can also be described in the form of Corollary 5.4.

Corollary 7.4 *There are absolute constants $C_1, c_1 > 0$ such that for any error tolerance $\mu > 0$, the tree T_μ produced by the* **Modified Second Algorithm** *has the properties:*

(i) If $\tilde{T} \subset T_*$ is any tree satisfying $E(\tilde{T}) \leq c_1\mu$ then

$$N(T_\mu) \leq C_1 N(\tilde{T}). \tag{7.24}$$

(ii) The number of evaluations of e and the number of arithmetic operations in producing T_μ is $\leq C_1(N_0 + N(T_\mu))$.

Proof. The proof of this corollary is the same as that of Corollary 5.4. □

References

[1] Binev, P., Dahmen, W., DeVore, R.: Adaptive finite element methods with convergence rates. IGPM Report # 218, RWTH Aachen, June 2002

[2] Cohen, A., Dahmen, W., Daubechies, I., DeVore, R.: Tree approximation and optimal encoding. Appl. Comp. Harm. Anal. **11**, 192–226 (2001)