SIAM J. NUMER. ANAL.
Vol. 38, No. 3, pp. 1013–1033

MODIFIED ADAPTIVE ALGORITHMS*

YINGKANG HU[†], KIRILL A. KOPOTUN[‡], AND XIANG MING YU[§]

Abstract. It is well known that the adaptive algorithm is simple and easy to program but the results are not fully competitive with other nonlinear methods such as free knot spline approximation. We modify the algorithm to take full advantages of nonlinear approximation. The new algorithms have the same approximation order as other nonlinear methods, which is proved by characterizing their approximation spaces. One of our algorithms is implemented on the computer, with numerical results illustrated by figures and tables.

Key words. nonlinear approximation, approximation spaces, adaptive algorithms, data reduction, piecewise polynomials, splines, Besov spaces, degree of approximation, modulus of smoothness

AMS subject classifications. 41-04, 41A10, 41A15, 41A17, 41A25, 41A27

PII. S0036142999353569

1. Introduction.

It is common knowledge that nonlinear approximation methods are better, in general, than their linear counterparts. In the case of splines, nonlinear approximation puts more knots where the function to be approximated changes rapidly, which results in dramatic improvements in approximating functions with singularities. There are various satisfactory results on *free knot spline approximation*, in which knots are chosen at one's will. Most related theorems are proved by showing the existence of certain *balanced partitions* (a more accurate description will be given later). This may cause difficulties in practice, since it is often numerically expensive to find such balanced partitions. Then, there is so-called *adaptive approximation* by *piecewise polynomial* (PP) functions, in which only dyadic intervals are used in the partition. Adaptive approximation draws great attention because of its simplicity in nature. As a price to pay for the simplicity, its approximation power is slightly lower than that of its free knot counterpart. Moreover, it is not known exactly what kind of functions can be approximated to a prescribed order; that is, there is no characterization of adaptive approximation spaces. We point out here that when we say adaptive algorithms in this paper, we mean those that approximate a given (univariate) function by PP functions/splines. There are other kinds of adaptive algorithms; some are characterized in the literature (see [10] for an example).

In this paper, we shall modify the existing adaptive algorithms in two ways. The resulting algorithms have the same approximation power as free knot spline approximation while largely keeping the simplicity of adaptive approximation. In the next section, we shall state some known results on free knot spline approximation. After describing our algorithms in section 3, in section 4 we shall give our main results, which are parallel to those on free knot spline approximation given in the next section. Numerical implementation and examples will be the contents of the last section.

*Received by the editors March 17, 1999; accepted for publication (in revised form) February 9, 2000; published electronically August 29, 2000.
http://www.siam.org/journals/sinum/38-3/35356.html

[†]Department of Mathematics and Computer Science, Georgia Southern University, Statesboro, GA 30460 (yhu@gasou.edu).

[‡]Department of Mathematics, University of Manitoba, Winnipeg, MB, Canada R3T 2N2 (kkopotun@math.vanderbilt.edu). This author was supported by NSF grant DMS 9705638.

[§]Department of Mathematics, Southwest Missouri State University, Springfield, MO 65804 (xmy944f@mail.smsu.edu).

We emphasize that we consider only the univariate case in this paper. The idea of merging cubes was initially introduced and used by Cohen et al. in their recent paper [11] on multivariate adaptive approximation. The resulting partition consists of rings, which are cubes with a (possibly empty) subcube removed. Their algorithm produces near minimizers in extremal problems related to the space $BV(R^2)$. The authors further explored this algorithm in [21]. In particular, we were able to obtain results on extremal problems related to the spaces $\mathbf{V}_{\gamma,p}(R^d)$ of functions of "bounded variation" and Besov spaces $\mathbf{B}^\alpha(R^d)$. This algorithm is ready to implement for some settings, depending on the value of p (if $\mathbf{L}_p$ norm is chosen) and order of local polynomials (it is more difficult for $r > 1$), though the bookkeeping may be messy. On the other hand, this algorithm is designed for the multivariate case. Its univariate version would not only be much more complex than necessary, but would also produce one-dimensional rings, that is, unions of the subintervals not necessarily neighboring, which are unpleasant and, as it turned out, unnecessary. Our modified algorithms take advantage of the simplicity of the real line topology, simply merging neighboring intervals, thus resulting in partitions consisting of only intervals. These algorithms cannot be easily generalized to multivariate setting, since a procedure of emerging neighboring cubes may generate very complicated and undesirable sets in a partition. This also makes it much more difficult to establish Jackson inequalities for local approximants. We refer the interested reader to [21], where one can find that the proof of Jackson inequality on a ring is already difficult enough. For these reasons, we strongly believe that simpler and more efficient univariate algorithms are necessary.

2. Preliminaries.

Throughout this paper, when we say that f, the function to be approximated, belongs to $\mathbf{L}_p(I)$, we mean $f \in \mathbf{L}_p(I)$ if $0 < p < \infty$, and $f \in \mathbf{C}(I)$ if $p = \infty$. If r is an integer, $0 < \alpha < r$ and $0 < p, q \le \infty$, then the Besov space $\mathbf{B}_q^\alpha(\mathbf{L}_p(I))$ is the set of all functions $f \in \mathbf{L}_p(I)$ such that the semi-(quasi)norm

$$|f|_{\mathbf{B}_q^\alpha(\mathbf{L}_p(I))} := \begin{cases} \left(\int_0^\infty \left[t^{-\alpha} \omega_r(f,t,I)_p \right]^q \frac{dt}{t} \right)^{1/q}, & 0 < q < \infty, \\ \sup\limits_{t>0} t^{-\alpha} \omega_r(f,t,I)_p, & q = \infty, \end{cases}$$

is finite, where ω_r is the usual rth modulus of smoothness. The (quasi)norm for $\mathbf{B}_q^\alpha(\mathbf{L}_p(I))$ is defined by

$$\|\cdot\|_{\mathbf{B}_q^\alpha(\mathbf{L}_p(I))} := \|\cdot\|_{\mathbf{L}_p(I)} + |\cdot|_{\mathbf{B}_q^\alpha(\mathbf{L}_p(I))}.$$

We also define a short notation for a special case that is used frequently in the theory:

$$\mathbf{B}^\alpha(I) := \mathbf{B}_\gamma^\alpha(\mathbf{L}_\gamma(I)), \qquad \gamma := (\alpha + p^{-1})^{-1}.$$

If there is no potential confusion, especially in the case $I = [0,\, 1]$, the interval I will be omitted in the notation for the sake of simplicity. For example, $\mathbf{L}_p$ stands for $\mathbf{L}_p[0,\, 1]$ and $\mathbf{B}^\alpha$ for $\mathbf{B}^\alpha[0,\, 1]$.

If X_0 and X_1 are quasi-normed, complete, linear spaces continuously embedded in a Hausdorff space X, then the K-functional for all $f \in X_0 + X_1$ and $t \ge 0$ is defined as

$$K(f,t,X_0,X_1) := \inf \left\{ \|f_0\|_{X_0} + t\|f_1\|_{X_1} \mid f = f_0 + f_1, f_0 \in X_0, f_1 \in X_1 \right\}.$$

This can be generalized if we replace $\|\cdot\|_{X_1}$ by a quasi-seminorm $|\cdot|_{X_1}$ on X_1:

$$K(f,t,X_0,X_1) := \inf \left\{ \|f_0\|_{X_0} + t|f_1|_{X_1} \mid f = f_0 + f_1, f_0 \in X_0, f_1 \in X_1 \right\}.$$

The interpolation space $(X_0, X_1)_{\theta,q}$, $0 < \theta < 1$, $0 < q \le \infty$, consists of all functions $f \in X_0 + X_1$ such that $|f|_{(X_0,X_1)_{\theta,q}} < \infty$, where

$$|f|_{(X_0,X_1)_{\theta,q}} := \begin{cases} \left(\displaystyle\int_0^\infty \left[t^{-\theta} K(f,t,X_0,X_1) \right]^q \frac{dt}{t} \right)^{1/q}, & 0 < q < \infty, \\ \displaystyle\sup_{t>0} t^{-\theta} K(f,t,X_0,X_1), & q = \infty. \end{cases}$$

When studying an approximation method, it is very revealing to know its approximation spaces, which we now define. Let functions in a quasi-normed linear space X be approximated by elements of its subsets Φ_n, $n = 0, 1, \dots$, which are not necessarily linear but are required to satisfy the assumptions

(i) $\Phi_0 = \{0\}$;
(ii) $\Phi_n \subset \Phi_{n+1}$;
(iii) $a\Phi_n = \Phi_n$ for any $a \neq 0$;
(iv) $\Phi_n + \Phi_n \subset \Phi_{cn}$ where $c := c(\{\Phi_n\})$ does not depend on n;
(v) $\bigcup_{n=0}^\infty \Phi_n$ is dense in X;
(vi) Any $f \in X$ has a best approximation from each Φ_n.

All approximant sets in this paper satisfy these assumptions. Denoting

$$E_n(f) := E(f, \Phi_n)_X := \inf_{\varphi \in \Phi_n} \|f - \varphi\|_X,$$

we define the approximation space

$$A_q^\alpha := A_q^\alpha(X) := A_q^\alpha(X, \{\Phi_n\}), \qquad \alpha > 0,$$

to be the set of all $f \in X$ for which $E_n(f)$ is of order $n^{-\alpha}$ in the sense that the following seminorm is finite:

$$|f|_{A_q^\alpha} := \begin{cases} \left(\displaystyle\sum_{n=1}^\infty [n^\alpha E_n(f)]^q \frac{1}{n} \right)^{1/q}, & 0 < q < \infty, \\ \displaystyle\sup_{n \ge 1} n^\alpha E_n(f), & q = \infty. \end{cases}$$

The general theorem below enables one to characterize an approximation space by merely proving the corresponding Jackson and Bernstein inequalities (see [13, sections 7.5 and 7.9], [9], and [15]).

THEOREM A. *Let $Y := Y_\beta$, $\beta > 0$, be a linear space with a semi-(quasi)norm $|\cdot|_Y$ that is continuously embedded in X. If $\{\Phi_n\}$ satisfies the six assumptions above, and Y satisfies the Jackson inequality*

$$E_n(f) \le C n^{-\beta} |f|_Y, \qquad f \in Y, \tag{2.1}$$

and the Bernstein inequality

$$|\varphi|_Y \le C n^\beta \|\varphi\|_X, \qquad \varphi \in \Phi_n, \tag{2.2}$$

then for all $0 < \alpha < \beta$, $0 < q \le \infty$, the approximation space

$$A_q^\alpha(X, \{\Phi_n\}) = (X, Y)_{\alpha/\beta, q}. \tag{2.3}$$

By a partition $\mathcal{P} = \{I_i\}_{i=1}^n$ of the interval [0, 1] we mean a finite set of subintervals whose union $\cup_{i=1}^n I_i = [0, 1]$, where $I_i := [x_i, x_{i+1})$, $i = 1, 2, \dots, n-1$, $I_n := [x_n, x_{n+1}]$, and $x_1 := 0 < x_2 < \cdots < x_n < 1 =: x_{n+1}$. The (nonlinear) spaces $\Sigma_{n,r}$ of all PP functions of order r on [0, 1] with no more than $n > 0$ pieces are defined by

$$\Sigma_{n,r} := \Sigma_{n,r}[0,\,1] := \{S \mid S(x) = \sum_{I \in \mathcal{P}} P_I(x)\chi_I(x),\ |\mathcal{P}| \le n\},$$

where P_I are in $\mathbf{P}_{r-1}$, the space of polynomials of degree $< r$, and χ_I are the characteristic functions on I. $\Sigma_{0,r}$ is defined as $\{0\}$. These $\Sigma_n = \Sigma_{n,r}$, with r fixed, satisfy all assumptions (i)–(vi) on $\{\Phi_n\}$ (see p. 3). The degree of best approximation of a function f by the elements of $\Sigma_{n,r}$ is denoted by $\sigma_{n,r}(f)_p := E(f, \Sigma_{n,r})_p$.

Remark. *Some authors use the notation $\Sigma_{(n-1)r,r}$ in place of $\Sigma_{n,r}$, since PP functions can be viewed as special kinds of splines with each interior break point x_i, $i = 2, \ldots, n$, considered as a knot of multiplicity r. Also in use is $\mathcal{PP}_{n,r}$. Following general notation in nonlinear approximation, we use the first subscript for the number of coefficients in the approximant. See* [13], [14], [17], [26]*. Strictly speaking, all n-piece PP function of order r only form a proper subset of the free knot spline space $\Sigma_{(n-1)r,r}$, but this subset has the same approximation power in $\mathbf{L}_p$ as the whole space (see Theorem* 12.4.2 *of* [13]*).*

In his 1988 paper [23] (also see [24] and [13, section 12.8]), Petrushev characterized the approximation space $A^\alpha_q(\mathbf{L}_p, \{\Sigma_{n,r}\}_0^\infty)$ using the Besov spaces; see the following theorem.

Theorem B. *Let $0 < p < \infty$, $n > 0$, and $0 < \alpha < r$. Then we have*

$$\sigma_{n,r}(f)_p \le Cn^{-\alpha}|f|_{\mathbf{B}^\alpha} \le Cn^{-\alpha}\|f\|_{\mathbf{B}^\alpha}, \qquad f \in \mathbf{B}^\alpha, \tag{2.4}$$

and

$$\|S\|_{\mathbf{B}^\alpha} \le Cn^\alpha \|S\|_p, \qquad S \in \Sigma_{n,r}. \tag{2.5}$$

Therefore for $0 < q \le \infty$ and $0 < \alpha < \beta < r$

$$A^\alpha_q(\mathbf{L}_p) := A^\alpha_q(\mathbf{L}_p, \{\Sigma_{n,r}\}_0^\infty) = (\mathbf{L}_p, \mathbf{B}^\beta)_{\alpha/\beta,q}. \tag{2.6}$$

In particular, if $\gamma := (\alpha + p^{-1})^{-1}$,

$$A^\alpha_\gamma(\mathbf{L}_p) = (\mathbf{L}_p, \mathbf{B}^\beta)_{\alpha/\beta,\gamma} = \mathbf{B}^\alpha. \tag{2.7}$$

The inequality (2.4) can be proved by finding a balanced partition $\mathcal{P} = \{I_i\}_{i=1}^n$ according to the function

$$G(x) := \int_0^1 \int_0^\infty t^{-\alpha\gamma-2}\chi_{[0,\,t]}(s)\chi_{[0,\,1-rs]}(x)|\Delta_s^r(f,x)|^\gamma\, dsdt \tag{2.8}$$

in the sense that

$$\int_{I_i} G(x)\,dx = \int_{I_j} G(x)\,dx, \qquad i,j = 1,\ldots,n, \tag{2.9}$$

(see [13] for details of the proof). In fact, many Jackson-type inequalities can be proved by showing the existence of a balanced partition (see, e.g., Theorems 12.4.3, 5, and 6 in [13], Theorem 1.1 in [19], and parts of Theorems 2.1 and 4.1 in [17]). We state here Theorem 12.4.6 of [13], given by Burchard [8] in 1974 for the case $p = \infty$ (see also de Boor [3]).

THEOREM C. *Let r and n be positive integers, and let $\gamma := (r + p^{-1})^{-1}$. Let $f \in \mathbf{L}_p[0, 1]$, $1 \le p \le \infty$. If $f \in \mathbf{C}^r(0, 1)$ with $|f^{(r)}(x)| \le \varphi(x)$, where $\varphi \in \mathbf{L}_\gamma$ is a monotone function, then*

$$\sigma_{n,r}(f)_p \le C_r n^{-r} \|\varphi\|_\gamma. \tag{2.10}$$

Let $f \in \mathbf{L}_p[0, 1]$, $0 < \gamma < p \le \infty$, $\alpha := \gamma^{-1} - p^{-1}$, and $r := [\alpha] + 1$. Let $\mathbf{V}_{\gamma,p}$ be the space of functions $f \in \mathbf{L}_p[0, 1]$ for which the variation

$$|f|_{\mathbf{V}_{\gamma,p}} := \sup_{\mathcal{P}} \left(\sum_{I \in \mathcal{P}} \omega_r(f, |I|, I)_p^\gamma \right)^{\frac{1}{\gamma}}$$

is finite, where the sup is taken over all finite partitions $\mathcal{P}$ of [0, 1]. Following [17] (see also Brudnyi [7] and Bergh and Peetre [1]), we define a modulus of smoothness for $f \in \mathbf{V}_{\gamma,p}$ by

$$\Omega(f, t)_{\gamma,p} := \sup_{0<h\le t} \sup_{|\mathcal{P}| \le [h^{-1}]+1} h^\alpha \left(\sum_{I \in \mathcal{P}} \omega_r(f, |I|, I)_p^\gamma \right)^{\frac{1}{\gamma}}. \tag{2.11}$$

The following theorem, which is due to DeVore and Yu [17], provides characterization of $A_q^\alpha(\mathbf{L}_p, \{\Sigma_{n,r}\}_0^\infty)$ using interpolation spaces involving $\mathbf{V}_{\gamma,p}$.

THEOREM D. *Let $0 < p \le \infty$, $0 < \alpha < r$, and $\gamma := (\alpha + p^{-1})^{-1}$. Then for approximation by elements from $\{\Sigma_{n,r}\}_0^\infty$, we have the Jackson inequality*

$$\sigma_{n,r}(f)_p \le C n^{-\alpha} |f|_{\mathbf{V}_{\gamma,p}}, \qquad f \in \mathbf{V}_{\gamma,p}, \tag{2.12}$$

and the Bernstein inequality

$$|S|_{\mathbf{V}_{\gamma,p}} \le C n^\alpha \|S\|_p, \qquad S \in \Sigma_{n,r}. \tag{2.13}$$

Therefore

$$A_q^\alpha(\mathbf{L}_p) := A_q^\alpha(\mathbf{L}_p, \{\Sigma_{n,r}\}_0^\infty) = (\mathbf{L}_p, \mathbf{V}_{\sigma,p})_{\alpha/\beta,q}.$$

In particular, if $p < \infty$,

$$A_\gamma^\alpha(\mathbf{L}_p) = (\mathbf{L}_p, \mathbf{V}_{\sigma,p})_{\alpha/\beta,\gamma} = \mathbf{B}^\alpha.$$

The Jackson inequality (2.12) follows from the definition of $\Omega(f, t)_{\gamma,p}$ and the existence for any $f \in \mathbf{V}_{\gamma,p}$ of an $S \in \Sigma_{n,r}$ with $n := [t^{-1}] + 1$ such that

$$\|f - S\|_p \le C\Omega(f, t)_{\gamma,p}, \tag{2.14}$$

which can be proved (see [17]) by showing the existence of a balanced partition $\mathcal{P} = \{I_i\}_{i=1}^n$ such that

$$\omega_r(f, |I_i|, I_i)_p = \omega_r(f, |I_j|, I_j)_p, \qquad i, j = 1, \dots, n,$$

and then defining S by

$$S(x) := \sum_{i=1}^n P_i(x) \chi_{I_i}(x), \tag{2.15}$$

where P_i are best $\mathbf{L}_p$ approximations to f on I_i from the space $\mathbf{P}_{r-1}$.

3. Adaptive algorithms.

3.1. The original adaptive algorithm. More than likely it will be hard to find an *exactly* balanced partition numerically. An algorithm of this sort by Hu [20], for instance, uses two nested loops (there is another level of loop that increases the number of knots). This is probably one of the reasons why much attention is paid to adaptive approximation, which selects break points by repeatedly cutting the intervals into two equal halves, and produces PP functions with *dyadic* break points, which can be represented by finite binary numbers of the form $m \cdot 2^{-k}$, $0 \le k < \infty$, $0 \le m \le 2^k$. Denote the spaces of such PP functions by $\Sigma^{\mathrm{d}}_{n,r}$ and their approximation errors $E(f, \Sigma^{\mathrm{d}}_{n,r})_p$ by $\sigma^{\mathrm{d}}_{n,r}(f)_p$. We now describe the original adaptive algorithms in the univariate setting.

Let $\mathcal{E}$ be a nonnegative set function defined on all subintervals of [0, 1] which satisfies

$$\mathcal{E}(I) \le \mathcal{E}(J) \quad \text{if } I \subseteq J; \tag{3.1}$$

$$\mathcal{E}(I) \to 0 \quad \text{uniformly as } |I| \to 0. \tag{3.2}$$

Given a prescribed tolerance $\varepsilon > 0$, we say that an interval I is *good* if $\mathcal{E}(I) \le \varepsilon$; otherwise it is called *bad.* We want to generate a partition $\mathcal{G} := \mathcal{G}(\varepsilon, \mathcal{E})$ of [0, 1] into good intervals. If [0, 1] is good, then $\mathcal{G} = \{[0, 1]\}$ is the desired partition; otherwise we put [0, 1] in $\mathcal{B}$, which is a temporary pool of bad intervals. We then proceed with this $\mathcal{B}$ and divide every interval in it into two equal pieces and test whether they are good, in which case they are moved into $\mathcal{G}$, or bad, in which case they are kept in $\mathcal{B}$. The procedure terminates when $\mathcal{B} = \emptyset$ (and, hence, all resulting intervals are good and are in $\mathcal{G}$), which is guaranteed to happen by (3.2).

The set function $\mathcal{E}(I)$ usually depends on the function f that is being approximated and measures the error of approximation of f on I, such as $\int_I G(x)\,dx$ in (2.9), thus will be called the *(error) measure* of I throughout this paper. In the simplest case, $\mathcal{E}(I)$ is taken as the local approximation error of f on $I \subseteq [0, 1]$ by polynomials of degree $< r$:

$$\mathcal{E}(I) := E_r(f, I)_p := \inf_{P \in \mathbf{P}_{r-1}} \|f - P\|_{\mathbf{L}_p(I)}, \tag{3.3}$$

and the corresponding approximant on $\mathcal{G}$ is defined by (2.15). This gives an error

$$\|f - S\|_{\mathbf{L}_p[0,\,1]} \le |\mathcal{G}|^{1/p}\, \varepsilon,$$

where $|\mathcal{G}|$ is the number of intervals in $\mathcal{G}$. One can estimate in different ways

$$a_n(f)_p := a_n(f, \mathcal{E})_p := \inf |\mathcal{G}|^{1/p}\, \varepsilon, \tag{3.4}$$

where the infimum is taken over all $\varepsilon > 0$ such that $|\mathcal{G}| = |\mathcal{G}(\varepsilon, \mathcal{E})| \le n$. See Birman and Solomjak [2] and DeVore [12] for estimates for functions f in Sobolev spaces. Other estimates can be found in Rice [25], de Boor and Rice [6], and DeVore and Yu [18] and the references therein. We only mention the following two results.

Theorem E (see [18, Theorem 5.1]). *Let $r = 1, 2, \ldots$, $\gamma := r^{-1}$, and $f \in \mathbf{C}[0, 1]$. If $f \in \mathbf{C}^r(0, 1)$ with $|f^{(r)}(x)| \le \varphi(x)$, where $\varphi \in \mathbf{L}_\gamma$ is a monotone function such that*

$$\int_I \varphi(x)^\gamma dx \le C_1 n^{-1} \quad \text{if } |I| \le 2^{-n}, \tag{3.5}$$

where C_1 is an absolute constant, then we have

$$a_n(f)_\infty \le Cn^{-r}\|\varphi\|_\gamma. \tag{3.6}$$

Note that compared with Theorem C with $p = \infty$, its free knot counterpart, this theorem has an extra requirement (3.5) on φ.

THEOREM F (see [18, Corollary 3.3]). *Let $0 < p < \infty$, $\alpha > 0$, and $q > \gamma := (\alpha + p^{-1})^{-1}$. Then for $f \in \mathbf{B}^\alpha_\sigma(\mathbf{L}_q)$, $0 < \sigma \le \infty$, we have*

$$a_n(f)_p \le Cn^{-\alpha}|f|_{\mathbf{B}^\alpha_\sigma(\mathbf{L}_q)}. \tag{3.7}$$

Since $|f|_{\mathbf{B}^\alpha} = |f|_{\mathbf{B}^\alpha_\gamma(\mathbf{L}_\gamma)} \le |f|_{\mathbf{B}^\alpha_\sigma(\mathbf{L}_q)}$, we see (3.7) is weaker than (2.4), which is for free knot spline approximation. The reason for this is not hard to see: adaptive algorithms not only select break points from a smaller set of numbers (that is, the set of all finite binary numbers), but they also do it in a special order. Consider $f(x) = \sqrt{x}$ on [0, 1] as an example, a good free knot approximant will have most knots very close to 0 (see examples in [20] and Table 5.2 later in this paper). However, an adaptive algorithm needs at least $n - 1$ knots, $2^{-1}, 2^{-2}, \dots, 2^{1-n}$, before it can put one at 2^{-n} and thus needs more knots than a free knot spline algorithm. Although one classifies adaptive approximation as a special kind of free knot spline approximation (since the knots sequence depends on the function to be approximated), one is far from free when choosing knots. It is considered "more restrictive" (DeVore and Popov [14]) than free knot spline approximation.

We should point out that all theorems mentioned in this subsection are of a Jackson-type, that is, so-called *direct theorems*. Bernstein inequalities (closely related to *inverse theorems*, sometimes referred to also as inverse theorems themselves) for free knot splines, such as (2.5) and (2.13), are valid for all splines, including PP functions produced by adaptive algorithms. The problem is that all Jackson inequalities for the original adaptive algorithms are not strong enough to match those Bernstein inequalities in the sense of Theorem A. From this point of view, Theorems E and F are weaker than they look. We do not know exactly what kind of functions can be approximated by the original adaptive algorithms to a prescribed order, that is, we can not characterize their approximation spaces A^α_q. They do not fully exploit the power of nonlinear approximation, and sometimes they generate too many intervals, many of which may have an error measure much smaller than ε.

As mentioned above, there are two major aspects in which adaptive approximation is different from free knot spline approximation: (a) a smaller set of numbers to choose knots from and (b) a special, and restrictive, way to select knots from the set. It turns out that (b) is the reason for its drawback. Although it is also the reason why adaptive approximation is simple (and we want to keep it that way), it does not mean we have to keep all the knots it produces. In this paper, we modify the usual adaptive algorithm in two ways. The idea is that of splitting AND *merging* intervals/cubes used in a recent paper by Cohen et al. [11]. The two new algorithms generate partitions of [0, 1] with fewer dyadic knots which are nearly balanced in some sense. In section 4, we prove that they have the same approximation order as that of free knot splines.

3.2. Algorithm I. We start with the original adaptive procedure with some $\varepsilon > 0$, which generates a partition $\mathcal{G} = \{I'_i\}_{i=1}^{N'}$ of [0, 1] into good intervals. The number N' may be much larger than it has to be. To decrease it, we merge some of the intervals I'_i. We begin with I'_1 and check the union of I'_1 and I'_2. If it is still a good interval, that is, if its measure $\mathcal{E}(I'_1 \cup I'_2) \le \varepsilon$, we add I'_3 to the union and check whether $\mathcal{E}(I_1' \cup I_2' \cup I_3') \leq \varepsilon$, and we proceed until we find the largest good union $I_1' \cup \cdots \cup I_k'$ in the sense that

$$\mathcal{E}(I_1' \cup \cdots \cup I_k') \leq \varepsilon$$

but

$$\mathcal{E}(I_1' \cup \cdots \cup I_{k+1}') > \varepsilon \quad \text{or} \quad k = N'.$$

We name $I_1' \cup \cdots \cup I_k'$ as I_1. If $k < N'$, we continue with I_{k+1}' and find the next largest good union as I_2. At the end of this procedure, we obtain a modified partition consisting of $N \leq N'$ good intervals $\mathcal{P} = \{I_i\}_{i=1}^N$ for which each union $J_i := I_i \cup I_{i+1}$ is bad,

$$\mathcal{E}(J_i) = \mathcal{E}(I_i \cup I_{i+1}) > \varepsilon, \qquad i = 1, 2, \ldots, N-1.$$

This partition is considered *nearly balanced.* For the size of N we have

$$N \leq 1 + 2 \cdot [N/2] \leq 1 + 2 \sum_{i=1}^{[N/2]} \frac{\mathcal{E}(J_{2i-1})}{\varepsilon}. \tag{3.8}$$

3.3. Algorithm II. Our second algorithm generates a nearly balanced partition in another way. It does not make heavy use of prescribed tolerance ε; rather, it merges intervals with relatively small measures while dividing those with large ones. As in the ordinary adaptive algorithms, we start with dividing [0, 1] into two intervals I_1 and I_2 of equal length. However, this is where the similarity ends. We then compare measures $\mathcal{E}(I_1)$ and $\mathcal{E}(I_2)$ and divide the interval with larger measure into two equal pieces. In the case of equal measure, we divide, rather randomly, the one on the left. Now we have three intervals and are ready for the three-step loop below.

Step 1. Assume there is currently a partition $\{I_i\}_{i=1}^k$ and I_j has the largest measure among all I_i. If $\mathcal{E}(I_{j+1}) < M := \theta \max_i \mathcal{E}(I_i) = \theta \mathcal{E}(I_j)$, where $0 < \theta < 1$ is a fixed parameter, we check the union of $I_{j+1} \cup I_{j+2}$ to see whether its measure $\mathcal{E}(I_{j+1} \cup I_{j+2}) < M$. If so, add the next interval I_{j+3} into the union and check its measure again. We continue until we get a largest union $\cup_{i=j+1}^{j+m_1} I_i$ whose measure is less than M, and replace this union by the intervals it contains. Then, if $j + m_1 < k$, we find the next largest union $\cup_{i=j+m_1+1}^{j+m_1+m_2} I_i$ in the same manner and replace these intervals by their union. Furthermore, we do the same to the intervals to the left of I_j (but keep I_j intact). In this way we obtain a new partition with (the old) I_j still having the largest measure. This partition is nearly balanced in the sense that the measure of the union of any two consecutive new intervals is no less than $\theta \max_i \mathcal{E}(I_i)$ (because these new intervals were largest unions of old intervals). At the end of this step we renumber the new intervals and update the value of k.

Step 2. Check whether the new partition produced in Step 1 is satisfactory using an application-specific criterion, for instance, whether k has reached a prescribed value n or the error is reduced to a certain level. If not, continue with Step 3; otherwise define the final spline by (2.15) and terminate the algorithm.

Step 3. Divide the interval with the largest measure (I_j) into two equal pieces, renumber the intervals, update the values of k and M, and then go back to Step 1.

REMARK. *In Step* 1, *if I_l and I_{l+1} are the two newest intervals (two "brothers" with equal length), one needs only to check $I_{l-1} \cup I_l$ if $l-1, l \neq j$, and/or $I_{l+1} \cup I_{l+2}$ if $l+1, l+2 \neq j$, since other unions of two consecutive intervals have measures no* *less than the value of M in the previous iteration, which is, in turn, no less than the current M. We stated it in the way above only because it shows the purpose of the step more clearly.*

It should be pointed out that one needs to be careful about the stopping criterion in Algorithm II. For example, if it is applied to the characteristic function $f(x) = \chi_{[\sqrt{2}/2,\,1]}(x)$ with $\mathcal{E}(I) = \omega_r(f, |I|, I)_p$, $p < \infty$, after two iterations we will always have $k = 3$ and $\mathcal{E}(I_1) = \mathcal{E}(I_3) = 0$. The break point $\sqrt{2}/2$ in this example can be replaced by any number in (0, 1) which does not have a finite binary representation such as 0.4. If $k \geq n = 4$ is used as the sole stopping criterion, the algorithm will fall into infinite loop. Fortunately, the error in this example still tends to 0; therefore, infinite loop can be avoided by adding error checking in the criterion. The next lemma shows this is the case in general.

LEMMA 3.1. *Let $\mathcal{E}$ be an interval function satisfying* (3.1) *and* (3.2)*, and let $\varepsilon > 0$ and $n > 0$ be prescribed. Then the criterion*

$$\max_i \mathcal{E}(I_i) \leq \varepsilon \quad \text{or} \quad k \geq n \tag{3.9}$$

will terminate Algorithm II.

Proof. We show that if k never exceeds n, then $\max_i \mathcal{E}(I_i) \to 0$ as the number of iterations goes to ∞. Let $0 < \theta < 1$ be fixed. Let $\widehat{M} := \theta \max_i \mathcal{E}(I_i)$ with the max taken at one moment. Fix this $\widehat{M}$ and denote the group of all subintervals in the partition with "large" errors by $G_{\widehat{M}} := \{I_i \mid \mathcal{E}(I_i) \geq \widehat{M}\}$. Let $M := \theta \max_i \mathcal{E}(I_i)$ be as in Step 1, changing from iteration to iteration. We have $\widehat{M} \geq M$ from now on.

We first make a few observations. Since the interval currently having the largest measure is always in $G_{\widehat{M}}$, each iteration cuts a member of $G_{\widehat{M}}$. However, the algorithm will not merge any member $I_i \in G_{\widehat{M}}$ with another interval because $\mathcal{E}(I_i) \geq \widehat{M} \geq M$ already; any union of I_i with another interval would have even larger measure by (3.1). By (3.2), there exists $\eta > 0$ such that $|I_i| > \eta$ for any $I_i \in G_{\widehat{M}}$. Note all intervals in a partition are disjoint, thus the total length of the intervals in $G_{\widehat{M}}$ is no larger than 1, and its cardinal number $|G_{\widehat{M}}| \leq 1/\eta$.

From these observations, we conclude the following. When an iteration cuts a member I_i of $G_{\widehat{M}}$ into two "children" of equal length, one of the three cases will happen: (a) neither child of I_i belongs to $G_{\widehat{M}}$, thus $|I_i| > \eta$ is removed from the total length of $G_{\widehat{M}}$; (b) exactly one of the children belongs to it (hence having a length $> \eta$) and the other child, with the same length $|I_i|/2 > \eta$, is removed from $G_{\widehat{M}}$; or (c) both children belong to it. The case (a) decreases $|G_{\widehat{M}}|$ by 1, (b) keeps it unchanged, and (c) increases it by 1. Now one can see that at most $\left[3/\eta\right] + 1$ iterations will empty $G_{\widehat{M}}$, since at least one third of them will be cases (a) or (b) to keep $|G_{\widehat{M}}| \leq 1/\eta$, which will remove all the total length of $G_{\widehat{M}}$, thus emptying it. This reduces the maximum error $\max_i \mathcal{E}(I_i)$ by a factor $\theta < 1$. Repeat this enough times and the maximum error will eventually tend to 0. ☐

Although (3.2) does not say anything about the convergence rate of $\mathcal{E}(I)$ as $|I| \to 0$, and the proof of the above lemma may make it sound extremely slow, one can expect a fairly fast convergence in most cases. For example, in the case $\mathcal{E}(I) = \omega_r(f, |I|, I)_p$, if f is in the *generalized Lipschitz space* $\mathrm{Lip}^*(\alpha, p) := \mathrm{Lip}^*(\alpha, \mathbf{L}_p[a,\,b])$, $0 < \alpha < r$, that is, if

$$|f|_{\mathrm{Lip}^*} := |f|_{\mathrm{Lip}^*(\alpha,p)} := \sup_{t>0} (t^{-\alpha} \omega_r(f, t, [a,\,b])_p) < \infty,$$

then for any $I \subseteq [a,\, b]$

$$\mathcal{E}(I) = \omega_r(f, |I|, I)_p \le \omega_r(f, |I|, [a,\, b])_p \le |I|^\alpha |f|_{\mathrm{Lip}^*}.$$

We feel it is safe to say that most functions in applications belong to $\mathrm{Lip}^*(\alpha, p)$ with an α reasonably away from 0, at least on subintervals not containing singularities, thus halving an interval often reduces its error by a factor of 2^α.

A natural question that may arise here is: How complex are the new algorithms? We give brief comparisons below to answer this question. Algorithm I is straight forward. It is the original adaptive algorithm with a second (merging) phase added. This phase consists of no more than $N' + N \le 2N'$ merging attempts, where N' is the number of subintervals the original algorithm generates, and N that of the final subintervals. As for Algorithm II, there are two major differences from the original version. The first one, as mentioned in the remark after the algorithm description, is: up to two merging attempts are made after cutting each interval. The other one is in the book-keeping. In the original version, a vector is needed to record errors on all intervals (or to indicate which intervals are bad), while Algorithm II keeps the index of the interval that has the largest error $\mathcal{E}(I)$ in a scalar variable, in addition to the vector containing all errors. This requires a search for the largest element in the vector after each cutting or merging operation.

One can see from above that the new algorithms are not much more complex in terms of programming steps. The added CPU time, in terms of the number of arithmetic operations, results mainly from the evaluations of the error measure $\mathcal{E}(I)$ required by merging operations. Our estimate is that either algorithm uses two or three times as much CPU time as the original algorithm. More information on CPU time will be given in section 5 together with numerical details.

4. Approximation power of the algorithms.

We now show that our modified adaptive algorithms have the full power of nonlinear approximation. More precisely, we prove that they produce piecewise polynomials satisfying the very same Jackson inequalities for free knot spline approximation (with possibly larger constants on the right-hand side since the partitions are not exactly balanced). As we mentioned earlier, the corresponding Bernstein inequalities hold true for all splines; therefore we are really proving that the approximation spaces for the modified adaptive algorithms are the same as those for free knot spline approximation.

We state below our results as three main theorems, parallel to Theorems B, C, and D, respectively. In fact, we can prove most results of this kind for our algorithms, such as Kahane's theorems and its generalization [13, Theorems 12.4.3 and 5], but the proofs would be too similar to the ones below.

We recall that throughout this paper, I_j denotes the interval with largest measure among all I_i in the partition, the union of any two consecutive intervals $J_i = I_i \cup I_{i+1}$ has a measure $\mathcal{E}(J_i) > \mathcal{E}(I_j)$, and J_i is called bad in Algorithm I. All PP functions on the resulting partitions are defined by (2.15).

THEOREM 4.1. *Let n and r be positive integers, and let $0 < p < \infty$, $0 < \alpha < r$, and $\gamma := (\alpha + p^{-1})^{-1}$. If $f \in \mathbf{B}^\alpha$, then the two modified adaptive algorithms* (*with* (i) $\mathcal{E}(I) = \int_I G(x)\,dx$ *and* $\varepsilon = n^{-1}\int_0^1 G(x)\,dx$, *where $G(x)$ is defined in* (2.8) *or* (ii) $\mathcal{E}(I) = E_r(f, I)_p^\gamma$ *and* $\varepsilon = n^{-1}|f|_{\mathbf{B}^\alpha}^\gamma$) *produce PP functions S of* (2.15) *that satisfy the Jackson inequality*

$$\|f - S\|_p \le Cn^{-\alpha}|f|_{\mathbf{B}^\alpha}. \tag{4.1}$$

From Theorem A we obtain the approximation space $A_q^\alpha(\mathbf{L}_p, \{\Sigma_{n,r}^{\mathrm{d}}\}_0^\infty)$ as a by-product. It turns out to be the same as $A_q^\alpha(\mathbf{L}_p, \{\Sigma_{n,r}\}_0^\infty)$, which is not surprising since $\Sigma_{n,r}^{\mathrm{d}}$ is dense in $\Sigma_{n,r}$. The surprising part is that one can get such an approximant using a simple adaptive algorithm.

COROLLARY 4.2. *Let $0 < p < \infty$, $0 < q \le \infty$, $0 < \alpha < \beta < r$, and $\gamma = (\alpha + p^{-1})^{-1}$. For approximation by PP functions in $\Sigma_{n,r}^{\mathrm{d}}$, we have*

$$A_q^\alpha(\mathbf{L}_p) := A_q^\alpha(\mathbf{L}_p, \{\Sigma_{n,r}^{\mathrm{d}}\}_0^\infty) = (\mathbf{L}_p, \mathbf{B}^\beta)_{\alpha/\beta,q}. \tag{4.2}$$

In particular,

$$A_\gamma^\alpha(\mathbf{L}_p) = (\mathbf{L}_p, \mathbf{B}^\beta)_{\alpha/\beta,\gamma} = \mathbf{B}^\alpha. \tag{4.3}$$

Proof of Theorem 4.1. The proofs of the theorem in the cases (i) and (ii) are very similar. We only consider (i) and remark that, in the case (ii), the inequality $\sum_{i=1}^{[N/2]} |f|_{\mathbf{B}^\alpha(J_{2i-1})}^\gamma \le C|f|_{\mathbf{B}^\alpha}^\gamma$ plays the major role.

PP approximants produced by Algorithm I. Let $\mathcal{E}(I) := \int_I G(x)\,dx$, where G is as in (2.8), and $\varepsilon := n^{-1}M^\gamma$, where

$$M := \left(\int_0^1 G(x)\,dx\right)^{1/\gamma}.$$

We claim that the number N of intervals it produces is no greater than $2n+1$. Indeed, by (3.8)

$$\begin{aligned} N &\le 1 + 2\sum_{i=1}^{[N/2]} \frac{\mathcal{E}(J_{2i-1})}{\varepsilon} \le 1 + 2\sum_{i=1}^{[N/2]} \frac{\int_{J_{2i-1}} G(x)\,dx}{\varepsilon} \\ &\le 1 + \frac{2n}{M^\gamma}\int_0^1 G(x)\,dx = 2n+1. \end{aligned}$$

The rest of the proof of (4.1) is similar to that of (2.4) (cf. section 12.8, p. 386 of [13]); we sketch it here for completeness. It is proved in [13] that for any $f \in \mathbf{B}^\alpha[0,1]$, M is equivalent to $|f|_{\mathbf{B}^\alpha[0,1]}$ with constants of equivalence depending only on r and γ, and that for such an f

$$E_r(f,I)_p \le C|f|_{\mathbf{B}^\alpha(I)} \le C\left(\int_I G(x)\,dx\right)^{1/\gamma} = C\mathcal{E}(I)^{1/\gamma}. \tag{4.4}$$

Define the approximant S by (2.15) and we have

$$\begin{aligned} \|f - S\|_p^p &= \sum_{i=1}^N E_r(f,I_i)_p^p \le C\sum_{i=1}^N \mathcal{E}(I_i)^{p/\gamma} \le CN\varepsilon^{p/\gamma} \\ &\le cn \cdot n^{-p/\gamma}M^p = Cn^{-p\alpha}M^p \le Cn^{-p\alpha}|f|_{\mathbf{B}^\alpha}^p; \end{aligned}$$

here in the fifth step we have used the equality $\gamma = (\alpha + p^{-1})^{-1}$, and in the last step we have used the equivalence of M and $|f|_{\mathbf{B}^\alpha}$.

PP approximants produced by Algorithm II. Let $\mathcal{E}(I)$, M, and ε be the same as above, and use (3.9) as stopping criterion in Step 2. If the algorithm terminates due to $\max_i \mathcal{E}(I_i) \le \varepsilon$ (thus giving less than n pieces), it is the same situation as with Algorithm I. Otherwise we have n pieces when it terminates, and (4.1) follows:

$$\begin{aligned}\|f-S\|_p^p &= \sum_{i=1}^n E_r(f,I_i)_p^p \le C\sum_{i=1}^n \biggl(\int_{I_i} G(x)\,dx\biggr)^{p/\gamma} \\ &\le Cn\biggl(\int_{I_j} G(x)\,dx\biggr)^{p/\gamma} \le Cn\biggl(\frac{2}{n}\sum_{i=1}^{[n/2]}\int_{J_{2i-1}} G(x)\,dx\biggr)^{p/\gamma} \\ &\le \frac{Cn}{n^{p/\gamma}}\biggl(\int_0^1 G(x)\,dx\biggr)^{p/\gamma} = Cn^{-p\alpha}M^p \le Cn^{-p\alpha}|f|_{\mathbf{B}^\alpha}^p. \qquad \square\end{aligned}$$

THEOREM 4.3. *Under the conditions of Theorem C, the modified adaptive algorithms (with $\mathcal{E}(I) := \int_I \varphi(x)^\gamma dx$ and $\varepsilon := n^{-1}\int_0^1 \varphi(x)^\gamma dx$) produce PP approximants S in $\Sigma_{n,r}^{\mathrm{d}}$ that satisfy the Jackson inequality:*

$$\sigma_{n,r}^{\mathrm{d}}(f)_p \le \|f-S\|_p \le Cn^{-r}\|\varphi\|_\gamma. \tag{4.5}$$

Proof of Theorem 4.3.
PP approximants produced by Algorithm I. Let

$$\mathcal{E}(I) := \int_I \varphi(x)^\gamma dx, \quad M := \biggl(\int_0^1 \varphi(x)^\gamma dx\biggr)^{1/\gamma} = \|\varphi\|_\gamma, \quad \text{and } \varepsilon := n^{-1}M^\gamma.$$

Defining P_i as the Taylor polynomial for f of degree $r-1$ at the point x_{i+1} (not best $\mathbf{L}_p$ approximation), we have (see equation (4.15) in Chapter 12 of [13])

$$\|f-P_i\|_{\mathbf{L}_p(I)} \le C\|\varphi\|_{\mathbf{L}_\gamma(I)} = C\mathcal{E}(I)^{1/\gamma}. \tag{4.6}$$

Using (4.6) in place of (4.4), then (4.5) for $p<\infty$ can be proved by arguments very similar to those in the proof of Theorem 4.1 by Algorithm I. We also refer the reader to the proof of Theorem C in [13]. For $p=\infty$, the estimate of N is the same and we need only to replace $\bigl(\sum_{i=1}^N \|f-P_i\|_{\mathbf{L}_p(I_i)}^p\bigr)^{1/p}$ by $\max_{1\le i\le N}\|f-P_i\|_{\mathbf{L}_\infty(I_i)}$:

$$\begin{aligned}\|f-S\|_\infty &= \max_{1\le i\le N}\|f-P_i\|_{\mathbf{L}_\infty(I_i)} \le C\max_{1\le i\le N}\|\varphi\|_{\mathbf{L}_\gamma(I_i)} \\ &= C\max_{1\le i\le N}\mathcal{E}(I_i)^{1/\gamma} \le Cn^{-1/\gamma}M = Cn^{-r}\|\varphi\|_\gamma.\end{aligned}$$

PP approximants produced by Algorithm II. Let $\mathcal{E}(I)$, ε, and M be the same as above. Use (3.9) again as the stopping criterion in Step 2. If the algorithm terminates because $\max_i \mathcal{E}(I_i) \le \varepsilon$, it is the same situation as in Algorithm I. Otherwise, for $p<\infty$ we have n intervals, and

$$\begin{aligned}\|f-S\|_p^p &= \sum_{i=1}^n \|f-P_i\|_{\mathbf{L}_p(I_i)}^p \le C\sum_{i=1}^n \|\varphi\|_{\mathbf{L}_\gamma(I_i)}^p \le Cn\bigl(\|\varphi\|_{\mathbf{L}_\gamma(I_j)}^\gamma\bigr)^{p/\gamma} \\ &\le Cn\left(\frac{2}{n}\sum_{i=1}^{[n/2]}\|\varphi\|_{\mathbf{L}_\gamma(J_{2i-1})}^\gamma\right)^{p/\gamma} \le Cn^{1-p/\gamma}M^p = Cn^{-rp}M^p,\end{aligned}$$

where we have used the inequality (4.6) in the second step, and $\gamma = (r + 1/p)^{-1}$ in the last one. For $p = \infty$, we make similar changes to those in Algorithm I:

$$\|f - S\|_\infty = \max_{1\le i\le n} \|f - P_i\|_{\mathbf{L}_\infty(I_i)} \le C \max_{1\le i\le n} \|\varphi\|_{\mathbf{L}_\gamma(I_i)} = C\|\varphi\|_{\mathbf{L}_\gamma(I_j)}$$
$$\le C\left(\frac{2}{n}\sum_{i=1}^{[n/2]} \|\varphi\|^\gamma_{\mathbf{L}_\gamma(J_{2i-1})}\right)^{1/\gamma} \le Cn^{-1/\gamma}M = Cn^{-r}M. \qquad \square$$

THEOREM 4.4. *Let n and r be positive integers, and let $0 < p, \le \infty$, $0 < \alpha < r$, and $\gamma := (\alpha + p^{-1})^{-1}$. If $f \in \mathbf{V}_{\gamma,p}$, then the two modified adaptive algorithms* (*with $\mathcal{E}(I) := E_r(f, I)^\gamma_p$ and $\varepsilon := n^{-\gamma/p}\Omega(f, 1/n)^\gamma_{\gamma,p}$*) *produce PP functions S of* (2.15) *that satisfy the Jackson inequality*

$$\|f - S\|_p \le Cn^{-\alpha}|f|_{\mathbf{V}_{\gamma,p}}. \tag{4.7}$$

Using Theorems 4.4 and A we have the following characterization of $A^\alpha_q(\mathbf{L}_p, \{\Sigma^{\rm d}_{n,r}\}^\infty_0)$.

COROLLARY 4.5. *For approximation by PP functions in $\Sigma^{\rm d}_{n,r}$ we have*

$$A^\alpha_q(\mathbf{L}_p) := A^\alpha_q(\mathbf{L}_p, \{\Sigma^{\rm d}_{n,r}\}^\infty_0) = (\mathbf{L}_p, \mathbf{V}_{\sigma,p})_{\alpha/\beta,q}.$$

In particular, if $p < \infty$,

$$A^\alpha_\gamma(\mathbf{L}_p) = (\mathbf{L}_p, \mathbf{V}_{\sigma,p})_{\alpha/\beta,\gamma} = \mathbf{B}^\alpha.$$

Proof of Theorem 4.4. It suffices to show (2.14) since (4.7) immediately follows from it with any $t > 0$ and $n := [t^{-1}] + 1$ (see the end of section 2). We only prove it for $p < \infty$. The case of $p = \infty$ can be verified by making changes similar to those in the proof of the $\mathbf{L}_\infty$ case in the previous theorem.

PP approximants produced by Algorithm I. Let $\mathcal{E}(I) := E_r(f, I)^\gamma_p$ and $\varepsilon := n^{-\gamma/p}\Omega(f, t)^\gamma_{\gamma,p}$. From (3.8), the number N of intervals the algorithm produces can be estimated as

$$N \le Cn.$$

Indeed, if $N > 2n$ (otherwise, it's done) we have

$$N \le 1 + 2\sum_{i=1}^{[N/2]} \frac{E_r(f, J_{2i-1})^\gamma_p}{n^{-\gamma/p}\Omega(f, t)^\gamma_{\gamma,p}} \le \frac{Cn^{\gamma/p}}{\Omega(f, t)^\gamma_{\gamma,p}} \sum_{i=1}^{[N/2]} \omega_r(f, |J_{2i-1}|, J_{2i-1})^\gamma_p \le Cn^{\gamma/p}N^{\alpha\gamma},$$

where we have used the definition (2.11) of $\Omega(f, t)_{\gamma,p}$. Since $1 - \alpha\gamma = \gamma/p$, this gives $N \le Cn$. Now (2.14) follows, since

$$\|f - S\|^p_p = \sum_{i=1}^{N} \|f - P_i\|^p_{\mathbf{L}_p(I_i)} = \sum_{i=1}^{N} E_r(f, I_i)^p_p$$
$$\le CN\big(n^{-\gamma/p}\Omega(f, t)^\gamma_{\gamma,p}\big)^{p/\gamma} \le C\Omega(f, t)^p_{\gamma,p}.$$

PP approximants produced by Algorithm II. We set $\mathcal{E}(I) := E_r(f, I)^\gamma_p$, and use $n := [t^{-1}] + 1$ and $\varepsilon := n^{-\gamma/p}\Omega(f, t)^\gamma_{\gamma,p}$ in the stopping criterion (3.9). If it stops because $\max_i \mathcal{E}(I_i) \le \varepsilon$, we have exactly the same situation as with Algorithm I, with the same partition; otherwise there are n intervals when it terminates. In the latter case, we have

$$\|f - S\|_p = \left(\sum_{i=1}^{n} \|f - P_i\|^p_{\mathbf{L}_p(I_i)}\right)^{1/p} \leq C \left(\sum_{i=1}^{n} \omega_r(f, |I_i|, I_i)^p_p\right)^{1/p}$$
$$\leq Cn^{1/p}\omega_r(f, |I_j|, I_j)_p \leq Cn^{1/p} \left(\frac{1}{n}\sum_{i=1}^{[n/2]} \omega_r(f, |J_{2i-1}|, J_{2i-1})^\gamma_p\right)^{1/\gamma}$$
$$\leq Cn^{1/p} \cdot n^{-1/\gamma} \cdot n^\alpha \Omega(f, t)_{\gamma,p} = C\Omega(f, t)_{\gamma,p}. \qquad \square$$

5. Numerical implementation and examples. Theoretically, the two algorithms have the same approximation power. However, when it comes to numerical implementation, we prefer Algorithm II since it directly controls the number of polynomial pieces n, while ε in Algorithm I is neither a power of n nor a tolerance for $\|f - S\|$ (though it is closely related to both). We implemented Algorithm II on the computer, using Fortran 90 and mainly for $p = 2$. The error measure used in the code is $\mathcal{E}(I) = E_r(f, I)^2_2$ unless we have a better one to use, such as $\int_I \varphi^\gamma := \int_I |f^{(r)}|^\gamma$ for the square root function in the first example in this section. The $\mathbf{L}_2$ norm of f on the interval $I_i = [t_i, t_{i+1}]$ is estimated by the composite Simpson rule for integral

$$\int_{I_i} [f(x)]^2 dx \approx \frac{h}{3}\big([f(x_1)]^2 + 4[f(x_2)]^2 + 2[f(x_3)]^2 + 4[f(x_4)]^2 + \cdots + 2[f(x_{n_p-2})]^2 + 4[f(x_{n_p-1})]^2 + [f(x_{n_p})]^2\big), \tag{5.1}$$

and its $\mathbf{L}_\infty$ norm is estimated by

$$\max_{1\leq i\leq n_p} |f(x_i)|, \tag{5.2}$$

where $t_i = x_1 < x_2 < \cdots < x_{n_p} = t_{i+1}$ are equally distributed nodes, n_p is a program parameter roughly set as 6 times r, and $h = (t_{i+1} - t_i)/n_p$. The best $\mathbf{L}_2$ polynomial approximant on I_i, discretized by (5.1) as an overdetermined $n_p \times r$ system of linear equations for the least squares method, is calculated by either QR decomposition or singular value decomposition by calling LINPACK subroutines `Sqrdc` and `Sqrsl`, or `Ssvdc` (or their double precision counterparts). The latter takes longer but we did not see any difference in the first four or five digits of the local approximation errors they computed; thus we did not test it extensively.

The $\mathbf{L}_\infty$ version of algorithm is basically the same, except that we use $\mathcal{E}(I) = \|f - P_I\|_\infty$, estimated by (5.2). The local polynomials P_I (and the global smooth splines) are still obtained by the least squares method, that is, still best $\mathbf{L}_2$ approximants. This is common in the literature, and it is justified by the fact that the best $\mathbf{L}_2$ polynomial approximant is also a near-best $\mathbf{L}_\infty$ polynomial on the same interval; see Lemma 3.2 of DeVore and Popov [16].

The number of polynomial pieces is used as the main termination criterion, while ε in (3.9) is set to a small value mainly to protect the program from falling into infinite loops, rather than the sophisticated ones as in proofs in the previous section. It turned out that infinite loop is not a problem. A nonfull rank matrix in the least squares method is a problem, which happens far before it falls into an infinite loop. This is because if I_i is too small, the machine will have difficulties distinguishing the n_p

points needed in (5.1). Therefore, we added a third condition to protect the program from failing: stop the program when

$$t_{i+1} - t_i < n_p \cdot \max(|t_i|, |t_{i+1}|) \cdot \varepsilon_m. \tag{5.3}$$

We also added a second part in the code, namely, finding an $\mathbf{L}_2$ *smooth* spline approximation to the function with the knot sequence $\{t_i\}_{i=2-r}^{n+r}$, where the interior knots $a < t_2 < t_3 < \cdots < t_n < b$ are the break points of the PP function obtained by Algorithm II, used as *single* knots, and the auxiliary knots are set as $t_{2-r} = t_{3-r} = \cdots = t_1 = a$, and $t_{n+1} = t_{n+2} = \cdots = t_{n+r} = b$. Despite the fact that the partitions are guaranteed to be good only for PP functions, they usually work well for smooth splines, too. De Boor gave some theoretical justification in the discussion of his subroutine `Newnot` [4, Chapter XII].

The least square objective function for finding this smooth spline $\bar{S}$ is

$$\sum_{j=1}^{n_s n+1} w_j [f(\tau_j) - \bar{S}(\tau_j)]^2, \tag{5.4}$$

where n_s, set as $5r+1$, is the number of equal pieces into which we cut each subinterval I_i, τ_j are the points resulted from such cutting, and the weights w_j are chosen so that (5.4) becomes a composite trapezoidal rule for the integral $\int_a^b \left[f(x) - \bar{S}(x)\right]^2 dx$: $w_j = (t_{i+1} - t_i)/n_s$ if $\tau_j \in (t_i, t_{i+1})$; $w_j = (t_{i+1} - t_{i-1})/(2n_s)$ if $\tau_j = t_i$ for some $2 \le i \le n$; $w_1 = (t_2 - a)/(2n_s)$ and $w_{n_s n+1} = (b - t_n)/(2n_s)$. The actual calculation of the B-spline coefficients of

$$\bar{S}(x) = \sum_{i=2-r}^{n} c_i B_{ir}(x),$$

where $N_{ir}(x) := N(x; t_i, \ldots, t_{i+r})$ are the B-splines with the knot sequence $\{t_i\}$ scaled so that $\sum N_{ir}(x) \equiv 1$, is done by de Boor's subroutine `L2Appr` in [4, Chapter XIV]. We used the source code of all the subroutines in the book from the package PPPACK on the Internet.

We tested our code on a SUN UltraSparc, with a clock frequency 167MHz, 128MB of RAM, and running Solaris 2.5.1. The speed is so fast that it is not an issue here: for finding break points, it is somewhere from 0.015 second for $n = 6$ to 0.1 second for $n = 30$ when printing minimum amount of messages on the screen, and it is less than 10% of these for computing smooth splines. We also tested the code on a 300 MHz Pentium II machine with 64 MB of RAM running Windows NT 4.0. The speed is at least three times as fast. None of the problems we tested used more than 0.1 second. (The reason for the great difference in speed may be that the SUN we used is a file server, not ideal for numerical computation.) There is still room for improvement in efficiency. For example, one can use a value of n_p, larger than what we use, at the beginning and decrease it as n increases (and the error on each subinterval decreases). The value of n_s should be related to n, too, for the same reason.

The main cost of CPU time is the evaluation of the error measure $\mathcal{E}(I)$ for each subinterval I. We use $\mathcal{E}(I) = E_r(f, I)_2^2$, estimated by QR decomposition, as an example. Each such problem involves n_p function evaluations, and $(n_p - \frac{r}{3})r^2$ arithmetic operations required in QR decomposition, plus some more for estimating the error from the resulting matrices. Each cutting of intervals requires two $\mathcal{E}(I)$ evaluations,

TABLE 5.1
Approximation order of $f(x) = \sqrt{x}$ on [0, 1].

r	1	1	2	2	3	3	4	4	5	5
n	2–60	11–60	2–60	11–60	2–60	11–60	2–60	11–60	2–60	11–60
α ($\mathbf{L}_2$)	1.00	1.00	1.95	1.99	2.91	2.98	3.89	4.20	4.91	5.48
α ($\mathbf{L}_\infty$)	0.98	0.99	1.90	1.96	2.81	3.05	3.62	4.05	4.42	4.97

and each merging attempt requires one. Our numerical experiments show that a typical run resulting in n subintervals cuts intervals about $2n$ time. Each cutting results in up to two attempts of merging subintervals. That gives about $8n$ least squares problems, each of which involves n_p function evaluations plus about $n_p r^2$ arithmetic operations. In view of the approximation order we proved in the previous section, and the fact that n_p is roughly a multiple of r, we think it pays to use a relatively large r, at least 4 or 5. For $r = 5$, the error will reach the machine epsilon (single precision) when n is somewhere between 30 and 70 in most cases.

We use the square root function $f(x) = \sqrt{x}$ to test the PP function approximation order. This function is only in the Lipschitz space $\mathrm{Lip}(\frac{1}{2}, \mathbf{L}_\infty)$, thus the approximation order is only 1/2 for splines with equally spaced knots in the $\mathbf{L}_\infty$ norm, no matter what their order r is. By Theorem 4.3, we should have $e_n := \|f - S_n\|_p \le Cn^{-r}$, where S_n is the function consisting of n polynomial pieces computed by Algorithm II using $\mathcal{E}(I) = \int_I \varphi(x)^\gamma \, dx := \int_I |f^{(r)}(x)|^\gamma \, dx$, and we have combined $\|\varphi\|_\gamma$ in the theorem into the constant C. After the knot sequence has been found, QR decomposition is used at the end of the program on each subinterval to estimate e_n. Since the error decreases fast for $r = 5$, double precision had to be used in QR decomposition for large values of n. Assume that what we actually obtain from the code is $e_n = Cn^{-\alpha}$, where α is the approximation order. Since $\log e_n = \log C - \alpha \log n$, if we plot the points $(x_n, y_n) := (\log n, \log e_n)$ in the plane, they should form a line. Since such a plot zigzags very much, we calculated the least squares line $y = -\alpha x + b$ fitting (x_n, y_n) to find the order. Table 5.1 gives values of α for different r using both $\mathbf{L}_2$ and $\mathbf{L}_\infty$ norms. We should mention that the points (x_n, y_n) for small values of n are too low and ruin the obvious line pattern formed by those for larger n, thus we give two values of α, one from the points for $n = 2, \dots, 60$, and the other from $n = 11, \dots, 60$. As can be seen from the table, the latter values are right around or even exceed r.

REMARK. *We tried some power of $E_r(f, I)_p$ for $\mathcal{E}(I)$ and felt, in view of* (4.6), *it would yield a better balance of subintervals, thus a higher order. But the orders so obtained were well below r* (4.46 *for $r = 5$ and $p = \infty$, e.g.*). *The reason might be that $\int_I \varphi^\gamma$ is additive, but (power of) $E_r(f, I)_p$ is not.*

To illustrate the advantage of interval merging, we compare the original adaptive algorithm and our modified ones with the function

$$f(x) = \frac{\log_2(2^{-m} + x)}{-m}.$$

This function is in $\mathbf{C}^\infty$, and is decreasing and convex on [0, 1] with $f(0) = 1$ and $f(1) \approx -2^{-m}/(m \ln 2)$. We use $m = 50$, $r = 1$, and $\mathcal{E}(I_i) = E_1(f, I_i)_\infty$. Note that $E_1(f, I_i)_\infty = (f(x_i) - f(x_{i+1}))/2$ since f is decreasing on [0, 1]. Table 5.2 shows comparison in numbers of knots produced for the same approximation error by the original adaptive algorithm and our Algorithm II. Both programs try to put first knots near $x = 0$ where the graph is very steep. The original algorithm has to, as pointed out early, lay down knots $2^{-1}, 2^{-2}, \dots, 2^{-23}$ one by one before reaching an error of

Table 5.2

Comparison in numbers of interior knots produced by the original and modified adaptive algorithms for the same error in approximating $f(x) = -\log_2(2^{-m} + x)/m$.

Error	0.27	0.23	0.14	0.12	0.060	0.032
Original	23	27	36	38	44	47
Alg. II	1	2	3	5	10	15

0.27, while Algorithm II, after trying all these knots one at a time and merging all but the last interval, puts the very first knot at 2^{-23}.

It is interesting to watch how Algorithm II moves a knot toward a better position in successive iterations without increasing the total number of pieces. The following screen output shows that in iterations 1 and 2 the program moves the break point 0.5 to 0.25 and then to 0.125, while the error decreases form 0.5 to 0.47; in iterations 3–22 it moves the break point all the way to $2^{-23} \approx 1.192 \times 10^{-7}$ with the error decreased to 0.27. What happened internally is, in iteration 1, e.g., it cuts the interval [0, 0.5] into [0, 0.25] and [0.25, 0.5]. Since the error on the union of [0.25, 0.5] and [0.5, 1] is smaller than that on [0, 0.25], it then merges the two intervals into [0.25, 1]. The net effect of these steps is moving the break point 2^{-1} to 2^{-2}.

```
Iteration  0:  # of intervals =  2,   Knots & errors:
 errors=
   4.90000E-01  1.00000E-02
 L_\infty error on [a, b] =  4.90000E-01
 knots=
   0.00000E+00  5.00000E-01  1.00000E+00

Iteration  1:  # of intervals =  2,   Knots & errors:
 errors=
   4.80000E-01  2.00000E-02
 L_\infty error on [a, b] =  4.80000E-01
 knots=
   0.00000E+00  2.50000E-01  1.00000E+00

Iteration  2:  # of intervals =  2,   Knots & errors:
 errors=
   4.70000E-01  3.00000E-02
 L_\infty error on [a, b] =  4.70000E-01
 knots=
   0.00000E+00  1.25000E-01  1.00000E+00

    (Many lines deleted....)

Iteration 22:  # of intervals =  2,   Knots & errors:
 errors=
   2.70000E-01  2.30000E-01
 L_\infty error on [a, b] =  2.70000E-01
 knots=
   0.00000E+00  1.19209E-07  1.00000E+00
```

Table 5.3
Approximation errors to the Runge function on $[-5, 5]$.

n	$\|f - S_n\|_2/\sqrt{10}$	$\|f - \bar{S}_n\|_2/\sqrt{10}$	$\|f - \bar{S}_n\|_\infty$	$\|f - \bar{S}_n\|_\infty$ Hu	$\|f - \bar{S}_n\|_\infty$ LM
3	0.0035	0.029	0.074	0.079	0.070
5	0.00084	0.0058	0.013	0.0035	0.0032
7	0.00029	0.0039	0.0098		
9	0.00011	0.00087	0.0023		
11	0.000039	0.00018	0.00091	0.00086	0.00090

We now consider the infamous Runge function, which is also in $\mathbf{C}^\infty$ but, on the other hand, is hard to interpolate or approximate. Lyche and Mørken [22] approximated it by the knot removal algorithm, and Hu [20] approximated it by balancing the rth derivative of the function on subintervals in two nested loops. Here and in the rest of the paper, we use $r = 4$. In Table 5.3, we compare our results with those of Lyche and Mørken (LM) [22] and Hu [20]. For the same number of knots (that is, $n - 1$), we list our errors measured in $\|\cdot\|_2/\sqrt{b-a}$ for the PP function S_n and the smooth spline $\bar{S}_n$, also that of $\bar{S}_n$ measured in $\mathbf{L}_\infty$ norm. We divide the $\mathbf{L}_2$ norm by $\sqrt{b-a}$ since it is more comparable to the $\mathbf{L}_\infty$ norm, which is what LM and Hu used. The errors by LM are estimated from figures in [22]. Because of the simple nature of our algorithm, we only expected to compete with their results by splines with two or three times as many knots. It turns out that our approximation errors are almost as good as theirs, which were produced by more sophisticated methods.

By now, the reader may begin to wonder: what is the effect of the parameter $0 < \theta < 1$ in Step 1 of Algorithm II, used in Lemma 3.9 to guarantee the termination of Algorithm II. We tried $\theta = 1$ with all functions we tested, it worked excellently except that the number of polynomial pieces went up and down a few times with the square root function using $\mathcal{E}(I) = \int_I |f^{(r)}(x)|^\gamma \, dx$, in which case $\theta = 0.9$ was used instead. It is true that in theory it might get into an infinite loop, but since our goal is to find a nearly balanced partition, $\theta = 1$ works better in this aspect, provided infinite loop does not happen. It did not. As a matter of fact, sometimes we feel the need for a value slightly larger than 1, e.g., with symmetric functions such as the Runge function. What happens with $\theta \le 1$ is that if there are two subintervals having the same largest measure at the moment, symmetric about the center of the interval, then the outcome of the next iteration, which processes the subinterval on the left, will very often interfere with the processing of the subinterval on the right later. It may not make the approximation error worse, at least not by much, it is just that the knot sequence becomes unsymmetrical, thus unnatural and unpleasant. Furthermore, most algorithms in the literature produce symmetric knots for symmetric functions; it would be hard to compare our results with theirs. For these minor reasons, we used $\theta = 1.01$ in preparation of Table 5.3. In the next example, we consider the PP function

$$f(x) = \begin{cases} x^2 & \text{if } x \le \sqrt{2}/2, \\ 1 & \text{otherwise,} \end{cases}$$

which has a jump at $\sqrt{2}/2$. As we mentioned in the discussion before Lemma 3.1, since $\sqrt{2}/2$ has no finite binary representation, this function can never be approximated exactly by a PP function with dyadic break points. The program (with $p = \infty$) keeps cutting and merging around the jump (since the number of pieces is always 3 after two iterations), until it is stopped by the criterion (5.3), resulting in $t_2 = 0.70710659$

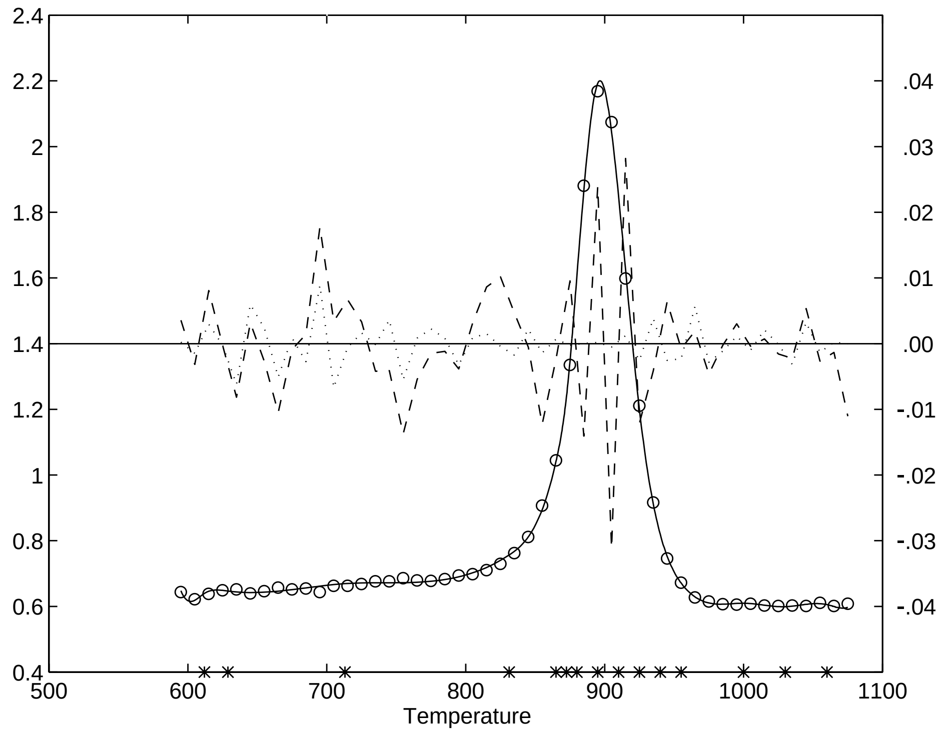

FIG. 5.1. *Titanium Heat Data (circles). The final spline (solid line) has* 15 *interior knots. The errors for preapproximation (dotted) and for the final spline (dashed) use scales on the right.*

and $t_3 = 0.70710754$. The PP function matches f exactly on the computer screen since the two points are indistinguishable. One can very well combine them into a single break point, thus virtually *reproducing* f. The original adaptive algorithm, in contrast, would put many many knots around the jump while trying to narrow the subinterval containing the jump: $0.5, 0.75, 0.625, 0.6875, 0.71875, \ldots$. All these knots are useless except the newest two.

In practice, one often wants to approximate discrete data points other than known functions as in the previous examples. In this case, we preapproximate the points by a spline with as many parameters as we wish to use, then apply our algorithm to this spline. For smooth-looking data, we interpolate the data by a $\mathbf{C}^1$ cubic spline with knots at the data points, using de Boor's subroutine `Cubspl` in [4]. This worked very well. We produced some sample data points from the Runge function and square root function and applied this approach to them. It resulted in virtually the same knot sequences as those generated by directly approximating the original functions.

In the real world, however, it is likely that the data will contain errors. If the data points are interpolated, one can see small wiggles in the graph, which tricks the program laying knots in areas where the curve is otherwise flat. One such example is the Titanium Heat Data (experimentally determined), see [4, Chapter XIII], and also LM [22] and Hu [20]. In Figure 5.1 the reader can see wiggles on both the left and right. De Boor [4, Chapter XIV] suggests that the data be approximated by a less smooth spline. We absolutely agree. For the same reason, we used fewer knots for preapproximating spline in the flat parts at both ends, than we did near the high peak around 900°, trying to ignore the wiggles. In fact, we used almost the same knot sequence for preapproximating spline as in Figure 4 of [20].

TABLE 5.4
Approximation errors to the Titanium Heat Data.

Obtained by	Order	# of knots	Error
De Boor	6	15	0.032
LM	4	6	0.045
Hu	4	10	0.048
Hu	4	13	0.024
Alg. II	4	11	0.070
Alg. II	4	15	0.031

Since de Boor, LM, and Hu all used $\mathbf{L}_\infty$ norm for approximating these data, we also used the $\mathbf{L}_\infty$ version of our program. Figure 5.1 shows a cubic spline approximation to the Titanium Data obtained by this method. It has 15 interior knots with an error of 0.031. Table 5.4 gives a comparison of our results with those by others on the same data.

Acknowledgments. We are deeply indebted to Professor Ron DeVore, who inspired us by discussing the excellent ideas in [11] during our visit to the University of South Carolina. We want to thank him and Professors Pencho Petrushev and Albert Cohen for providing us with drafts of their manuscript [11]. Credit is also due to Professor Dietrich Braess, the editor of this paper, and the referees, whose opinions and suggestions helped very much in improving the manuscript. As a matter of fact, we reshaped the last section during the communication with them.

REFERENCES

[1] J. Bergh and J. Peetre, *On the space V_p $(0 < p \leq \infty)$*, Bol. Un. Mat. Ital. (4), 10 (1974), pp. 632–648.
[2] M. Birman and M. Solomjak, *Piecewise polynomial approximation of functions of classes* $\mathbf{W}_p^\alpha$, Mat. Sb (N.S.), 73 (1967), pp. 331–355.
[3] C. de Boor, *Good approximation by splines with variable knots*, in Spline Functions and Approximation, A. Meir and A. Sharma, eds., Birkäuser, Basel, 1973, pp. 57–72.
[4] C. de Boor, *A Practical Guide to Splines*, 4th ed., Springer-Verlag, New York, 1987.
[5] C. de Boor and J. R. Rice, *Least squares cubic spline approximation* II*–Variable knots*, CSD TR 21, Purdue University Report, Lafayette, IN, 1968.
[6] C. de Boor and J. R. Rice, *An adaptive algorithm for multivariate approximation giving optimal convergence rates*, J. Approx. Theory, 25 (1979), pp. 337–359.
[7] Yu. Brudnyi, *Spline approximation and functions of bounded variation*, Dokl. Akad. Nauk SSSR, 215 (1974), pp. 511–513.
[8] H. Burchard, *Splines with optimal knots are better*, Appl. Anal., 3 (1974), pp. 309–319.
[9] P. L. Butzer and K. Scherer, *Jackson and Bernstein-type inequalities for families of commutative operators in Banach spaces*, J. Approx. Theory, 5 (1972), pp. 308–342.
[10] A. Cohen, W. Dahmen, and R. A. DeVore, *Adaptive wavelet methods for elliptic operator equations–Convergence rates*, Math. Comp., to appear.
[11] A. Cohen, R. A. DeVore, P. Petrushev, and H. Xu, *Nonlinear approximation and the space $BV(\mathbf{R}^2)$*, Amer. J. Math., 121 (1999), pp. 587–628.
[12] R. A. DeVore, *A note on adaptive approximation*, Approx. Theory Appl., 3 (1987), pp. 74–78.
[13] R. A. DeVore and G. G. Lorentz, *Constructive Approximation*, Springer-Verlag, Berlin, 1993.
[14] R. A. DeVore and V. A. Popov, *Free multivariate splines*, Constr. Approx., 3 (1987), pp. 239–248.

[15] R. A. DeVore and V. A. Popov, *Interpolation spaces and non-linear approximation*, in Function Spaces and Applications, M. Cwikel, J. Peetre, Y. Sagher, and H. Wallin, eds., Lecture Notes in Math. 1302, Springer, Berlin, 1988, pp. 191–205.
[16] R. A. DeVore and V. A. Popov, *Interpolation of Besov spaces*, Trans. Amer. Math. Soc., 305 (1988), pp. 397–414.
[17] R. A. DeVore and X. M. Yu, *K-functional for Besov spaces*, J. Approx. Theory, 67 (1991), pp. 38–50.
[18] R. A. DeVore and X. M. Yu, *Degree of adaptive approximation*, Math. Comp., 55 (1990), pp. 625–635.
[19] Y.-K. Hu, *Convexity preserving approximation by free knot splines*, SIAM J. Math. Anal., 22 (1991), pp. 1183–1191.
[20] Y.-K. Hu, *An algorithm for data reduction using splines with free knots*, IMA J. Numer. Anal., 13 (1993), pp. 365–381.
[21] Y.-K. Hu, K. A. Kopotun, and X. M. Yu, *On multivariate adaptive approximation*, Constr. Approx., 16 (2000), pp. 449–474.
[22] T. Lyche and K. Mørken, *A data reduction strategy for splines with applications to the approximation of functions and data*, IMA J. Numer. Anal., 8 (1988), pp. 185–208.
[23] P. Petrushev, *Direct and converse theorems for spline and rational approximation and Besov spaces*, in Functions Spaces and Approximation, M. Cwikel, J. Peetre, Y. Sagher, and H. Wallin, eds., Lecture Notes in Math. 1302, Springer, Berlin, 1988, pp. 363–377.
[24] P. P. Petrushev and V. A. Popov, *Rational Approximation of Real Functions*, Cambridge Univeristy Press, Cambridge, UK, 1987.
[25] J. R. Rice, *Adaptive approximation*, J. Approx. Theory, 16 (1976), pp. 329–337.
[26] L. L. Schumaker, *Spline Functions: Basic Theory*, John Wiley, New York, 1981.