Adaptive Approximation of Curves *

Peter Binev, Wolfgang Dahmen, Ronald DeVore, Nira Dyn

July 22, 2004

Abstract

We propose adaptive multiscale refinement algorithms for approximating and encoding curves by polygonal curves. We establish rates of approximation of these algorithms in the Hausdorff metric. For example, we show that under the mere assumption that the original curve has finite length then the first of these algorithms gives a rate of convergence $O(1/n)$ where n is the number of vertices of the approximating polygonal curve. Similar results giving second order approximation are proven under weak assumptions on the curvature such as L_p integrability, $p > 1$. Note that for nonadaptive algorithms, to obtain the same order of approximation would require that the curvature is bounded.

Key Words: Polygonal approximation of planar curves, adaptive multiscale refinement, maximal functions, convergence rates, encoding

AMS Subject Classification: 53A04, 41A25, 65D05, 65D17

1 Introduction

In this paper, we shall consider certain problems concerning the approximation and encoding of curves in $\mathbb{R}^3$ that arise in compression of terrain surfaces. One strategy for compressing and encoding such surfaces (see e.g. [7]) begins by extracting a finite number of curves that represent the given surface well. These are typically level curves, ridge curves and drainage curves. They can be viewed as giving a wire frame for the surface. In an encoding algorithm, these curves would be approximated, e.g. by piecewise linear functions, to get an approximate wire frame for the original surface. Extension techniques can then be used to generate a surface which interpolates the approximation curves. This new surface, which is viewed as an approximation to the original surface can be encoded by simply encoding the approximating curves since the extension part of the algorithm does not require any encoding. This type of encoding strategy can also be made progressive by prioritizing the original curves in terms of importance and sending bits to describe the highest priority curves first and then bits to describe the next curves, in order

*This work has been supported by the Office of Naval Research Contract Nr. N0014-03-10051, the Army Research Office Contract Nr. DAAD 19-02-1-0028, the National Science Foundation Grants DMS 0221642, DMS 9872890, the Deutsche Forschungsgemeinschaft grant SFB 401, the European Community's Human Potential Programme under Contract HPRN-CT-2002-00286, "Breaking Complexity".

of priority, as well as update bits to improve the accuracy of the approximation of the previous curves.The implementation of this encoding strategy rests on two issues: (i) how to extract a suitable network of curves, (ii) how to efficiently approximate and encode these curves. The purpose of the present paper is to discuss algorithms for the second of these problems. We shall restrict ourselves to planar curves, for example the level curves of the original surface. We shall always assume that these curves are continuous. Thus, we shall be interested in the approximation of a given continuous planar curve γ by polygonal curves. We consider the curve γ to be parametrized with respect to a parameter $t \in \Delta$, where Δ is some finite interval. For specificity, we shall use the arclength parametrization and shall denote the arclength parameter by s. Thus, $\Delta = [0, \ell(\gamma)]$, where $\ell(\gamma)$ is the length of γ.The algorithms we propose are motivated by multiscale or wavelet decompositions which have proven so effective in image encoding. We shall propose two algorithms. Of these, the first is more of theoretical interest and the second, which is easier to use in encoding, is currently used in our surface encoding software.For the remainder of this paper, let γ denote a planar curve in $\mathbb{R}^2$ of finite length. In the case that γ has a finite number of self intersections, one can consider it as a union of curves which have no self intersections, and work with them separately. So from now on we shall assume in addition that γ does not intersect itself.The algorithms begin with a fixed polygonal curve λ_0 which interpolates γ. The choice of this first polygon is arbitrary and can be defined by selecting three points arbitrarily from the curve and then interpolating with vertices of a triangle. In practical encoding, a good choice for this interpolatory polygon is crucial. For example, if the original curve has a finite number of point singularities (e.g. cusps), then choosing these points for the initial interpolation is effective. This will separate the curve into arcs with no singularities. However, we shall not consider this issue further here and the theorems we prove about our algorithms are impervious to the effectiveness of this initial polygonal approximation.The algorithms proceed to adaptively create new polygons by adding new interpolation points between the existing ones. This can be considered also as a refinement process. Given two points A and B on γ we denote by $I = AB = [A, B]$ the linear segment that connects A and B, and by $\gamma_I = \gamma_{AB}$ the arc of γ between A and B. More generally, if we have a set $\Lambda = \{Q_0, Q_1, ..., Q_n\}$ of consecutive points on γ, we denote by λ_Λ the polygonal curve that interpolates the points Q_k, $k = 0, \ldots, n$, and is linear in between each pair of points (Q_{k-1}, Q_k). Thus λ_Λ is composed of linear segments $I_k := [Q_{k-1}, Q_k]$ (which we call *intervals*). Going further in the paper, we shall mostly use intervals instead of points to describe a partition, and in particular, shall use the same notation $\Lambda = \{I_k\}_{k=1}^n$ to denote the set of intervals for the partition Λ. Given an interval $I = AB$, the algorithms that we shall analyze give a specific rule for generating a new point R on γ_I. We call this rule the *refinement rule*. Applying the refinement rule we obtain two new segments $I' = AR$ and $I'' = RB$ that correspond to two subarcs of I, namely $\gamma_{I'}$ and $\gamma_{I''}$. We call I' and I'' *children* of I and denote by $\mathcal{C}(I) = \{I', I''\}$ the set of these two children. Then $I = \mathcal{P}(I') = \mathcal{P}(I'')$ is called the *parent* of I' and I''. Starting from some initial partition Λ_0 and applying the refinement rule successively generates an infinite binary tree T^* called the *master tree* associated to the algorithm. The nodes $I \in \Lambda_0$ are the root nodes of T^*. By a *tree* we shall mean a finite subtree of T^* that contains all the root nodes of T^* and in addition whenever $I \in T$ is not a root node then the parent of I and the sibling of I must be in T. We shall work separately with each of the arcs in which the curve γ is subdivided by the initial interpolation points from Λ_0.

To make the notation simpler, we shall assume that γ is one of these arcs, and hence λ_0 will be the linear segment connecting the endpoints of γ which is the only element of the initial partition Λ_0. The algorithms that we shall analyze generate new partitions by adaptively choosing which intervals to subdivide based on certain error criteria. Each such adaptively generated partition can be identified with a tree T which records the refinement made in creating the partition. The leaves of the tree are the intervals in the final partition. If we add a point R to the current list of interpolating points, it subdivides some interval $I \in \Lambda$ and thereby creates two new intervals I' and I'' and a new partition $\Lambda^+ = (\Lambda \backslash \{I\}) \cup \{I', I''\}$. The tree $T^+ = \mathcal{T}(\Lambda^+)$ is then obtained from T by adding two branches from the node I to its children I' and I''.We can associate a multiresolution analysis to the refinement rule. Namely starting with the initial partitions Λ_0 we do one refinement and thereby obtain Λ_1. In general, we subdivide every arc corresponding to Λ_j by using our refinement rule. Thus, Λ_{j+1} is the collection of all children of the intervals $I \in \Lambda_j$. Thus, $\Lambda_j = \{I_{j,k}\}_{k=1}^{2^j}$ where the $I_{j,k}$ are the intervals for the j-th partition. Thus, Λ_j can be considered as the j-th level of the full binary tree T^* with root node $I_{0,1}$.Our algorithms for generating finite subtrees of T^* are adaptive. Given an $\varepsilon > 0$ and an error indicator $\mathcal{E}(I)$ defined for every $I \in T^*$, the algorithms start initially with $\Lambda = \Lambda_0$ and adaptively generate new partitions Λ as follows:

- Subdivide the interval I which has the largest $\mathcal{E}(I)$ by using the refinement rule. In case there are several intervals with the largest $\mathcal{E}(I)$ choose one in an arbitrary way to refine.
- Redefine $\Lambda = (\Lambda \backslash \{I\}) \cup \{I', I''\}$ and calculate $\mathcal{E}$ for the new intervals I' and I''.
- Stop, if $\mathcal{E}(I) \leq \varepsilon$ for all $I \in \Lambda$.

An alternative algorithm would begin with a budget N of the number of intervals allowed in the final partition and then stop when the number of intervals in Λ is N. This algorithm can be further improved using the techniques described in [2] but we shall not discuss that issue here.We shall measure the error of approximation in terms of the Hausdorff metric which matches well the intended application to surface encoding. As usual the (one sided) Hausdorff distance between the two curves γ and λ is defined by

$$\mathcal{H}(\gamma, \lambda) := \sup_{Q \in \gamma} \operatorname{dist}(Q, \lambda) = \sup_{Q \in \gamma} \inf_{R \in \lambda} |QR| \,, \tag{1.1}$$

where $|QR| = \ell(QR)$ is the length of QR. The *Hausdorff distance* between γ and Λ is then defined as

$$d_{\mathcal{H}}(\gamma, \lambda) = \max(\mathcal{H}(\gamma, \lambda), \mathcal{H}(\lambda, \gamma)) \tag{1.2}$$

In this paper we shall consider two basic algorithms. Both of them are using the same error indicator, namely,

$$\mathcal{E}(I) = \mathcal{H}(\gamma_I, \lambda_I). \tag{1.3}$$

The distinction in the two algorithms lies in the refinement rule.

Algorithm A: *In this algorithm, given I, we subdivide the arc γ_I into two arcs with equal length, i.e. the two children I', I'' must satisfy $\ell(\gamma_{I'}) = \ell(\gamma_{I''})$.*

Algorithm B: *In this algorithm, given I, we choose a point P from γ_I which is equally distanced from the ends of the interval I. This point is on the perpendicular bisector of I. If there is more than one such point then we choose the one whose distance from the endpoints is largest. Notice that for the two children I', I'' of I we have $\ell(\lambda_{I'}) = \ell(\lambda_{I''})$.*

As we shall see, the estimates for **Algorithm A** require a little bit less restrictive conditions on the smoothness of the curve γ.

2 Approximation of a single arc

Given two points A, B from γ, we shall study in this section how well the linear segment $I = AB$ approximates the arc γ_I. Let us begin by making some observations about approximating an arbitrary point P from $\mathbb{R}^2$ by the line segment $I = AB$. Let $Q = Q(P) \in I$ be the best approximation of P from I, i.e. $Q(P)$ is the projection of P onto I. If Q is in the interior of I then PQ is perpendicular to I. The mapping $P \to Q(P)$ is a continuous function of P.We use these observations to note that

$$\mathcal{H}(\lambda_I, \gamma_I) \leq \mathcal{H}(\gamma_I, \lambda_I). \tag{2.4}$$

Indeed, consider the projection of an arbitrary point P from γ_I onto I. Since, by assumption, γ is continuous, this projection is a continuous mapping. Since both A and B are images of this projection, it follows that every point Q from I is an image of this projection. Hence, given any point Q from I, there is a point P which projects onto it. We have that $|QP|$ does not exceed the right side of (2.4) and (2.4) follows. To describe our first bound for $\mathcal{H}(\gamma_I, \lambda_I)$, we introduce a certain second order modulus of smoothness for curves. For any two points $Q, R \in \gamma$ we define a finite difference of order two as twice the maximal distance between the segment QR and any point on γ_{QR} which lies on the symmetry line with respect to Q and R:

$$\Delta^2(\gamma, Q, R) := \max\left\{2 \text{ dist}\left(P, \frac{Q+R}{2}\right) : P \in \gamma_{QR}, |PQ| = |PR|\right\} . \tag{2.5}$$

Let us mention that $\sqrt{2(|PQ|^2 + |PR|^2) - |QR|^2} = 2\text{dist}\left(P, \frac{Q+R}{2}\right)$. The reason we put the factor 2 in front of the distance is to be analogous to the definition of finite differences of functions. Using the quantity from (2.5) we define a modulus of smoothness of order two for γ

$$\omega_2(\gamma; A, B) = \omega_2(\gamma_{AB}) := \sup_{P \in \gamma_{AB}} \max\{\Delta^2(\gamma, A, P), \Delta^2(\gamma, P, B)\} . \tag{2.6}$$

The following lemma is an analogue for curves of the Whitney theorem [8] for approximation by linear functions.

Lemma 1 *Given a continuous curve γ_{AB} of finite length with end points A and B, the following estimate for the approximation of γ_{AB} by the linear segment AB holds*

$$\mathcal{H}(\gamma_{AB}, AB) \leq \omega_2(\gamma_{AB}) . \tag{2.7}$$

Proof. Since γ_{AB} is a compact set in $\mathbb{R}^2$, the left side of (2.7) is finite and the supremum from (1.1) is realized at some point $Q^* \in \gamma_{AB}$. Let $E = \mathcal{H}(\gamma_{AB}, AB)$. Then $\text{dist}(Q^*, AB) = E$. Without loss of generality we can assume that $|AQ^*| \leq |BQ^*|$. Then the arc γ_{Q^*B} crosses the circle with center Q^* and radius $|AQ^*|$ at some point P. We denote by R the midpoint of the linear segment AP and observe that by the definitions (2.5) and (2.6)

$$|Q^*R| \leq \frac{1}{2}\Delta^2(\gamma, A, P) \leq \frac{1}{2}\omega_2(\gamma; A, B) \; . \tag{2.8}$$

Let $P_0 \in AB$ be the point for which $|PP_0| = \text{dist}(P, AB)$ and let $R_0 \in AB$ be the midpoint of AP_0. Then

$$\text{dist}(R, AB) \leq |RR_0| \leq \frac{1}{2}|PP_0| \leq \frac{1}{2}E \; . \tag{2.9}$$

Using the triangle inequality, (2.8), and (2.9) we receive

$$E = \text{dist}(Q^*, AB) \leq |Q^*R| + \text{dist}(R, AB) \leq \frac{1}{2}\omega_2(\gamma; A, B) + \frac{1}{2}E \; . \tag{2.10}$$

This proves $E \leq \omega_2(\gamma; A, B)$. □ Next, we consider another estimate of $\mathcal{H}(\gamma_{AB}, AB)$ in terms of a certain first order modulus of continuity of the tangent vector to γ. We define

$$\omega_{\measuredangle}(\gamma_{AB}) = \omega_{\measuredangle}(\gamma; A, B) := \max(\sup_{P \in \gamma_{AB}} \measuredangle PAB, \sup_{P \in \gamma_{AB}} \measuredangle PBA). \tag{2.11}$$

Lemma 2 *Let γ be any curve of finite length. Then*

$$\mathcal{H}(\gamma_{AB}, AB) \leq \frac{1}{2} \; \ell(\gamma_{AB}) \; \omega_{\measuredangle}(\gamma_{AB}) \; . \tag{2.12}$$

Proof. Let $P \neq A, B$ be any point on the curve segment γ_{AB}. We want to show that $\text{dist}(P, AB)$ can be bounded by the right side of (2.12). We can assume that $\omega_{\measuredangle}(\gamma_{AB}) \leq 1 \leq \pi/2$ since otherwise (2.12) follows trivially since the distance from P to AB is always $\leq \ell(\gamma_{AB})/2$. We can also assume that $|AP| \leq \ell(\gamma_{AB})/2$ since otherwise we reverse the roles of A an d B in the following derivation. It follows from these assumptions that P projects onto a point Q in the interior of AB and

$$|PQ| \leq |\sin\alpha||AP| \leq |\sin\alpha|\ell(\gamma_{AB})/2, \tag{2.13}$$

where $\alpha = \measuredangle PAB$ is the angle made by the vectors AP and AB. Since $|\sin\alpha| \leq |\alpha| \leq \omega_{\measuredangle}(\gamma_{AB})$, (2.12) follows. □ We are next going to derive a bound for Hausdorff distance in terms of curvature.

Lemma 3 *If γ has finite curvature at almost every point, then*

$$\mathcal{H}(\gamma_{AB}, AB) \leq \frac{\ell(\gamma_{AB})}{2} \int_{\{s:\gamma(s)\in\gamma_{AB}\}} \kappa(s)ds \; . \tag{2.14}$$

Proof. In view of Lemma 2, it is enough to show that for any points A, B on γ, we have

$$\omega_{\measuredangle}(\gamma_{AB}) \leq \int_{\{s:\gamma(s)\in\gamma_{AB}\}} \kappa(s)ds \; . \tag{2.15}$$

Let P be any point in γ_{AB}. The total angular change of the tangent in traversing the curve between A and B is larger than $\measuredangle PAB + \measuredangle PBA$ and the (2.15) follows. □

3 Main Results

In this section we shall give sufficient conditions on the curve γ such that the application of our algorithms will yield a polygonal curve with n segments which approximates γ in the Hausdorff metric to accuracy Cn^{-r}. Here r can be any positive real number. The condition we impose on the curve is given in terms of certain maximal functions. The reader unfamiliar with maximal functions and their use in analyzing adaptive algorithms should look at the simple paper [4] for orientation. The two most interesting cases of our theorems are $r = 1$ and $r = 2$ since then the conditions we impose on γ relate simply to arc length or curvature. The theorems actually hold for $r > 2$ but then the spaces are very esoteric and very thin in content. In this section, we assume that the curve γ is parametrized with respect to the arclength parameter s. We recall our notation from the last section. For each I in the master tree T^*, we have the arc γ_I and the linear segment λ_I connecting the endpoints of γ_I. The local error $\mathcal{E}(I)$ is defined by

$$\mathcal{E}(I) := \mathcal{H}(\gamma_I, \lambda_I). \tag{3.16}$$

Let $\Lambda = \Lambda(\varepsilon)$ be the partition obtained by the adaptive algorithm (i.e. by either **Algorithm A** or **Algorithm B**) for a given choice of ε. We recall that the algorithm terminates when $\mathcal{E}(I) \leq \epsilon$ for all $I \in \Lambda$. The approximation λ_Λ of the curve γ is defined as the union of the linear segments λ_I for $I \in \Lambda$. It follows that

$$d_{\mathcal{H}}(\gamma, \lambda_\Lambda) \leq \max_{I\in\Lambda} d_{\mathcal{H}}(\gamma_I, \lambda_I) \leq \max_{I\in\Lambda} \mathcal{H}(\gamma_I, \Lambda_I) \leq \varepsilon, \tag{3.17}$$

The efficiency of the algorithm is determined by the size of $\Lambda(\varepsilon)$ as a function of ε. In this section, we shall show how we can bound the size of $\Lambda(\varepsilon)$ through certain maximal functions. The idea goes back to [4] where the Hardy-Littlewood maximal function was used in bounding the error in adaptive algorithms for approximating functions. For the analysis of **Algorithm A**, we shall use the maximal function

$$\mathcal{M}_r(s) = \mathcal{M}_r(\gamma; s) := \sup_{\gamma(s)\in\gamma_{QR}} \frac{\mathcal{E}(QR)}{[\ell(\gamma_{QR})]^r}. \tag{3.18}$$

In the case of **Algorithm B** we shall use

$$\overline{\mathcal{M}}_r(s) = \overline{\mathcal{M}}_r(\gamma; s) := \sup_{\gamma(s)\in\gamma_{QR}} \frac{\mathcal{E}(QR)}{|QR|^r}. \tag{3.19}$$

Theorem 1 *Let $\varepsilon > 0$ be given and let $\Lambda := \Lambda(\varepsilon)$ be obtained by applying our algorithm for this choice of ε. Then,*

$$d_{\mathcal{H}}(\gamma, \lambda_\Lambda) \leq \varepsilon. \tag{3.20}$$

(i) In the case of **Algorithm A**, *whenever $\mathcal{M}_r(\gamma, s) \in L_p$, for $p = 1/r$ and some $r > 0$, then*

$$\#(\Lambda(\varepsilon)) \leq 2\varepsilon^{-1/r} \|\mathcal{M}_r\|_{L_p}^{1/r}. \tag{3.21}$$

(ii) In the case of **Algorithm B**, *whenever $\overline{\mathcal{M}}_r(\gamma, s) \in L_p$, for $p = 1/r$ and some $r > 0$, then*

$$\#(\Lambda(\varepsilon)) \leq 2\varepsilon^{-1/r} \|\overline{\mathcal{M}}_r\|_{L_p}^{1/r}. \tag{3.22}$$

Proof. We have already observed that (3.20) holds. To prove (i), we note that whenever $I \in \Lambda(\varepsilon)$, we have that the parent I^* of I satisfies $\mathcal{E}(I^*) > \varepsilon$ and therefore

$$\begin{aligned}
\varepsilon &< \ell(\gamma_{I^*})^r \frac{\mathcal{E}(I^*)}{\ell(\gamma_{I^*})^r} \leq \ell(\gamma_{I^*})^r \left[\inf_{\{s:\gamma(s)\in\gamma_{I^*}\}} \mathcal{M}_r(s)^p\right]^r \\
&\leq \left[\frac{\ell(\gamma_{I^*})}{\ell(\gamma_I)}\right]^r \left[\int_{\{s:\gamma(s)\in\gamma_I\}} \mathcal{M}_r(s)^p\, ds\right]^{1/p} \\
&= 2^r \left[\int_{\{s:\gamma(s)\in\gamma_I\}} \mathcal{M}_r(s)^p\, ds\right]^{1/p}. \qquad (3.23)
\end{aligned}$$

If we raise both sides of (3.23) to the power $p = 1/r$ and then sum over $I \in \Lambda(\varepsilon)$, we arrive at (3.21) because the arcs γ_I are disjoint.The proof of (ii) is similar. Now, we have

$$\begin{aligned}
\varepsilon &< |I^*|^r \frac{\mathcal{E}(I^*)}{|I^*|^r} \leq |I^*|^r \left[\inf_{\{s:\gamma(s)\in\gamma_{I^*}\}} \overline{\mathcal{M}}_r(s)^p\right]^r \\
&\leq \left[\frac{|I^*|}{\ell(\gamma_I)}\right]^r \left[\int_{\{s:\gamma(s)\in\gamma_I\}} \overline{\mathcal{M}}_r(s)^p\, ds\right]^{1/p} \\
&\leq \left[\frac{|I^*|}{|I|}\right]^r \left[\frac{|I|}{\ell(\gamma_I)}\right]^r \left[\int_{\{s:\gamma(s)\in\gamma_I\}} \overline{\mathcal{M}}_r(s)^p\, ds\right]^{1/p} \\
&\leq 2^r \left[\int_{\{s:\gamma(s)\in\gamma_I\}} \overline{\mathcal{M}}_r(s)^p\, ds\right]^{1/p}. \qquad (3.24)
\end{aligned}$$

We complete the proof as before. □ The remainder of this section will be used to give sufficient conditions which will guarantee that the maximal function $\mathcal{M}_r$ or $\overline{\mathcal{M}}_r$ is in L_p, $p = 1/r$. We begin with the following case which corresponds to a first order approximation.

Corollary 1 *Let γ be a curve with finite length $\ell(\gamma)$ and let $\varepsilon > 0$.*
(i) The polygon λ_Λ, $\Lambda := \Lambda(\varepsilon)$ obtained by applying **Algorithm A** *for this ε has the following approximation property*

$$d_{\mathcal{H}}(\gamma, \lambda_\Lambda) \leq \varepsilon \quad \textit{and} \quad \#(\Lambda(\varepsilon)) \leq \frac{\ell(\gamma)}{\varepsilon}. \qquad (3.25)$$

(ii) Define $\tilde{\mathcal{M}}(s) := \sup_{Q,R\in\gamma\,:\,\gamma(s)\in\gamma_{QR}} \frac{\ell(\gamma_{QR})}{|QR|}$ *and assume that $\tilde{\mathcal{M}} \in L_1(\gamma)$. Then, the polygon λ_Λ, $\Lambda = \Lambda(\varepsilon)$ obtained from* **Algorithm B** *for this ε satisfies*

$$d_{\mathcal{H}}(\gamma, \lambda_\Lambda) \leq \varepsilon, \quad \#(\Lambda(\varepsilon)) \leq \frac{\|\tilde{\mathcal{M}}\|_{L_1(\gamma)}}{\varepsilon}. \qquad (3.26)$$

Proof. It is enough to apply Theorem 1 with $r = 1$ and $p = 1$. In the case of (i), we have $\mathcal{E}(I) \leq \ell(\lambda_I)/2$ and so $\mathcal{M}_1(s) \leq 1/2$ and therefore the corollary follows. In the case of (ii), we have $\overline{\mathcal{M}}_1(s) \leq \tilde{\mathcal{M}}(s)/2$ and the corollary again follows. □ To ensure an order of approximation $\leq C/n$ using a polygonal curve with n pieces, Corollary

1 says that it is enough to have γ of finite length in the case of **Algorithm A**. In this case C is just the length of γ. For **Algorithm B** we need to assume slightly more, namely that $\tilde{\mathcal{M}}$ is in L_1. In the analogous case (see [6]) of approximating functions g rather than curves, the difference is between conditions like $g \in L_1$ or $\mathcal{M}^*(g) \in L_1$ where $\mathcal{M}^*$ is the Hardy-Littlewood maximal operator

$$M^*(g)(x) := \sup_{I \ni x} \frac{1}{|I|} \int_I |f(u)|\, du \tag{3.27}$$

where the sup is taken over all intervals I containing x. The condition $M^*(g) \in L_1$ is equivalent to $g \in L \log L$ and is therefore only slightly stronger than $g \in L_1$.Our next example derives a better order of approximation in terms of curvature. Our starting point is the estimate (2.14) which gives

$$\mathcal{E}(I) \leq \frac{\ell(\gamma_I)}{2} \int_{\{s:\gamma(s)\in\gamma_I\}} \kappa(s)\, ds. \tag{3.28}$$

Using this estimate for $\mathcal{E}$, we obtain that

$$\mathcal{M}_2(s) \leq \frac{1}{2\ell(\gamma_I)} \int_{\{s':\gamma(s')\in\gamma_I\}} \kappa(s')\, ds' \leq \frac{1}{2}\mathcal{M}^*(\kappa)(s), \quad \gamma(s) \in \gamma_I. \tag{3.29}$$

Corollary 2 *Let $\varepsilon > 0$.*
(i) Let κ be the curvature of γ. If $\kappa \in L \log L$ then the polygon λ_Λ, $\Lambda := \Lambda(\varepsilon)$ obtained by applying **Algorithm A** *for this ε has the following approximation property*

$$d_{\mathcal{H}}(\gamma, \lambda_\Lambda) \leq \varepsilon \quad \textit{and} \quad \#(\Lambda(\varepsilon)) \leq \left[\frac{C_0 \ell(\gamma) \|\kappa\|^{1/2}_{L\log L}}{\varepsilon}\right]^{1/2} \tag{3.30}$$

with C_0 an absolute constant.
(ii) If in addition γ is in $Lip\,1$, then, the polygon λ_Λ, $\Lambda = \Lambda(\varepsilon)$ obtained from **Algorithm B** *for this ε satisfies*

$$d_{\mathcal{H}}(\gamma, \lambda_\Lambda) \leq \varepsilon, \quad \#(\Lambda(\varepsilon)) \leq \left[\frac{C_0 \ell(\gamma) \|\gamma\|^2_{Lip\,1} \|\kappa\|^{1/2}_{L\log L}}{\varepsilon}\right]^{1/2}. \tag{3.31}$$

Proof. We apply Theorem 1 with $r = 2$, $p = 1/2$. In the case of (i), we have from (3.29) and the application of Theorem 1 that

$$\begin{aligned} \#(\Lambda(\epsilon)) &\leq \sqrt{2}\frac{\|\mathcal{M}^*(\kappa)\|^{1/2}_{L_{1/2}}}{\varepsilon^{1/2}} \leq \frac{\sqrt{2\ell(\gamma)}\|\mathcal{M}^*(\kappa)\|^{1/2}_{L_1}}{\varepsilon^{1/2}} \\ &\leq \frac{\sqrt{C_0\ell(\gamma)}\|\kappa\|^{1/2}_{L\log L}}{\varepsilon^{1/2}}. \end{aligned} \tag{3.32}$$

where the last result is a classical inequality about the Hardy-Littlewood maximal operator (see [1], p. 250, Theorem 6.7).In the case of (ii), we use our assumption that γ is in $Lip\,1$ to replace $|I|$ by $\|\gamma\|^{-1}_{Lip\,1}\ell(\gamma_I)$ in the definition of $\overline{\mathcal{M}}_2$. This shows that $\overline{\mathcal{M}}_2(\kappa)(s) \leq \|\gamma\|^2_{Lip\,1}\mathcal{M}_2(s)$ which reduces us to the case already proven. □ If we take $\varepsilon = 1/n^2$,

we see that these algorithms give an order of approximation C/n^2 with polygons having at most n pieces. The assumption in the case of **Algorithm A** is that $\kappa \in L \log L$ which holds for example whenever $\kappa \in L_p$ with $p > 1$. In the case of **Algorithm B**, we have the same assumption on κ but the additional assumption on γ. Note that this additional assumption in and of itself would only give approximation order $O(1/n)$. If we compare either of these two results with nonadaptive processes we see that the requirement in this corollary are much less demanding. Namely, for a nonadaptive process we would require that γ have bounded curvature whereas here we are requiring only a certain integrability of κ.For our last example, we consider the case of general r with $0 < r \leq 2$ and we use the second order modulus of smoothness to bound $\mathcal{E}(I)$ as amplified in the estimate in Lemma 1. We discuss only the case of **Algoritihm B**. By Lemma 1, we can bound the maximal function $\overline{\mathcal{M}}_r$ by

$$\overline{\mathcal{M}}_r(\gamma, s) \leq \tilde{\mathcal{M}}_r(\gamma, s) := \sup_{\{I:\gamma(s)\in\gamma_I\}} \frac{\omega_2(f, \gamma_I)}{|I|^r}. \tag{3.33}$$

Therefore, we have as an immediate consequence of Theorem 1

Corollary 3 *Let $\varepsilon > 0$, $0 < r \leq 2$ and $p = 1/r$. Whenever, $\tilde{\mathcal{M}}_r(\gamma, s)$ is in L_p, then the polygon λ_Λ output of* **Algorithm B** *for this ε satisfies*

$$d_{\mathcal{H}}(\gamma, \lambda_\Lambda) \leq \varepsilon \quad \textit{and} \quad \#(\Lambda(\varepsilon)) \leq 2\varepsilon^{-1/r} \|\tilde{\mathcal{M}}_r(\gamma, s)\|_{L_p}^r. \tag{3.34}$$

One can view the maximal function $\tilde{\mathcal{M}}_r(\gamma, s)$ as akin to a maximal function of a fractional derivative of order r of γ. In the case of functions, this viewpoint is amplified upon in [6] and its connections to Besov regularity is spelled out there. Note that the regularity condition imposed on γ in this corollary is much weaker than the regularity that would be needed in a nonadaptive algorithm which would require the curve to be in $Lip\, r$.

4 Encoding planar curves

We have begun this paper by motivating our algorithms for the purpose of encoding. We return to the topic of encoding in this section. We begin by giving a simple encoding of polygonal curves with n vertices. This encoding will replace the original polygonal curve λ by a quantized polygonal curve $\bar{\lambda}$ which can be represented with a finite bitstream. We first show that we can do such an encoding so as to preserve the original approximation accuracy to γ while using $O(n \log n)$ bits[1]. Later in this section we shall briefly describe a more sophisticated encoding scheme, along the lines of the tree encoding algorithm of [3] for images that allows a progressive encoding for **Algorithm B** and would have improved accuracy. For example, it would allow us to remove the $\log n$ factor in many settings. However, we shall not give a rigorous analysis of the properties of this encoding here. To start our discussion, we shall assume, for notational convenience only, that all of the points P on the curve γ satisfy $|P| < 1$, i.e. the Euclidean distance of P to the origin is bounded by 1. A simple modification would allow us to replace 1 by any fixed positive

[1]Here and later log always denotes logarithms to the base 2.

constant M. We can write any real number x with $|x| < 1$ in its binary expansion

$$x = (-1)^{b_0} \sum_{i=1}^{\infty} b_i 2^{-i}, \quad b_i = b_i(x) \in \{0, 1\}, \ i = 0, 1, \ldots. \tag{4.35}$$

For any integer $k > 0$, we define

$$B_k(x) := (b_0(x), \ldots, b_k(x)), \quad Q_k(x) = (-1)^{b_0(x)} \sum_{i=1}^{k} b_i(x) 2^{-i}. \tag{4.36}$$

It follows that

$$|x - Q_k(x)| \leq 2^{-k}, \quad k = 1, 2, \ldots. \tag{4.37}$$

If λ is a polygonal curve with n vertices lying on γ, we can encode λ with the bitstream

$$B(\lambda) := B_0(\lambda), B_1(\lambda), \ldots, B_n(\lambda). \tag{4.38}$$

which is a concatenation of the bitstreams $B_0, B_1, \ldots, B_n$. The bitstream B_0 encodes n as follows. If $b_1, \ldots, b_{\lceil \log n \rceil}$ are the binary bits of n, then B_0 is $b_1, b_1, b_2, b_2, \ldots, b_{\lceil \log n \rceil}, b_{\lceil \log n \rceil}, 0, 1$. The purpose of the repetition of bits is so that the receiver knows when $0, 1$ appears that it signifies the end of the bitstream for determining n. The bitstreams $B_1, \ldots, B_n$ each correspond to the vertices $P_1, \ldots, P_n$ on the curve λ which appear in their natural order. Thus $B_k = B_m(x_k), B_m(y_k)$ where $P_k = (x_k, y_k)$. Since the length of each of these bitstreams is known, the receiver knows when the bits of P_k end and those of the next vertex begin. After receiving the bitstream $B(\lambda)$, the receiver can construct the quantized polygon $\bar{\lambda}$ whose vertices are $\bar{P}_k = (Q_m(x_k), Q_m(y_k))$. In view of (4.37) we have

$$d_{\mathcal{H}}(\gamma, \bar{\gamma}) \leq 2^{-m+1/2} \leq \sqrt{2} n^{-2}. \tag{4.39}$$

Thus, if λ approximate γ to accuracy Cn^{-r}, $0 < r \leq 2$ (as was the case in our theorems which analyze the performance of **Algorithm A** and **Algorithm B**), then $\bar{\lambda}$ preserves this same accuracy with C replaced by $C + \sqrt{2}$. The total number of bits in $B(\lambda)$ is $(2n + 1)m + 2 \leq Cn \log n$. We now turn to giving a sketch of how a progressive encoding method for curves could be given based on our **Algorithm B**. This encoding is the analogue for curves of the image encoding algorithm given in [3]. Let us consider an interval $I = AB$ which occurs in the master tree T^* for this algorithm. We let w_I denote the unit vector orthogonal to AB pointing outward from AB. We recall that if I is refined then this corresponds to adding a new point P which is on the perpendicular bisector of I and also on γ. We can write

$$P = \frac{1}{2}(A + B) + d_I w_I \tag{4.40}$$

and

$$|d_I| \leq \mathcal{E}(I). \tag{4.41}$$

In going further, we shall assume, for notational convenience only, that $\mathcal{E}(I) < 1$ for all $I \in T^*$. Given λ, and the initial set V_0 of vertices, we shall associate a sequence of trees T_k, $k = 0, 1, \ldots$, with each T_k a growing of T_{k-1}. The tree T_0 consists of the root nodes

of the master tree T^*. To describe T_k, $k \geq 1$, let Λ_k be the set of all $I \in T^*$ for which $|d_I| > 2^{-k}$. Then T_k is defined to be the smallest subtree of T^* which contains Λ_k. We shall also use the notation $\Delta_k := T_k \setminus T_{k-1}$, $k \geq 1$. The set Δ_k of nodes tells how to grow T_{k-1} into T_k.Given any finite binary tree T, it can be encoded with at most $2\#(T)$ bits (see e.g. [3]). Given that T_{k-1} has already been encoded, we can encode T_k with $\leq 2\#(\Delta_k)$ bits. We denote the bitstream to do this encoding by B_k^t. We remark that it is shown in [3] that the encoding of these trees is done in such a way that the receiver knows when the encoding of the tree is complete. He also then knows the size of T_k and of Δ_k. We shall use the following notation. Given a set $\Lambda \subset T^*$, we denote by $B_j(\Lambda)$ the bitstream consisting of the j-th binary bit $b_j(d_I)$ of the d_I, $I \in \Lambda$ with the natural ordering[2] . Similarly, we denote by $B_j(V_0)$ the bitstream consisting of the j- th bit of the vertices in V_0 in the order they appear on the curve. There are two bits in $B_j(V_0)$ for each vertex in V_0 corresponding to the two coordinates.We can now describe our progressive encoding. The first bits we send identify the number n_0 of initial vertices. We do this in exactly the same way we encoded n in our first encoding algorithm. The next bits we send are $B_0(V_0)$ followed by $B_1(V_0)$. Since the receiver knows $n_0 = \#(V_0)$, he knows when each of the bitstreams end. Note that also, after receiving these bits the receiver has an approximation to the root nodes of T^*. He knows the number of these root nodes exactly but since he only knows an approximation to the initial vertices he can only find the root intervals approximately. The next bits we send are the $B_0(T_0)$ and $B_1(T_0)$ which gives the receiver the first approximation to the d_I, $I \in T_0$. We are now at the point where a general recursive structure generates the remainder of the bitstream. For each $k \geq 1$ we shall send in order the following bitstreams

$$B_k^t, B_0(\Delta_k), B_k(V_0), B_k(T_0), B_k(\Delta_1), \ldots, B_k(\Delta_k) \tag{4.42}$$

Let us make some observations about what this portion of the bitstream gives. First it tells us how to grow the tree T_{k-1} into the tree T_k. Secondly, this bitstream together with the previous bitstreams allow the receiver to construct an approximation to each of the initial vertices and the d_I for I in $T_0 \cup \cdots \cup T_k$ to accuracy 2^{-k}. Note that we do not need the bits $b_0(d_I), \ldots, b_{k-1}(d_I)$, for $I \in \Delta_k$ because by the definition of this set $|d_I| < 2^{-k+1}$ so these bits are zero.

References

[1] C. Bennett and R. Sharpley, Interpolation of Operators, Academic Press, N. Y., 1988.

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

[3] A. Cohen, W. Dahmen, I. Daubechies, and R. DeVore, Tree approximation and encoding, ACHA **11** (2001), 192–226.

[2]The natural order when transversing elements from a binary tree is first by depth in the tree and then from left to right at a given depth.

[4] R. DeVore, A note on adaptive approximation, *Approximation Theory and its Application* **3** (1987), 74–78.

[5] R. DeVore and G.G. Lorentz, Constructive Approximation, Springer, Grundlehren, vol. 303, 1993.

[6] R. DeVore and X. M. Yu, Degree of adaptive approximation, *Math. Comp* **55** (1988), 625–635.

[7] A. Solé, V. Caselles, G. Sapiro, and F. Arándiga, Morse description and geometric encoding of digital elevation maps, preprint.

[8] H. Whitney, On functions with bounded n-th differences, *J. Math. Pures Appl.* **36** (1957), 67–95.

Peter Binev

Department of Mathematics
University of South Carolina
Columbia, SC 29208
U.S.A.
E-mail: binev@math.sc.edu

Wolfgang Dahmen

Institut für Geometrie und Praktische Mathematik
RWTH Aachen
Templergraben 55
52056 Aachen
Germany
E-mail: dahmen@igpm.rwth-aachen.de

Ronald DeVore

Department of Mathematics
University of South Carolina
Columbia, SC 29208
U.S.A.
E-mail: devore@math.sc.edu

Nira Dyn

School of Mathematical Sciences
Tel Aviv University
Tel Aviv 69978
Israel
E-mail: niradyn@post.tau.ac.edu