Technische Universität Graz Institut für Mathematik

The Convergence Rate of the Sandwich Algorithm for Approximating Convex Functions

Günter Rote

Report 118 January 1992

Technische Universität Graz, Steyrergasse 30, A-8010 Graz, Austria

Version 2b

A slightly shortened version of this report appeared in *Computing* **48** (1992), 337–361.

This version is a little extended and shortened with respect to the printed revised version of October 1991.

Title page processed by TEX on February 10, 1995

The Convergence Rate of the Sandwich Algorithm for Approximating Convex Functions

Günter Rote [*]
Institut für Mathematik, Technische Universität Graz,
Steyrergasse 30, A-8010 Graz, Austria

Bericht Nr. 118, revidierte Fassung October 1991

Abstract — Zusammenfassung

The Convergence Rate of the Sandwich Algorithm for Approximating Convex Functions. The Sandwich algorithm approximates a convex function of one variable over an interval by evaluating the function and its derivative at a sequence of points. The connection of the obtained points is a piecewise linear upper approximation, and the tangents yield a piecewise linear lower approximation. Similarly, a planar convex figure can be approximated by convex polygons.

Different versions of the Sandwich algorithm use different rules for selecting the next evaluation point. We consider four natural rules (interval bisection, slope bisection, maximum error rule, and chord rule) and show that the global approximation error with n evaluation points decreases by the order of $O(1/n^2)$, which is optimal.

By special examples we show that the actual performance of the four rules can be very different from each other, and we report computational experiments which compare the performance of the rules for particular functions.

Die Konvergenzrate des Belegten-Brot-Algorithmus zur Approximation konvexer Funktionen. Der Belegte-Brot-Algorithmus approximiert eine konvexe Funktion einer Variablen über einem Intervall, indem er die Funktion und ihre Ableitung an einer Folge von Stützstellen ausrechnet. Die Verbindung der Punkte ergibt eine stückweise lineare obere Approximation, und die Tangenten liefern eine stückweise lineare untere Approximation. Auf ähnliche Art kann man einen konvexen Bereich der Ebene durch konvexe Polygone approximieren.

Verschiedene Versionen des Belegten-Brot-Algorithmus unterscheiden sich durch die Regel, nach der sie die nächste Stützstelle bestimmen. Wir zeigen für vier natürliche Regeln (Intervallhalbierung, Steigungshalbierung, maximaler-Fehler-Regel und Sehnenregel), daß der globale Approximationsfehler mit der Anzahl n der Stützstellen mit der bestmöglichen Ordnung $O(1/n^2)$ abnimmt.

Anhand von besonders konstruierten Beispielen zeigen wir, daß die vier Regeln sehr unterschiedliches Konvergenzverhalten haben können, und wir berichten über Rechenexperimente zum Vergleich der Regeln für einige ausgesuchte Funktionen.

AMS 1991 mathematics subject classification: 52-04, (41A15, 41A25, 52A10, 65D07, 68U05)

CR categories and subject descriptors (1987 version): F.2.1. [**Analysis of algorithms and problem complexity**]: Numerical algorithms and problems; G.1.2. [**Numerical analysis**]: Approximation — *spline and piecewise polynomial approximation*; I.3.5. [**Computer graphics**]: Computational geometry — *geometric algorithms*

General terms: algorithms, theory

Additional key words and phrases: piecewise linear convex approximation, convex polygonal approximation, Sandwich algorithm.

[*]Electronic mail: `rote@ftug.dnet.tu-graz.ada.at`. This work was partially supported by the Fonds zur Förderung der wissenschaftlichen Forschung, Project P7486-Phy.

1 Introduction

The Sandwich algorithm is an iterative procedure for approximating a convex function of one variable by piecewise linear convex functions. It starts by evaluating the function and its one-sided derivatives at the endpoints of the given interval. The line connecting the two endpoints of the graph of the function yields an initial upper bound of the function, and the two supporting lines described by the derivatives at the endpoints give an initial lower bound of the function (cf. figure 1). Now we select some point in the interval and evaluate the function and its derivative. (If the function is not differentiable, any subgradient will do.) In this way, we get an improved upper and lower approximation, and the problem is split into two subintervals. Now we select the subinterval in which the error is larger, and we partition it in the same way as above. We continue this process for a given number of iterations or until a specified error bound is met.

Since the convex function in which we are interested is enclosed from above and below by the two approximations, we call such a pair of piecewise linear approximating functions a *Sandwich approximation*, and an algorithm which produces a Sandwich approximation in the way described above is called a *Sandwich algorithm* (cf. Burkard, Hamacher, and Rote [1992], Martelli [1962]).

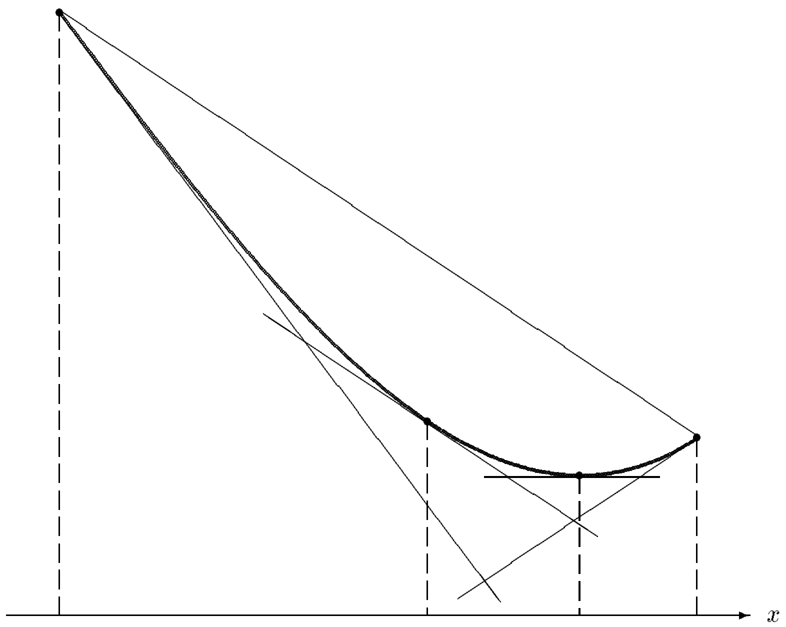


Figure 1: The Sandwich algorithm after two partitioning steps.

The error measure that we consider in this paper is the maximum vertical distance between the lower and the upper approximation. Other error measures, like the area between the two approximations or their maximum projective distance (normal distance, Hausdorff distance), can be handled by the Sandwich algorithm in the same way (see also section 5).

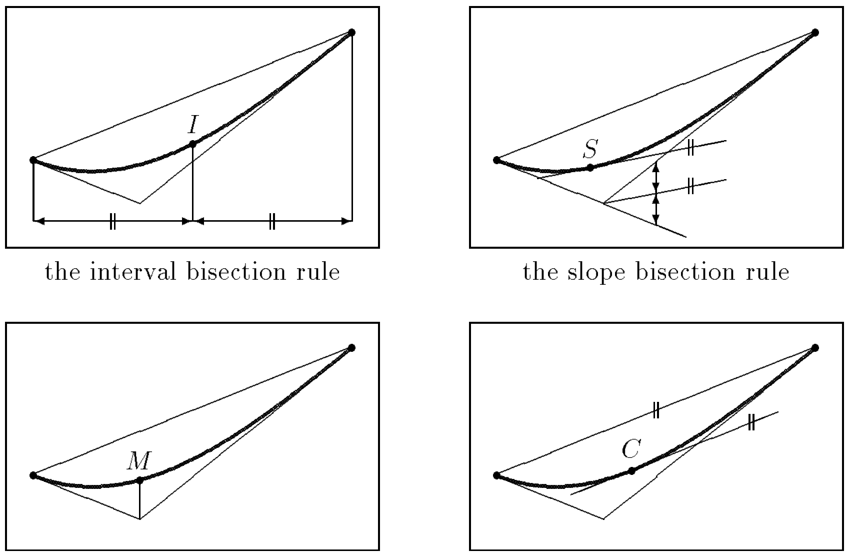
Different versions of the Sandwich algorithm differ in the way how they partition their intervals into subintervals. We consider four partition rules, that would naturally come to one's mind (cf. figure 2):

(i) *The interval bisection rule:* The interval is partitioned into two equal parts.

(ii) *The slope bisection rule:* We find the supporting line whose slope is the mean value of the slopes of the tangents at the endpoints. We partition the interval at some point where this line touches the function.

(iii) *The maximum error rule:* The interval is partitioned at the breakpoint of the lower approximation, i. e., at the point where the error between the two approximations is maximum.

(iv) *The chord rule* is similar to the slope bisection rule. However, we take the slope of the line connecting the two endpoints as the slope of the supporting line.



the interval bisection rule

the slope bisection rule

the maximum error rule

the chord rule

Figure 2: Four partition rules for the Sandwich algorithm

The rules fall into two classes: Rules (i) and (iii) specify the *abscissa* of the new point, whereas rules (ii) and (iv) find the point by specifying the *slope* of a supporting line. Which way of specifying the new breakpoint is more convenient depends on the application. However, we will see in section 2 that there is a very close connection between rules (i) and (ii) and between rules (iii) and (iv).

There is another classification of the rules: The two bisection rules (i) and (ii) are the “stubborn” rules, whereas rules (iii) and (iv) are more adaptive to the shape of the function.

The chord rule is actually also a kind of maximum error rule, since it selects the point *on the function* whose distance from the *upper* approximation is maximum. The chord rule also locally minimizes the *area* between the upper approximation and the function (cf. for example Lew and Quarles [1989]).

The main result of this paper is that, for all four partition rules, the maximum error decreases quadratically with the number of iterations.

Motivation of the problem. A very closely related and equally important problem is that of approximating a convex plane figure by a polygon: The convex hull of the graph of a convex function is a convex figure, and on the other hand the boundary of a convex figure can be decomposed into, say, four pieces, each of which can be described as a convex function.

The applications for approximation of convex bodies and convex functions can be classified into two categories:

1. It is computationally expensive to evaluate the function and its derivative at a given point, but we want to get an approximate idea what the function looks like.

2. The function is completely known, but it is nevertheless expedient to replace it by a piecewise linear function with few pieces.

We list two examples of applications falling into the first category and three examples of the second category:

- Bicriteria linear programs: We are given two linear objective functions (describing, e. g., time and costs) which are to be optimized subject to a set of linear constraints. Since not both objectives can be optimized at the same time, we have to be satisfied with a description of the feasible (time,cost)-pairs. These pairs form a polygon in the plane (cf. figure 3), and the

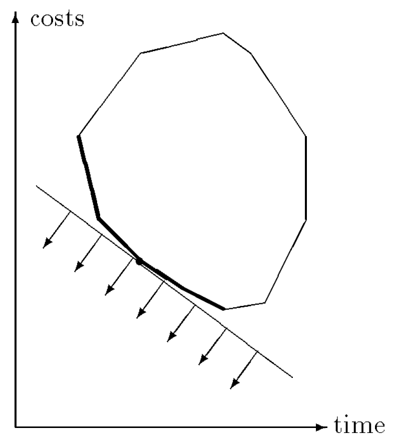


Figure 3: The feasible region of a bicriteria linear program.
The emphasized part of the boundary is the efficient point curve.

efficient points form part of its boundary. For large problems, the work to compute all vertices of this polygon might be prohibitive. In such a situation, an approximation of the polygon which can be computed at reasonable cost is better than nothing. For the case of bicriteria minimum-cost network flow problems, the Sandwich algorithm was successfully tested with a couple of different partition rules by Fruhwirth, Burkard, and Rote [1989].

- Parametric linear programs with one parameter in the objective function or on the right-hand sides are similar to the previous case.
- In mathematical programming a convex constraint is often replaced by a piecewise linear constraint, since such constraints can better be dealt with.
- In computational geometry, the dependence of the running time of algorithms on the number of polygon vertices is often quadratic or of even higher degree. (for example in motion planning). Replacing the input of a problem by a simpler approximation is a way to get approximate results while speeding up the calculations. (A different approximation problem from ours, which was motivated in this way, has been considered in Fleischer et al. [1990], [1992].)
- In image processing, a complicated polygon boundary might be replaced by a simpler polygon in order to eliminate input noise or to reduce the storage requirement.

In the applications of the first category, the problem is to get an acceptable approximation with as few function evaluations as possible. Here the Sandwich algorithm is the ideal candidate, because it uses just the best approximation that can be obtained from the information that is known.

In the second case, the problem of finding an approximating function with a specified error bound and the smallest possible number of linear pieces can be solved in principle, and there are direct methods for it: For optimal approximations, cf. Imai and Iri [1986], Cantoni [1971], or the surveys in Imai and Iri [1988] or Kurozumi and Davis [1982]. For approximations which are asymptotically optimal for small error bounds, cf. McClure and Vitale [1975] and Müller [1992]. However, some of these methods are complicated and slow, and it is often not even clear what the optimization criterion of the approximation should really be. Thus, the Sandwich algorithm might still be the method of choice if a simple and fast algorithm with a good performance guarantee is asked for.

Previous and related results. The problem of approximating a convex body by a polygon (or polytope) has attracted a great deal of attention in the theoretical literature (cf. the surveys in Gruber [1983], [1992] or Gruber and Kenderov [1982]). It is well known that the distance between a given convex plane figure and its best approximating n-gon is $O(1/n^2)$. regardless of whether we This holds for approximations by enclosed or enclosing or arbitrary polygons, and for a variety of error measures, like Hausdorff distance and area of the symmetric difference (cf. Gruber [1983]). This convergence rate is best possible (as is easy to see, by considering the case of a circle). For the approximation of convex *functions*, with the maximum *vertical* distance as the error, no algorithm can guarantee a convergence rate better than $O(1/n^2)$: The parabola $y = x^2$ is the worst case.

The Sandwich algorithm is so natural that is has been proposed very often, with different applications in mind; especially the chord rule has been popular, sometimes even for approximating non-convex curves (see e. g. Ramer [1972], Freeman and Shapiro [1975], Aneja and Nair [1979], or Khang and Fujiwara [1989] in higher dimensions). However, steps towards a thorough analysis were taken only in recent years. Noltemeier [1970] considered the maximum error rule for approximations which do not use derivatives and proved that the approximation error converges to zero. Ruhe [1988] (see also Ruhe [1991]) proved that the error of the chord rule decreases at least linearly. Lew and Quarles [1989] considered the chord rule for maximizing the area of a polygon inscribed inside a convex curve in the plane. However, in each iteration they subdivide *every* interval of the approximation, They showed that this non-adaptive method yields maximum-area inscribed polygons if and only if the convex curve is an arc of a conic section. Gruber [1991] showed that by this simple method of subdividing intervals uniformly a quadratic convergence rate can be achieved if the function $h(x)$ is twice continuously differentiable in the whole interval including the endpoints (see Gruber [1991], theorem 4 and remarks 8 and 4). Uniform subdivision is however not sufficient for a function like $h(x) = x^{3/2}$, $0 \le x \le 1$, or for non-differentiable functions. The Sandwich algorithm with any of the four partition rules can handle these functions well because it is more adaptive.

Incidentally, when the Sandwich algorithm is used to approximate a parabola $h(x) = x^2$, the sequence of upper approximations is just the sequence which Archimedes used to exhaust the area of a parabolic segment, in his second proof of his area formula (Archimedes [§§17–24], cf. also Boyer [1968], pp. 142–143). Archimedes constructed his sequence of polygons according to the chord rule.

A proof of the quadratic convergence rate of the Sandwich algorithm for the interval bisection rule was given in Sonnevend [1984] (in a more general setting), and for the case of the interval bisection slope bisection rules in Burkard, Hamacher, and Rote [1992] (with a different proof). Fruhwirth, Burkard, and Rote [1989] proved analogous results for the *angle bisection rule*, a natural variation of the slope bisection rule. The relation between these proofs and our proof will be discussed in the concluding section.

Ruhe and Fruhwirth [1990] considered the approximation criterion of ε-efficiency, which is useful in connection with multicriteria problems (see also Ruhe [1991]). They investigated the performance of the chord rule in this context.

Our results are a quantitative counterpart to the results on *probing* of polygons (cf. Cole and Yap [1987] or Skiena [1989]), where a polygon is to be reconstructed by asking for the intersection of the polygon with a specified line or for the supporting line with a specified direction. These two types of probes correspond, in our problem setting, to a function evaluation and to finding the point with a given value of the derivative. The results on probing are more qualitative in nature, since the polygons are to be reconstructed *exactly* and the number of probes depends only on the number of polygon sides.

Overview of the paper. In section 2 we discuss duality between convex functions and we establish a close connection between the partition rules which specify the abscissa and those which specify the slope. The next two sections (sections 3–4) are devoted to the quadratic convergence theorems, which constitute the main part of the paper. They are applied to the approximation of convex plane figures in section 5. The next two sections complement our worst-case convergence results by studying how the different rules may behave for particular functions: In section 6 we compare the four partition rules from a theoretical point of view. We will see that there is no reason to prefer any rule to another, because each rule can perform arbitrarily badly when compared to the other rules. In section 7 we show the behavior of our partition rules for a few selected functions. In the final section 8 we summarize the conclusions from the comparison of the rules, and we discuss possible extensions and open problems.

2 Geometric duality of convex functions

We can describe a convex function $h\colon [a,b] \to \mathbb{R}$ as the set of all pairs $((p,q),(k,d))$, where (p,q) is a point on the graph of the function, i. e., $q = h(p)$, and $y = kx + d$ is a supporting line in this point:

$$q = kp + d = h(p), \text{ and for all } x \in [a,b]\colon h(x) \ge kx + d.$$

The dual transformation $\mathcal{D}$ maps each pair $((p,q),(k,d))$ to the pair $((k,-d),(p,-q))$. It maps points to lines and lines to points by the polarity with respect to the parabola $y = x^2/2$. (This mapping is also a special projective duality.) This transformation is a tool which is often used in computational geometry (cf. Edelsbrunner [1987], section 1.4 or 15.2). It transforms the given function h into another convex function $\mathcal{D}(h)$. If the derivatives of h and $\mathcal{D}(h)$ are defined, they are inverse functions of each other. $\mathcal{D}(h)$ is also known as the *conjugate* of h (cf. Rockafellar [1970], §12 and §26) or the *Legendre transform* of h (cf. Arnol′d [1978], pp. 61-63).

As the slopes of the supporting lines at the endpoints of the definition interval are not unique, there is some arbitrariness regarding the domain of $\mathcal{D}(h)$. At the left endpoint a, there are supporting lines with slopes in the range between $-\infty$ and $h^+(a)$, the right derivative of h at a. If we include the all, $\mathcal{D}(h)$ will be defined on the whole of $\mathbb{R}$ and it will have linear pieces in $[-\infty, h^+(a)]$ and in $[h^-(b), \infty]$, but usually we don't want this. On the other hand, if we always define $\mathcal{D}(h)$ on the interval $[h^+(a), h^-(b)]$, we loose the involutory property of $\mathcal{D}$: If h has linear pieces at its ends, $\mathcal{D}(\mathcal{D}(h))$ will have lost these pieces. However, if h is strictly convex, then we will always have $\mathcal{D}(\mathcal{D}(h)) = h$.

Applying the interval bisection rule to a function corresponds to applying the slope bisection rule to its dual (with the above precautions regarding the domain of the dual). The vertical errors are the same in both cases. Similarly, the maximum error rule and the chord rule are dual to each other: With the maximum error rule, we look for the *point* with the same *abscissa* as the *intersection* of the two *supporting lines* at the boundaries of the interval; with the chord rule, we look for the *line* with the same *slope* as the *connection* of the two *points* at the boundaries of the interval.

Therefore, in the proofs below, it will be sufficient to consider only one rule of each pair of rules.

3 The interval and slope bisection rules

The following elementary lemma, which is also the heart of all existence proofs of $O(1/n^2)$ approximations, is the basis of our proofs.

Lemma 1 (Lemma 2.1 of Burkard, Hamacher, and Rote [1992]) *Let $L = b - a$ denote the length of an interval $[a,b]$, and let $\Delta = h^-(b) - h^+(a)$ denote the slope difference of the two parts of the lower approximation. Then the maximal vertical error ε in this interval is bounded as follows:*

$$\varepsilon \le \frac{L\Delta}{4}.$$

Proof: If the greatest error between the lower and the upper approximation occurs at some point x in the interval $[a,b]$, elementary geometric considerations lead to a lower bound for Δ in terms of $x - a$, L, and ε. The smallest value of this bound is assumed when x is the midpoint of the interval, and it yields the inequality in the lemma. ■

The next lemma is the basis of the induction step in our inductive proofs:

Lemma 2 *If $L_1, L_2, \Delta_1, \Delta_2 \ge 0$, $\varepsilon > 0$, $L_1 + L_2 \le L$, $\Delta_1 + \Delta_2 \le \Delta$, then*

$$\sqrt{L_1\Delta_1/\varepsilon} + \sqrt{L_2\Delta_2/\varepsilon} \le \sqrt{L\Delta/\varepsilon}.$$

Proof:

$$\begin{aligned}\left(\sqrt{L_1\Delta_1} + \sqrt{L_2\Delta_2}\right)^2 &\le \left(\sqrt{L_1\Delta_1} + \sqrt{L_2\Delta_2}\right)^2 + \left(\sqrt{L_1\Delta_2} - \sqrt{L_2\Delta_1}\right)^2 \\ &= (L_1 + L_2)(\Delta_1 + \Delta_2) \le L\Delta.\end{aligned}$$

The lemma follows by dividing by ε and taking square roots. ■

We need two more elementary lemmas, whose proofs are omitted. The next one is needed to ensure that the induction hypothesis can be applied, and lemma 4 will help us to deal with the rounding that occurs in the theorem:

Lemma 3 *For $x > 3$ we have:*

$$\left\lceil \sqrt{1/2} \cdot x \right\rceil \le \lceil x \rceil - 1.$$

Lemma 4 *If $c \geq a + b$ then $\lceil c-2 \rceil \geq 1 + \lceil a-2 \rceil + \lceil b-2 \rceil$.*

Now we are ready for the main theorem of this section. In the statement of the theorem we assume that the Sandwich algorithm computes $h(x)$ and the two one-sided derivatives $h^-(x)$ and $h^+(x)$ for each new breakpoint x, and it uses $h^-(x)$ as the slope of the lower approximation to the left of x and $h^+(x)$ on the right side of x. Since h is convex, the one-sided derivatives exist always. However, it is sufficient and sometimes more practical to just take any value between $h^-(x)$ and $h^+(x)$ as the slope of the lower approximation (as long as we take a steeper slope to the right of x than to the left of x), and the theorem still holds. (Some care has to be taken in the case of the slope bisection rule; see the remarks at the end of this section.)

Theorem 1 (Theorem 2.3 of Burkard, Hamacher, and Rote [1992]) *Let h be a convex function defined on an interval $[a, b]$ of length $L = b - a$, and suppose that the function values $f(a)$ and $f(b)$ and the one-sided derivatives $h^+(a)$ and $h^-(b)$ have been evaluated. Let $\Delta = h^-(b) - h^+(a)$. Then, in order to make the greatest vertical error between the upper and the lower approximation smaller than or equal to ε, the interval bisection rule or the slope bisection rule needs at most $z(L\Delta/\varepsilon)$ additional evaluations of $h(x)$, $h^-(x)$, and $h^+(x)$, where*

$$z(L\Delta/\varepsilon) = \begin{cases} 0, & \text{for } L\Delta/\varepsilon \leq 4, \\ \left\lceil \sqrt{\dfrac{9}{8}\dfrac{L\Delta}{\varepsilon}} - 2 \right\rceil, & \text{for } L\Delta/\varepsilon > 4. \end{cases}$$

Before proving the above bound, we would like to point out that the expression $L\Delta/\varepsilon$ is really the natural measure in terms of which $z(L\Delta/\varepsilon)$ should be expressed: If we scale the function in the y-direction (replacing $h(x)$ by $f \cdot h(x)$ for some $f > 0$), Δ and all vertical dimensions are multiplied by f. Thus, we also replace ε by $f\varepsilon$, and indeed, $L\Delta/\varepsilon$ is invariant under this transformation. Similarly, if we scale the function equally in both directions, replacing $h(x)$ by $f \cdot h(x/f)$, Δ remains unchanged but L and ε are multiplied by f, and $L\Delta/\varepsilon$ is again unchanged.

Proof: We first consider the interval bisection rule. We prove the theorem by induction on the integer number $z(L\Delta/\varepsilon)$. More precisely, our induction hypothesis is that the given expression for $z(L\Delta/\varepsilon)$ is an upper bound on the number of iterations for all convex functions for which $z(L\Delta/\varepsilon)$ is less than a certain value n, and then we prove it for $z(L\Delta/\varepsilon) = n$. The first case ($z(L\Delta/\varepsilon) = 0$), which establishes the basis for the induction, is directly equivalent to lemma 1.

In the other case, $z(L\Delta/\varepsilon)$ is at least one. If the vertical error is greater than ε then the interval bisection rule will evaluate $h(x)$, $h^-(x)$, and $h^+(x)$ for $x = (a + b)/2$, thus splitting the interval into two intervals of length $L/2$, with slope differences Δ_1 and Δ_2, where $\Delta_1 + \Delta_2 \leq \Delta$. Thus we have to prove

$$z(L\Delta/\varepsilon) \geq \max_{\Delta_1+\Delta_2 \leq \Delta} \left(1 + z\left(\tfrac{L}{2}\Delta_1/\varepsilon\right) + z\left(\tfrac{L}{2}\Delta_2/\varepsilon\right)\right).$$

We distinguish three subcases:

(i) Both $z(\frac{L}{2}\Delta_1/\varepsilon)$ and $z(\frac{L}{2}\Delta_2/\varepsilon)$ are zero. Then there is nothing to prove.

(ii) $z(\frac{L}{2}\Delta_2/\varepsilon) = 0$, but $z(\frac{L}{2}\Delta_1/\varepsilon) > 0$ (or vice versa): This means that $\frac{L}{2}\Delta_1/\varepsilon > 4$. We have to show that $z(L\Delta/\varepsilon) \geq 1 + z(\frac{L}{2}\Delta_1/\varepsilon)$. Lemma 3, applied to $x = \sqrt{\frac{9}{8}L\Delta_1/\varepsilon}$, implies that $z(\frac{L}{2}\Delta_1/\varepsilon)$ is strictly smaller than $z(L\Delta_1/\varepsilon)$, which in turn is smaller than $z(L\Delta/\varepsilon)$. This ensures that the induction hypothesis can be applied, and it also proves the theorem.

(iii) Both $z(\frac{L}{2}\Delta_1/\varepsilon)$ and $z(\frac{L}{2}\Delta_2/\varepsilon)$ are > 0. We have to show:

$$\left\lceil \sqrt{\frac{9}{8}\frac{L\Delta}{\varepsilon}} - 2 \right\rceil \geq 1 + \max_{\Delta_1+\Delta_2 \leq \Delta} \left(\left\lceil \sqrt{\frac{9}{8}\frac{\frac{L}{2}\Delta_1}{\varepsilon}} - 2 \right\rceil + \left\lceil \sqrt{\frac{9}{8}\frac{\frac{L}{2}\Delta_2}{\varepsilon}} - 2 \right\rceil \right).$$

As in the previous case, it follows from lemma 3 that the induction hypothesis can be used. By lemma 4, it is sufficient to show the following relation:

$$\sqrt{(9/8)L\Delta/\varepsilon} \geq \max_{\Delta_1+\Delta_2 \leq \Delta} \left(\sqrt{\frac{9}{8\varepsilon}\frac{L}{2}\Delta_1} + \sqrt{\frac{9}{8\varepsilon}\frac{L}{2}\Delta_2} \right).$$

By canceling the common factor $\sqrt{9/8}$ on both sides, the above statement becomes a direct consequence of lemma 2.

For the slope bisection rule, the roles of L and Δ must simply be interchanged. ■

The constant 9/8 in the theorem is best possible. For a worst-case example, x in lemma 3 must be just greater than 3. We define a function h on two adjacent intervals of lengths $L_1 = L_2 = 1$, with slope differences $\Delta_1 = 0$ in the left interval and Δ_2 just a little bigger than 4ε, in the right interval. This function is linear in the left interval. In the right interval, L_2 and Δ_2 just barely fail to satisfy the inequality of lemma 1. Thus, we can set up the function h in the right interval in such a way that the error is greater than ε, for example by setting h $h(x) = (\Delta_2/2)x^2 + ax + b$, for appropriate constants a and b. For the whole interval we have $L = 2$ and $\Delta = \Delta_2 > 4\varepsilon$, and thus we get $z(L\Delta/\varepsilon) = \lceil \sqrt{9 + \text{something}} - 2 \rceil = 2$, which is the true number of additional function evaluations. By taking 2^k instead of 2 intervals of length 1, alternatingly with $\Delta = 0$ and $\Delta > 4\varepsilon$, one can create infinitely many examples where the theorem is tight.

Corollary 1 *If we always subdivide the interval with largest error according to the interval bisection rule or the slope bisection rule, then the maximum vertical error after $n \geq 2$ evaluations of h, h^-, and h^+ is at most*

$$\frac{9}{8}\frac{L\Delta}{n^2}.$$

Proof: For the case $n = 2$ (evaluation only at the endpoints), we get the error bound $\varepsilon = 9/32 \cdot L\Delta > L\Delta/4$, and the result follows from lemma 1. For $n \geq 3$ we have $L\Delta/\varepsilon = 8n^2/9 \geq 8 > 4$, and from theorem 1 we conclude that we need at most

$$z(L\Delta/\varepsilon) = \left\lceil \sqrt{\frac{9}{8}\frac{L\Delta}{\varepsilon}} - 2 \right\rceil = \lceil n - 2 \rceil = n - 2$$

additional evaluations to achieve the claimed error bound ε. ■

The best possible Sandwich approximation (assuming that the whole function is given in advance, before any of the $n - 2$ partition points have to be chosen) would have the bound $(1/4)L\Delta/(n-1)^2$, as can be shown with the help of lemma 1. Thus, the bisection rules are only by a factor of 9/2 off the optimum.

A word of caution about implementing the slope bisection rule. Before the statement of theorem 1 we said that we may take any subgradient of h at x instead of the value $h^-(x)$ or $h^+(x)$. However, we cannot let this subgradient, which we denote by $h^\bullet(x)$, depend only on x. Consider the function $h(x) = \max(0, x^2 + 4x)$ in the interval $-1 \leq x \leq 1$, which has a breakpoint at $x = 0$. After one iteration of the slope bisection we have two subintervals $[-1, 0]$ and $[0, 1]$. For the right subinterval, the Sandwich algorithm determines the mean value between the slopes $h^\bullet(0)$ and $h^\bullet(1)$. If we always took $h^\bullet(0) = h^-(0) = 0$ and $h^\bullet(1) = h^-(1)$, we would get $(h^\bullet(0) + h^-(1))/2 = 3$ for the slope of next partition point. But this would again yield the point $x = 0$, and the Sandwich algorithm would cycle forever.

Therefore we have to take the following precaution. The Sandwich algorithm with the slope bisection rule uses derivatives on two occasions: as slopes of the lower approximation, and for computing the slope $(h^+(a) + h^-(b))/2$ of the next partition point. In both cases, if a point x has previously been determined as the point where a supporting line of slope s touches the function h then we may take instead of the value of $h^+(x)$ any subgradient of h at *x which is at least s*; similarly, the value that we take instead of $h^-(x)$ must be at most s. It is easy to incorporate this rule into the algorithm, and thus the slope bisection rule is viable even if we do not want to compute one-sided derivatives.

4 The maximum error rule and the chord rule

We need an additional lemma which is used to bound the product $L_1\Delta_1$ for the *left* subinterval from below when the error in the *right* subinterval is more than ε after the maximum error rule has been applied. This, in connection with lemma 2, will take the place of lemma 3 in ensuring that the induction hypothesis can be applied. (This is in fact the only point where the difference between the partition rules comes in.)

Lemma 5 *Consider an interval $[b,c]$ which has been subdivided at the breakpoint d of the lower approximation, as in the maximum error rule (see figure 4). Suppose that in one of the subintervals (say, $[d,c]$), the vertical error GG' is greater than ε. Then the length $L_1 = d-b$ of the other interval and its slope difference $\Delta_1 = h^-(d) - h^+(b)$ satisfy the following relation:*

$$L_1\Delta_1 > \varepsilon.$$

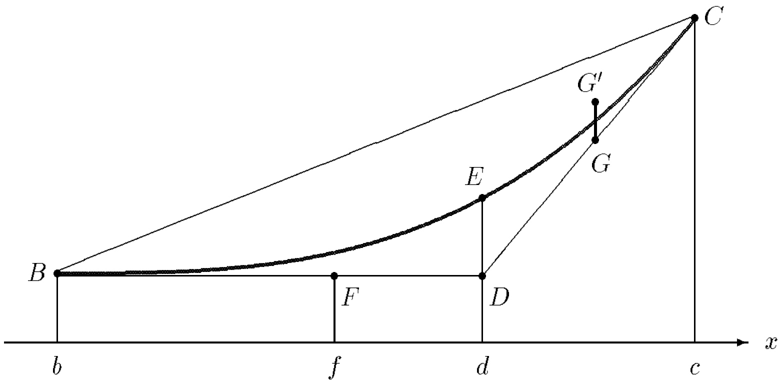


Figure 4: Proof of lemma 5. GG' is greater than ε.

Proof: We clearly have $DE \geq GG' > \varepsilon$, since the lines EG' and DG meet in C, which is to the right of G, and D is to the left of G. We denote the slope of the line through two points X and Y by slope(XY). From the triangle DEF, we have then

$$\begin{aligned} DE &= (d-f)\cdot(\mathrm{slope}(FE) - \mathrm{slope}(FD)) \\ &= (d-f)\cdot(h^-(d) - h^+(b)) \leq (d-b)\cdot(h^-(d)-h^+(b)) = L_1\Delta_1. \end{aligned}$$ ■

By duality, the above lemma is also true if the interval is subdivided according to the chord rule.

Theorem 2 *Let h be a convex function defined on an interval $[a,b]$ of length $L = b-a$, and suppose that the function values $f(a)$ and $f(b)$ and the one-sided derivatives $h^+(a)$ and $h^-(b)$ have been evaluated. Let $\Delta = h^-(b) - h^+(a)$. Then, in order to make the greatest vertical error between the upper and the lower approximation smaller than or equal to ε, the maximum error rule or the chord rule needs at most $m(L\Delta/\varepsilon)$ additional evaluations of $h(x)$, $h^-(x)$, and $h^+(x)$, where*

$$m(L\Delta/\varepsilon) = \begin{cases} 0, & \text{for } L\Delta/\varepsilon \leq 4, \\ \left\lceil \sqrt{\dfrac{L\Delta}{\varepsilon}} - 2 \right\rceil, & \text{for } L\Delta/\varepsilon > 4. \end{cases}$$

Note that the first formula in the definition of $m(L\Delta/\varepsilon)$ is a special case of the second one, except for $L\Delta \leq \varepsilon$, where it would be negative.

Proof: We consider only the maximum error rule. For the chord rule, the result follows by duality.

We prove the theorem by induction on $m_\varepsilon(L\Delta)$. The induction basis, $m_\varepsilon(L\Delta) = 0$, is equivalent to lemma 1.

In the other case, $m_\varepsilon(L\Delta) \geq 1$. Thus, if the error is at most ε in both subintervals after the function has been evaluated at the point of largest error, we have had only one additional evaluation and the theorem is true.

Otherwise, let L, Δ, L_1, Δ_1, L_2, and Δ_2 denote the lengths and slope differences of the original interval $[a,b]$ and of the left and right subintervals. We assume w. l. o. g. that the vertical error exceeds ε in the right subinterval. From lemma 5 we get $L_1\Delta_1/\varepsilon > 1$, and from lemma 1 we get $L_2\Delta_2/\varepsilon > 4$.

Thus all we have to prove is that

$$m(L\Delta/\varepsilon) \geq \max_{\substack{\Delta_1+\Delta_2\leq\Delta \\ L_1+L_2=L \\ L_1\Delta_1>\varepsilon,\, L_2\Delta_2>4\varepsilon}} \left(1+m(L_1\Delta_1/\varepsilon)+m(L_2\Delta_2/\varepsilon)\right).$$

Lemma 2, together with $\sqrt{L_1\Delta_1/\varepsilon} \geq 1$, implies that $m(L_2\Delta_2/\varepsilon)$ is strictly smaller than $m(L\Delta/\varepsilon)$, and similarly, $m(L_1\Delta_1/\varepsilon) < m(L\Delta/\varepsilon)$. Hence the induction hypothesis can be applied. If $m(L_1\Delta_1/\varepsilon) = 0$ the theorem follows directly; otherwise, we have to show the following relation:

$$\left\lceil \sqrt{L\Delta/\varepsilon} - 2 \right\rceil \geq 1 + \max_{\substack{\Delta_1+\Delta_2\leq\Delta \\ L_1+L_2=L \\ L_1\Delta_1>\varepsilon,\, L_2\Delta_2>4\varepsilon}} \left(\left\lceil \sqrt{L_1\Delta_1/\varepsilon} - 2 \right\rceil + \left\lceil \sqrt{L_2\Delta_2/\varepsilon} - 2 \right\rceil \right).$$

By lemma 4, it is sufficient to show

$$\sqrt{L\Delta/\varepsilon} \geq \max_{\substack{\Delta_1+\Delta_2\leq\Delta \\ L_1+L_2=L}} \left(\sqrt{L_1\Delta_1/\varepsilon} + \sqrt{L_2\Delta_2/\varepsilon} \right),$$

which follows directly from lemma 2. ■

Corollary 2 *If we always subdivide the interval with largest error according to the maximum error rule or to the chord rule, then the greatest vertical error after $n \geq 2$ evaluations of h, h^-, and h^+ is at most*

$$L\Delta/n^2.$$

Proof: Similar to the proof of the corollary of theorem 1. ■

Note that lemma 1 is a special case of this corollary.

The above proof shows that we could even mix the application of the chord rule and the maximum error rule arbitrarily and still get the same bounds. (If we also include the bisection rules, we would get the bounds of theorem 1.)

It is quite easy to find examples where the bounds of theorem 2 and its corollary cannot be improved. Simply take the function $h(x) = x^2$, $0 \leq x \leq 1$. We have $L = 1$ and $\Delta = 2$, and any of our four partition rules will split every interval into half. After $n = 2^k$ evaluations of h, the interval $[0, 1]$ will thus be split into $2^k - 2$ intervals of lengths $1/2^k$ and one interval of length $1/2^{k-1}$, in which the error is $1/2^{2k-1}$. This coincides with the bound in the corollary. If we impose a bound ε which is slightly smaller than this error, we have to evaluate h at $2^k + 1$ points, which is equal to the bound $m(L\Delta/\varepsilon)$ of theorem 2.

A comparison with the best possible "off-line" approximation bound (cf. the discussion after corollary 1) shows that the maximum error rule and the chord rule are worse by a factor of 4. The above example shows why one cannot expect more from a simple "on-line" algorithm that selects its next partition point without taking into account how many further iterations it will make in the future: If every subinterval looks completely symmetric, the Sandwich algorithm can do nothing better than split each interval into half. But after 2^k evaluations, this partition will be very unfavorable, with one left-over interval that is twice as long as the remaining intervals and almost twice as long as in an optimal (equal-length) partition into $2^k - 1$ intervals.

5 The approximation of convex plane figures

We shall now apply our results to the approximation of convex plane figures. For approximating a convex figure (or a convex curve), it makes no sense to measure the error in the "vertical" direction. Thus, we take the Hausdorff distance, which is defined for an inner approximation P_{inner} and an outer approximation P_{outer} as follows:

$$\sup_{x\in P_{\text{outer}}} \inf_{y\in P_{\text{inner}}} d(x, y).$$

Here $d(x, y)$ denotes the Euclidean distance. When we consider the graph of a convex function as a convex curve, the Hausdorff distance is always bounded by the maximum vertical distance. Thus the bounds of theorems 1 and 2 for the "vertical" error in some direction immediately carry over to the Hausdorff metric.

Theorem 3 *The Sandwich algorithm approximates a convex plane figure P of circumference D by two n-gons ($n \geq 4$) with an error at most $9D/(n-2)^2$ in case of the interval or slope bisection rule, or at most $8D/(n-2)^2$ in case of the maximum error rule or the chord rule. Or vice versa, to achieve a specified error bound ε, the Sandwich algorithm needs at most $\tilde{z}(D/\varepsilon)$ or $\tilde{m}(D/\varepsilon)$ points, respectively, where*

$$\tilde{z}(D/\varepsilon) = \max\left\{4,\ \left\lceil\sqrt{9D/\varepsilon}+2\right\rceil\right\},$$

and

$$\tilde{m}(D/\varepsilon) = \max\left\{4,\ \left\lceil\sqrt{8D/\varepsilon}+2\right\rceil\right\}.$$

Proof: We determine the smallest enclosing axis-parallel rectangle of P. The points where this rectangle touches P divide the boundary into four pieces. By looking from the proper "diagonal" direction, each piece can be viewed as a convex function with slope between -1 and $+1$. Thus we have $\Delta \leq 2$, and the lengths L_1, L_2, L_3, L_4 are bounded by the lengths of the corresponding pieces of P's circumference. Then we apply the Sandwich algorithm to each piece individually. Now let us prove the last part of the theorem. For the maximum error rule or the chord rule, we get from theorem 2:

$$m(L_i \cdot 2/\varepsilon) \leq \sqrt{L_i \cdot 2/\varepsilon}, \quad \text{for } i = 1,2,3,4.$$

Assuming that we need at least 5 points we even get $m(L_i \cdot 2/\varepsilon) < \sqrt{L_i \cdot 2/\varepsilon} - 1$ for at least one i. Thus the number of points is bounded by

$$4 + m(2L_1/\varepsilon) + m(2L_2/\varepsilon) + m(2L_3/\varepsilon) + m(2L_4/\varepsilon) < 3 + \sqrt{2/\varepsilon}\left(\sqrt{L_1}+\sqrt{L_2}+\sqrt{L_3}+\sqrt{L_4}\right).$$

We have $L_1 + L_2 + L_3 + L_4 \leq D$. The sum of the square roots is thus maximized when $L_1 = L_2 = L_3 = L_4 = D/4$. Therefore, the number of points is an integer number less than $3 + \sqrt{8D/\varepsilon}$, and our expression for $\tilde{m}(D/\varepsilon)$ is the largest such number. The proof for $\tilde{z}(D/\varepsilon)$ works in the same way. The proof of the error bounds in the first part of the theorem is analogous to the proof of corollary 1. ■

We could decrease the multiplicative constants from 8 arbitrarily closely towards 2π and from 9 towards $9\pi/4$, by initially decomposing the boundary of P into more than four pieces. This would reduce the error of bounding the Hausdorff distance by the vertical distance. The constant for the best possible Sandwich approximation is $\pi/2$.

The only partition rule which carries over naturally from convex functions to convex figures is the chord rule, which is completely insensitive to directions. This might be a reason why the approximation of plane figures seems to be less convenient to deal with than the approximation of functions. The analog of slope bisection is angle bisection, which has been treated in Fruhwirth, Burkard, and Rote [1989]. Analogs of the interval bisection rule and the maximum error rule can be applied if some point in the interior of P is chosen. Corresponding rules would then be related to each other by a polarity with respect to a circle centered at this point. However, this polarity preserves only the "relative error" with respect to the center of that circle, but it does not preserve absolute distances like the dual transform $\mathcal{D}$.

6 Comparison of the partition rules — theoretical results

All partition rules that we have considered are optimal in the worst-case: For given values of L and Δ their worst case is only by a constant factor off the worst case for an "optimal" Sandwich approximation. It is conceivable that the Sandwich algorithm with some partition rule would even remain "close to optimal" for *all* functions, in the following sense: If there is an optimal approximation for a given function with n intervals, achieving an error ε, then the Sandwich algorithm constructs an approximation with an error $c\varepsilon$ in at most dn iterations, for some constants c and d independent of the function, ε, and n.

The purpose of this section is to dismiss any hopes about such an optimality property for any of the four partition rules.

Bad examples for the bisection rules. It is easy to construct an example where the interval bisection rule takes many iterations to reduce the error to E, whereas only one interval partition would be required (see figure 5): Take an interval of length 2^n where the function is constant except in the left-most piece of length 1 where it has slope $-2E$. Then the interval bisection rule will make $n-1$ iterations, whereas all other rules will find the exact function after the first iteration.

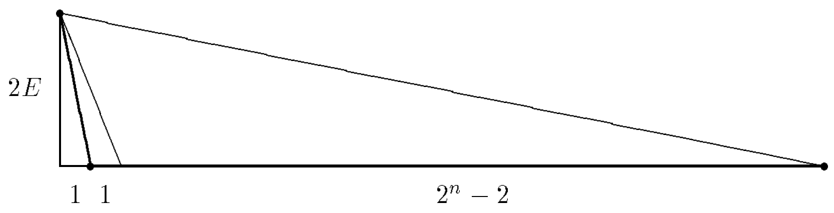


Figure 5: A bad example for the interval bisection rule.

By duality, an analogous example can be constructed where the slope bisection rule is arbitrarily bad when compared to the optimum or to the other rules. Before applying the duality to the example of figure 5, the function should be made strictly convex, for example by adding the function $x^2/10^n$ to it (see section 2).

Bad examples for the adaptive rules. We now construct an example where the chord rule performs badly. It is a piecewise linear function which we describe as a polygonal chain $(Q, O, P_n, P_{n-1}, \ldots, P_1, P_0)$, where $Q = (-E, E)$ for some parameter $E > 1$, $O = (0,0)$, and the points P_i have the following properties (see figure 6):

1. All points P_i lie above the x-axis, and each line P_iP_{i+1} intersects the x-axis in the point $(1-i/n, 0)$. This implies in particular that the function is indeed convex.
2. The line parallel to QP_i supporting the chain from below touches it in P_{i+1}. This implies that the chord rule, when subdividing the interval QP_i, will select P_{i+1} as its next partitioning point.

If we denote the coordinates by $P_i = (x_i, y_i)$ and write $d_i := 1 - i/n$, these conditions can be written as follows:

1.
$$\frac{y_{i+1}}{x_{i+1} - d_i} = \frac{y_i}{x_i - d_i}, \quad \text{for } i = 0, 1, \ldots, n-1 \tag{1}$$

2. The slope of QP_i is steeper than the slope $P_{i+2}P_{i+1}$; the latter line goes through the point $(d_{i+1}, 0)$, and thus we can write
$$\frac{y_i - E}{x_i + E} > \frac{y_{i+1}}{x_{i+1} - d_{i+1}}, \quad \text{for } i = 0, 1, \ldots, n-1 \tag{2}$$

Now we define P_i recursively by setting $P_n = (x_n, y_n) = (E + n, E)$ and
$$y_i = (2E+1)n \cdot x_{i+1}, \quad \text{for } i = n-1, n-2, \ldots, 2, 1, 0.$$

x_i is then defined from y_i, x_{i+1}, and y_{i+1} by (1). We have to show that this sequence also fulfills (2). But first we show the following, by induction on i (from $i = n$ down to 0):
$$x_i - y_i \geq i, \quad \text{for } i = n, n-1, \ldots, 1, 0.$$

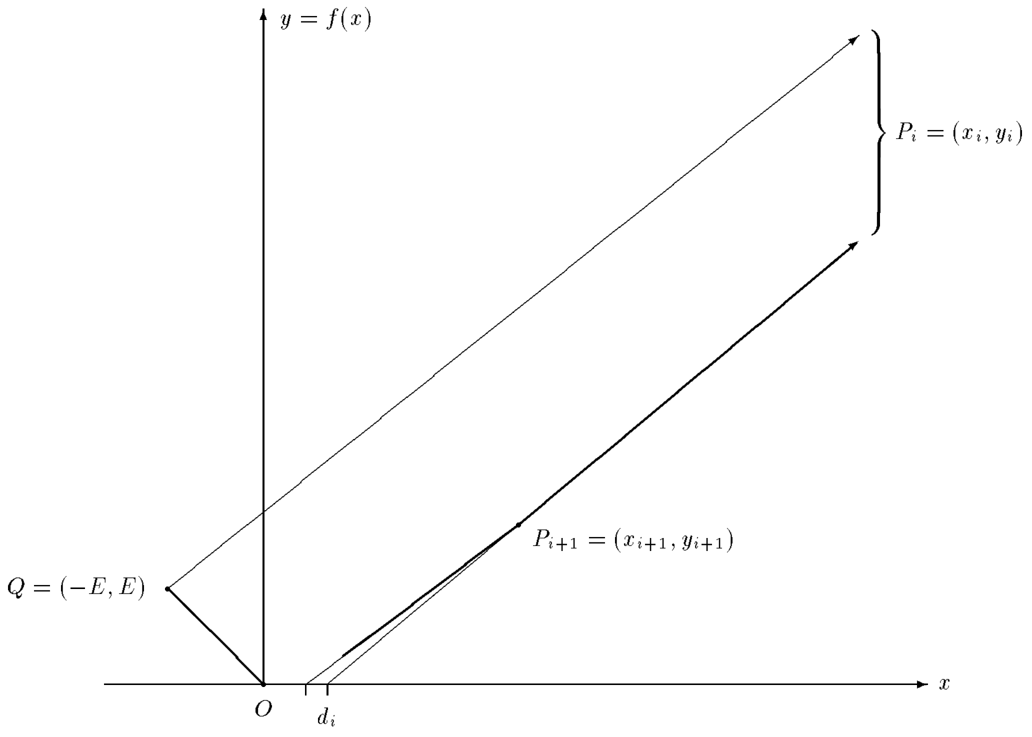


Figure 6: Construction of the bad example for the chord rule.
Because of the huge dimensions which arise in the actual construction, this figure cannot give true proportions. Although it is hard to notice, the two rays do eventually meet. Their intersection is P_i.

The case $i = n$ is true by the definition of P_n. For the inductive step, we have

$$\begin{aligned} x_i - y_i &= y_i\left(\frac{x_i - d_i}{y_i} - 1\right) + d_i \\ &= y_i\left(\frac{x_{i+1} - d_i}{y_{i+1}} - 1\right) + d_i \geq y_i\left(\frac{x_{i+1} - 1}{y_{i+1}} - 1\right) \\ &\geq y_i\left(\frac{y_{i+1} + (i{+}1) - 1}{y_{i+1}} - 1\right) = \frac{y_i}{y_{i+1}} \cdot i \geq i. \end{aligned}$$

In particular, $y_0 \leq x_0$ and thus P_0 lies below the line $y = x$. As the tangent at P_0 goes through the point $(d_0, 0) = (1, 0)$, the function between O and P_0 is contained in the triangle $O, (1,0), P_0$, and the initial Sandwich approximation for the interval OP_0 or any subinterval P_iP_0 has an error less than 1. On the other hand, the error of the approximation over an interval QP_i is always at least E, for all i.

Equation (2) is is shown as follows, using (1) and the definition of y_i:

$$\begin{aligned} \frac{y_{i+1}}{x_{i+1} - d_{i+1}} &= \frac{y_{i+1}}{x_{i+1} - d_i} \cdot \frac{x_{i+1} - d_i}{x_{i+1} - d_{i+1}} = \frac{y_i}{x_i - d_i} \cdot \frac{x_{i+1} - d_i}{x_{i+1} - d_{i+1}} \\ &\leq \frac{y_i}{x_i - d_i} \cdot \frac{x_{i+1} - 1/n}{x_{i+1}} \leq \frac{y_i}{x_i - 1} \cdot \left(1 - \frac{1}{nx_{i+1}}\right) \\ &= \frac{y_i}{x_i - 1} \cdot \left(1 - \frac{2E + 1}{y_i}\right) = \frac{y_i - 2E - 1}{x_i - 1} < \frac{y_i - E}{x_i + E}. \end{aligned}$$

Now, when we start the chord rule with the interval QP_0 and give it the error bound $\varepsilon = E$, it will take n iterations until it stops, whereas the single partition at point O would have sufficed to achieve even $\varepsilon = 1$.

If we extend our polygonal chain in the left through the points $Q_2 = (-x_0, y_0 + 2x_0 + 1)$ and $Q_1 = (-x_0/2, y_0/2 + 1)$, the other partition rules find a good approximation fast. This does not help the chord rule, since it will first split the interval $[-x_0, x_0]$ between Q_2 and P_0 at Q, in the right subinterval it idin the same position as before. It is not hard to check that the three other rules finish after at most four iterations.

By duality, an analogous family of bad examples exists for the maximum error rule.

Bad examples for an arbitrary subset of rules. By gluing together such bad examples that we constructed above, we can establish the following statement.

Theorem 4 *For any subset of the four partition rules {interval bisection, slope bisection, maximum error rule, chord rule}, there are functions where these rules perform close to the bounds of theorems 1 and 2, whereas the remaining rules perform arbitrarily much better than these bounds. More specifically: for any positive integer numbers N and m and real numbers $E \geq \varepsilon > 0$, there exists a convex function with the following properties:*

(i) *There is no Sandwich approximation with error bound ε and fewer than m intervals, but there is such a Sandwich approximation with at most $40m$ intervals.*

(ii) *The Sandwich algorithm with any of the designated "bad" partition rules does not achieve the error bound E in at most Nm iterations.*

(iii) *The Sandwich algorithm with any of the remaining partition rules achieves the error bound ε in at most $40m$ iterations.*

What this theorem says is that no partition rule "beats" another partition rule consistently for all functions; and one cannot even beat a partition rule by using any combination of other partition rules, letting them run independently, and taking the best approximation.

Proof: We take a bad example for the interval bisection rule (figure 5) with $n = \lceil N/2 \rceil + 1$; and we take a bad example for the chord rule with $n = \lceil N/2 \rceil$, with E/ε instead of E, and we scale it in the y-direction to make the error bound equal to ε instead of 1. Before gluing these "bad" pieces (and their duals) for the selected partition rules together we must make them look more uniform. Note that we can increase their slope difference Δ arbitrarily without destroying the essential properties of our examples, by replacing a very small piece at the left end with another piece of the appropriate steeper slope. In this way we can achieve that the two examples have equal values of the product $L\Delta$. By scaling each example in the x-direction, we can change the length L arbitrarily while leaving the product $L\Delta$ constant. Thus we can even achieve $L = \Delta = \sqrt{L\Delta}$, and our two bad examples will now have equal L and Δ values. By applying duality, we get two more examples for the dual rules with the same L and Δ values. Now we glue together side by side two copies of each of the four functions: We first add an appropriate linear function to the right copy so that the combined function has a breakpoint at the midpoint x with $h^+(x) - h^-(x) = 1$.

Now we have four function pieces, one for each rule, which look essentially the same from outside: They all have the same length L and the same slope difference Δ, and the two parts of the lower approximation intersect at the midpoint of the interval. We take m function pieces of each designated rule which we want to perform badly with respect to the other rules, we add more of these pieces to make the total number of pieces a power of two, and we glue them all together. This is the function whose existence is claimed in the theorem. Any rule will first decompose the interval evenly until it has broken the function into the half-pieces from which it was built. The function has less than $8m$ half-pieces, and the initial phase takes at most $8m$ iterations. Each of the "bad rules" of our designated subset will find at least $2m$ half-pieces which have been especially designed for it, and which will each take at least $N/2$ more iterations to achieve the bound E. This proves (ii). The other rules will finish off each half-piece in at most 4 iterations and achieve ε. Thus, we get a total number of at most $8m + 8m \cdot 4$ iterations, and this proves (iii) and the second part of (i). ■

7 Comparison of the partition rules — computational experiments

In this section we shall demonstrate the behavior of the Sandwich algorithm with the different partition rules for a couple of selected convex functions. We will show results for three functions that are given as explicit expressions and average results for convex functions that might be more typical of convex functions that occur in practice.

We have always subdivided the interval with the largest error, as in corollaries 1 and 2, and so we could observe the smallest possible error that the Sandwich algorithm achieves, for any number of iterations.

Example 1. The square root function. Our first function is the square root function $h(x) = -2\sqrt{x}$, $1 \le x \le 4$ (figure 7). The figure shows the maximum error for the four partition rules, interval bisection (I), slope bisection (S), maximum error rule (M) and chord rule (C), after $n = 240$ to $n = 1100$ iterations. Since the error ε decreases quadratically with n the ordinate axis has been normalized to show εn^2 instead of ε itself. Both axes are drawn to the same logarithmic scale. Thus, a period in the algorithm during which ε remains constant would be shown as a line of slope 2.

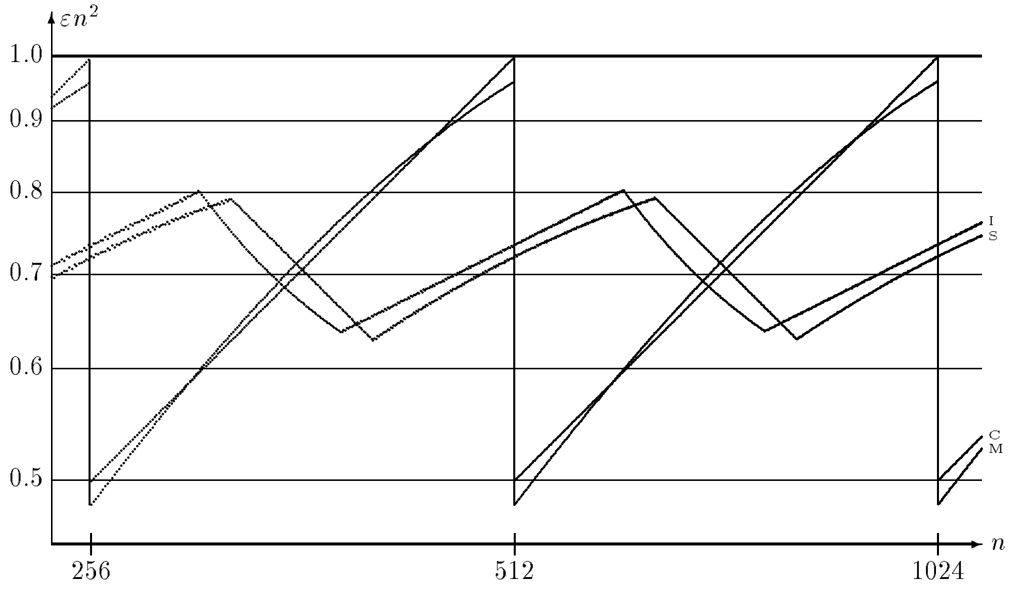


Figure 7: The error ε of the Sandwich algorithm for the function $h(x) = -2\sqrt{x}$, $1 \le x \le 4$, for iterations $n = 240, \ldots, 1100$. The four partition rules are designated by their initial letters.

We can make several observations: The curves are essentially periodic, with one period for every time the number of iterations is doubled: After the first few iterations the definition interval is divided into a number of subintervals. Subsequently, the Sandwich algorithm splits each of those intervals into two until the number of intervals is doubled.

For the maximum error rule and the chord rule, ε decreases very slowly until n reaches a power of two, where it suddenly drops. This means that these rules were rather successful in splitting the definition interval into 2^k subintervals where the approximation errors are quite balanced. The error decreases slowly for a long period of time, and then the error decreases drastically. (This phenomenon is shown in an extreme way by the function $h(x) = x^2$, see the remark at the end of section 4.) The other two rules have also a characteristic periodic shape, although there are no such sharp jumps and the error behaves more uniformly. Notice the tiny periodic oscillations on the rising portions of the curves.

In this example we have $L = 3$ and $\Delta = 1/2$. Thus corollaries 1 and 2 yield $\varepsilon n^2 \le 27/16 = 1.6875$ for the interval bisection rule and the slope bisection rule, and $\varepsilon n^2 \le 3/2$ for the maximum error rule and the chord rule. The dual of h is $(\mathcal{D}(h))(x) = -1/x$, $-1 \le x \le -1/2$. Thus, figure 7 could equally well represent $\mathcal{D}(h)$, with I and M interchanged with S and C, respectively.

Example 2. A hyperbolic arc. The next function that we are going to look at is an arc of the hyperbola $y^2 - x^2 = 1$. We have $h(x) = \sqrt{1 + x^2}$, $0 \le x \le 3/4$ (see figure 8). This time, all partition rules have big jumps at the powers of two. Quite different shapes of curves occur. We have $L = 3/4$ and $\Delta = 3/5$. Corollaries 1 and 2 yield $\varepsilon n^2 \le 0.50625$ and $\varepsilon n^2 \le 0.45$, respectively. The dual function is an arc of the unit circle: $(\mathcal{D}(h))(x) = -\sqrt{1 - x^2}$, $0 \le x \le 3/5$.

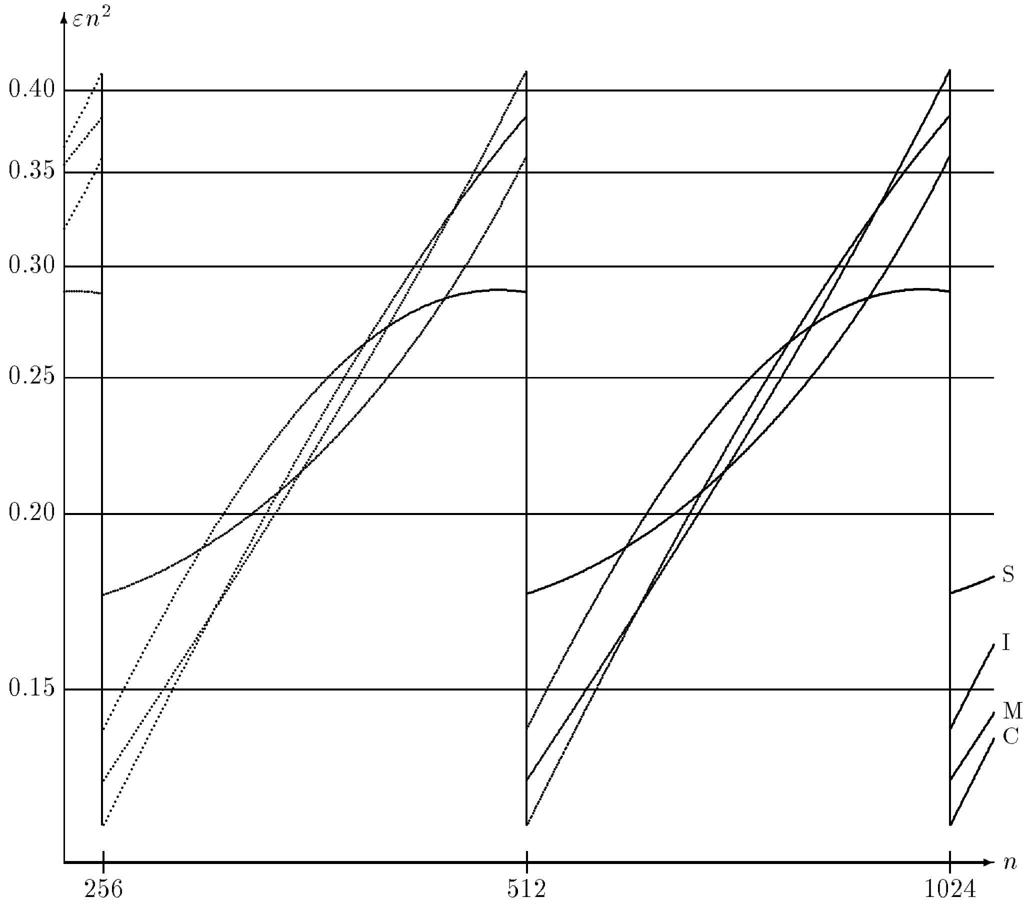


Figure 8: The error ε of the Sandwich algorithm for the function $h(x) = \sqrt{1 + x^2}$, $0 \le x \le 3/4$, (a hyperbolic arc).

Example 3. The sine function. Our third example is the sine function: $h(x) = -\sin x$, $0 \le x \le \pi/2$ (see figure 9). This example exhibits quite a variety of shapes, sometimes with little periodic jumps occurring between the main periods. We have $L = \pi/2$ and $\Delta = 1$, giving the bounds $\varepsilon n^2 \le 1.77$ and $\varepsilon n^2 \le 1.57$, respectively. The dual function is $(\mathcal{D}(h))(x) = -x \arccos(-x) - \sqrt{1 - x^2}$, $-1 \le x \le 0$.

Convex functions that "occur in practice". Since artificial examples alone are not satisfactory, we generated ten bicriteria minimum-cost network flow problems, similar to those in Fruhwirth, Burkard, and Rote [1989], and we computed the efficient point curves (cf. figure 3). We got ten piecewise linear convex curves which had between 4260 and 4800 linear pieces, with an average number of about 4550. Then we approximated these curves with each of the four partition rules. The plots of the approximation error are quite erratic and not so uniform as for the previous examples and are therefore not shown. Still, certain characteristic patterns that appear between iterations n and $2n$ tend to reappear between iterations $2n$ and $4n$.

Thus, we have measured for each function and each partition rule the average value, the minimum, and the maximum of εn^2 over the iterations $n = 51, 52, \ldots, 100$, over the iterations $n = 101, 102, \ldots, 200$, and over the iterations $n = 201, 202, \ldots, 400$. The following table shows the averages over these numbers over the ten examples.

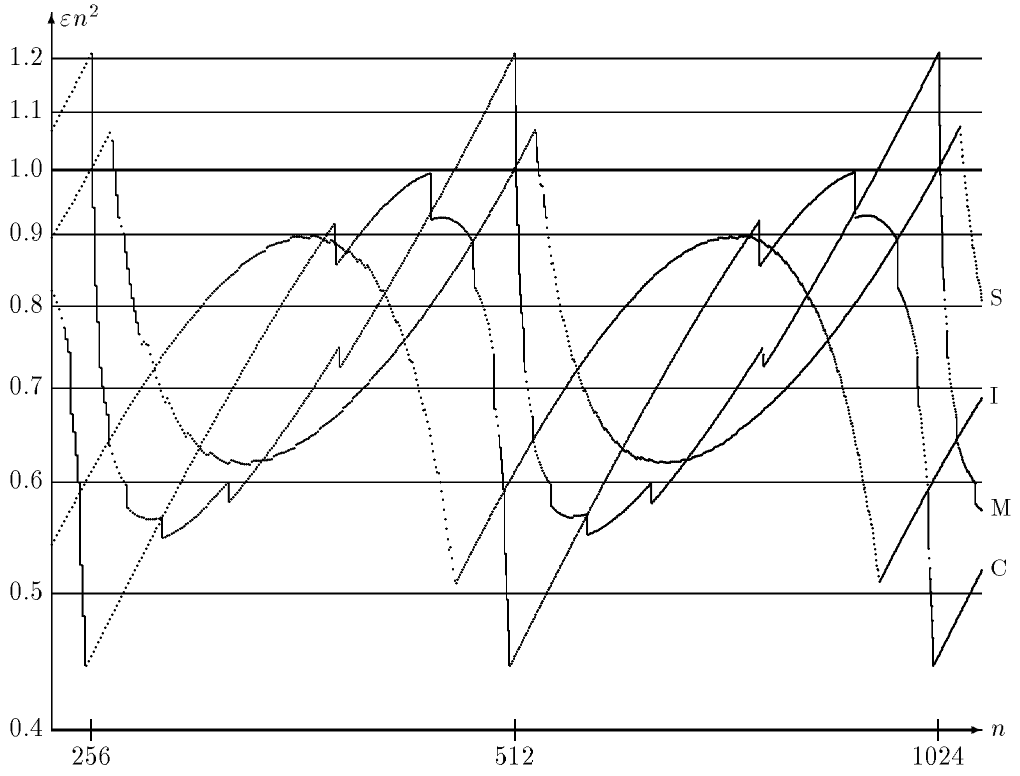


Figure 9: The error ε of the Sandwich algorithm for the function $h(x) = -\sin x,\ 0 \le x \le \pi/2$.

rule	εn^2: average, minimum, and maximum		
	$50 < n \le 100$	$100 < n \le 200$	$200 < n \le 400$
interval bisection	352.5 (316.0–387.0)	341.1 (315.9–367.1)	327.8 (309.2–353.3)
maximum error rule	334.3 (304.5–367.7)	326.1 (302.1–359.0)	308.9 (285.5–337.4)
slope bisection	332.0 (303.2–370.5)	309.7 (283.0–341.0)	273.9 (249.6–302.2)
chord rule	322.1 (291.4–361.2)	302.3 (280.4–332.7)	266.7 (246.1–293.5)

The three intervals from n to $2n$ are sufficiently wide to level out accidental deviations from the average behavior, and the maxima and minima are not just local irregularities.

If the function that is to be approximated is already piecewise linear, as in our case, the two rules which specify the slope (slope bisection and the chord rule) will terminate with the exact function after a finite number of iterations. This is reflected in the falling tendency of εn^2 as n gets larger. The maximum error rule will also eventually terminate with the exact function, although it will take about twice as many iterations. The interval bisection rule will in general never obtain the approximated function exactly. This gives the slope-specifying rules an advantage on the long run, but since we consider only $n \le 400$ iterations, this effect is not so strong. (It is possible to modifiy the maximum error rule and the interval bisection rule so that they take care of linear pieces of the function, and then this drawback disappears.)

The two adaptive rules are slightly superior to the corresponding bisection rules on the average. Unlike in the other three examples, the range of variation of the bisection rules is not always smaller than for the adaptive rules.

To get meaningful error bounds from theorems 1 and 2, we have to turn the efficient point curve counterclockwise by 45°, as in section 5. The average value of this bound for the chord rule was about 366, which compares reasonably with the actual figures.

Conclusion. When we compare the four partition rules as to their performance, we may say that the adaptive rules (maximum error rule and chord rule) are usually more successful in finding a good partition into intervals where the error is distributed evenly and the maximum global error is therefore small. However, this occurs only at certain times during the algorithm. With sequential algorithms that select the points one by one, the price that one may have to pay for a very good approximation at certain times is a great variation of the error at other times. On the average, the adaptive rules seem to win slightly. Generally, the bounds in theorems 1 and 2 and their corollaries estimate the order of magnitude of the actual error quite well.

8 Concluding remarks

The relation to other proofs. The proofs in the present paper work by induction on the number of function evaluations that are needed to make the global error smaller than a given error bound ε. In Burkard, Hamacher, and Rote [1992], our theorem 1 is proved in a different way: They consider the tree of intervals that is implicitly built during the algorithm. This proof gives more insight into the structure of the algorithm but it cannot be extended to cover the other two rules. Sonnevend [1984] establishes his results in a more general setting. His proof for the interval bisection rule works inductively, similarly to our proof; he does not specify constants in the expression $O(1/n^2)$.

Comparison of the partition rules. We have seen in section 6 that, in theory, everything can happen as regards the relative performance of the four rules for a particular function. In practice, the rules do not differ too much in their average behavior (see section 7). This is confirmed by the results reported in Fruhwirth, Burkard, and Rote [1989], who compared the chord rule and the angle bisection rule. The adaptive rules seem to be slightly better, but sometimes their performance may exhibit a much greater variation for different numbers of iterations, and thus the bisection rules might be preferable.

By the duality of convex functions, there is no theoretical difference between a rule which specifies the abscissa of the partition point and the corresponding slope rule. This is rather a matter of the practical implementation of computing the partition point. Consider the example of bicriteria linear programs mentioned in the introduction. Finding the point with a given slope is just an ordinary linear program with a single objective function (Figure 3 shows such an objective function.) Finding the point with a given abscissa amounts to solving the problem with one additional linear constraint, which might destroy some structure that is inherent in the problem (for example, the structure of a maximum flow problem).

All rules are invariant under affine transformations which leave the y-axis vertical. This includes in particular scalings of the coordinate axes. The chord rule is the only rule which is invariant under all affine transformations, including rotations. This makes it attractive for bicriteria problems, since it does not favor one objective function over the other.

One possible conclusion from the results of section 6 would be the development of some sort of combined "primal-dual" partition rule that would exclude most of those pathological examples. This would be of theoretical value only, and furthermore, we believe that it would be possible to construct for any conceivable "super-rule" pathological examples that fool this rule.

Higher dimensions. The problem of approximating a convex function of two variables is considerably more difficult, from an algorithmic viewpoint. If such a function and its (sub-)differentials have been evaluated at some points, the best upper approximation is the convex hull of a set of points in three dimensions, and the best lower approximation is the intersection of halfspaces.

Whereas these subproblems are still quite tractable in three dimensions — they can be solved in $O(n \log n)$ time, cf. Preparata and Shamos [1985] — they become more and more complex in higher dimensions. Moreover, in the one-variable case, the approximation problems for the subintervals are completely independent of each other. In higher dimensions, there is no straightforward way to define subregions into which the given domain is decomposed.

Sonnevend [1983] proposed an approximation scheme for convex functions of two variables which avoids the above difficulties by constructing the approximating functions only from local information: He essentially uses "square quadrisection" (in analogy to interval bisection). The resulting approximating function is in general not convex, and the bound on the global error could only be shown to be $O(\log n/n)$, as opposed to a rate of $O(1/n)$ for optimal approximations. It is open whether $O(\log n/n)$ is the true worst-case bound for that algorithm.

Gruber's result [1991] which was mentioned in the introduction holds in arbitrary dimensions: Essentially, it implies that one can evaluate the function on a set of points which form a "regular grid", and the resulting approximation will be asymptotically optimal if the function is twice continuously differentiable.

A bivariate extension of the Sandwich algorithm with the maximum error rule has been implemented and visualized (see Fruhwirth [1991]). Fruhwirth also discusses implementation issues, data structure representations, and he reports some numerical experience with tricriteria network flow problems. Khang and Fujiwara [1989] essentially use the maximum error rule as a method for finding all vertices of a higher-dimensional polytope which is given by linear inequalities.

Optimal Sandwich algorithms. The Sandwich algorithm acquires information about the function to be approximated step by step, it does not "see" the whole function. Therefore, it makes no sense to ask for optimality of the resulting approximation in the traditional meaning, with as few points as possible. However, the following type of question would still make sense:

> Given an initial Sandwich approximation and a number n, what is the best strategy to choose partition points so that the worst possible error after n iterations becomes as small as possible?

In other words, we ask for the best possible performance guarantee. Already for small values of n, this problem looks far from trivial. Even if the starting approximation is completely symmetric, and we are allowed to query the function at $n = 2$ additional points, it is hard to find the best choice for the first partition point. The situation is similar to the Fibonacci search algorithm for finding a minimum of a unimodal function, where the best search point is not the midpoint.

The proper setting for these questions would be the framework of informational complexity (cf. the monographs of Traub, Wasilkowski, and Woźniakowski [1988], Novak [1988], or Traub and Woźniakowski [1980]). Although optimization problems for convex functions have received a great deal of attention (cf. Nemirovsky and Yudin [1983]), problems of approximating convex functions have not been investigated in this context so far.

Acknowledgements

I thank Rainer E. Burkard and Horst Hamacher for the discussions initiating this work, and I thank Emo Welzl for pointing out the possibility of duality between the maximum error rule and the chord rule, which made it possible to eliminate one of two independent convergence proofs.

References

Yash P. Aneja and K. P. K. Nair [1979]
Bicriteria transportation problem, *Management Science* **25**, 73–78.

Archimedes
Quadratura parabolae, in: J. L. Heiberg (ed.): *Archimedis opera omnia*, vol. II, B. G. Teubner, Leipzig 1913, pp. 261–315.

Vladimir Igorevič Arnol'd [1978]
Mathematical Methods of Classical Mechanics, Springer-Verlag, New York, Berlin, Heidelberg.

Carl B. Boyer [1968]
A history of mathematics, Wiley.

Rainer E. Burkard, Horst Hamacher, and Günter Rote [1992]
Sandwich approximation of univariate convex functions with an application to separable convex programming, *Naval Research Logistics* **38**, 911–924.

A. Cantoni [1971]
Optimal curve fitting with piecewise linear functions, *IEEE Transactions on Computers* **C-20**, 59–67.

Richard Cole and Chee Keng Yap [1987]
Shape from probing, *J. Algorithms* **8**, 19–38.

Herbert Edelsbrunner [1987]
Algorithms in Combinatorial Geometry, Springer-Verlag, Berlin etc.

Rudolf Fleischer, Kurt Mehlhorn, Günter Rote, Emo Welzl, and Chee Yap [1990]
On simultaneous inner and outer approximation of shapes, in: *Proceedings of the Sixth Annual Symposium on Compututational Geometry*, Berkeley, California, June 6-8, 1990, Association for Computing Machinery, pp. 216–224.

Rudolf Fleischer, Kurt Mehlhorn, Günter Rote, Emo Welzl, and Chee Yap [1992]
Simultaneous inner and outer approximation of shapes, *Algorithmica* **8**, 365–389,

H. Freeman and R. Shapiro [1975], Determining the minimum-area encasing rectangle for an arbitrary closed curve, *Communications of the ACM* **18**, 409–413.

Bernd Fruhwirth [1991]
Approximation of convex functions and multicriteria linear programs, dissertation, Technische Universität Graz, Institut für Mathematik, August 1991.

Bernd Fruhwirth, Rainer E. Burkard, Günter Rote [1989]
Approximation of convex curves with application to the bicriteria minimum cost flow problem, *European Journal of Operational Research* **42**, 326–338.

Peter M. Gruber [1983]
Approximation of convex bodies, in: Peter M. Gruber and Jörg M. Wills (eds.): *Convexity and its applications*, Birkhäuser, Basel, Boston 1983, pp. 131–162.

Peter M. Gruber [1991]
Asymptotic estimates for best and stepwise approximation of convex bodies I, manuscript, Technische Universität Wien, Abteilung für Analysis, submitted for publication.

Peter M. Gruber [1992]
Aspects of approximation of convex bodies, chapter 1.10 in: Peter M. Gruber and Jörg M. Wills (eds.): *Handbook of Convex Geometry*, vol. **A**, North-Holland, Amsterdam, New York, Oxford, Tokio 1992, pp. 319–345.

Peter M. Gruber and Petar Kenderov [1982]
Approximation of convex bodies by polytopes, *Rendiconti Circ. Mat. Palermo, Serie II,* **31**, 195–225.

Hiroshi Imai and Masao Iri [1986]
An optimal algorithm for approximating a piecewise linear function, *J. Information Processing* **9**, 159–162.

Hiroshi Imai and Masao Iri [1988]
Polygonal approximations of a curve — formulations and algorithms, in: Godfried T. Toussaint (ed.): *Computational morphology — a computational geometric approach to the analysis of form*, North-Holland, Amsterdam, New York 1988, pp. 71–86.

Do Ba Khang and Okitugu Fujiwara [1989]
A new algorithm to find all vertices of a polytope, *Oper. Res. Lett.* **8**, 261–264.

Yoshisuke Kurozumi and Wayne A. Davis [1982]
Polygonal approximation by the minimax method, *Computer Graphics and Image Processing* **19**, 248–264.

John S. Lew and Donald A. Quarles [1989]
Optimal inscribed polygons in convex curves, *Amer. Math. Monthly* **96**, 886–902.

George Martelli [1962]
Jemmy Twitcher — A Life of the Fourth Earl of Sandwich, 1718–1792, Jonathan Cape, London.

D. E. McClure and R. A. Vitale [1975]
Polygonal approximation of plane convex bodies, *J. Math. Anal. Appl.* **51**, 326–358.

Josef S. Müller [1992]
Step by step approximation of plane convex bodies, *Archiv der Mathematik (Basel)* **58**, 606–610.

Arkadii Semenovič Nemirovsky and David Borisovič Yudin [1983]
Problem Complexity and Method Efficiency in Optimization, Wiley, Chichester, New York, Brisbane, Toronto, Singapore.

Hartmut Noltemeier [1970]
Sensitivitätsanalyse bei diskreten linearen Optimierungsproblemen, Lecture Notes in Operations Research and Mathematical Systems **30**, Springer-Verlag, Berlin, Heidelberg, New York.

Erich Novak [1988]
Deterministic and Stochastic Error Bounds in Numerical Analysis, Lecture Notes in Mathematics **1349**, Springer-Verlag, Berlin, Heidelberg, New York, London, Paris, Tokyo.

Franco P. Preparata and Michael Ian Shamos [1985]
Computational Geometry: an Introduction, Springer-Verlag, New York.

Urs Ramer [1972]
An iterative procedure for the polygonal approximation of plane curves, *Computer Graphics and Image Processing* **1**, 244–256.

R. Tyrrell Rockafellar [1970]
Convex Analysis, Princeton University Press, Princeton.

Günther Ruhe [1988]
Flüsse in Netzwerken — Komplexität und Algorithmen, Dissertation B, Technische Hochschule Leipzig, Sektion Mathematik und Informatik.

Günther Ruhe [1991]
Algorithmic Aspects of Flows in Networks, Kluwer, Dordrecht.

Günther Ruhe and Bernd Fruhwirth [1990]
ε-optimality for bicriteria problems and its application to minimum cost flows, *Computing* **44**, 21–34.

Steven S. Skiena [1989]
Problems in geometric probing, *Algorithmica* **4**, 599–605.

György Sonnevend [1983]
An optimal sequential algorithm for the uniform approximation of convex functions on $[0,1]^2$, *Appl. Math. Optim.* **10**, 127–142.

György Sonnevend [1984]
Sequential algorithms of optimal order global error for the uniform recovery of functions with monotone $(r-1)$ derivatives, *Analysis Mathematica* **10**, 311–335.

J. F. Traub, G. W. Wasilkowski, and H. Woźniakowski [1988]
Information-Based Complexity, Academic Press.

J. F. Traub and H. Woźniakowski [1980]
A General Theory of Optimal Algorithms, Academic Press.

Dr. Günter Rote
Technische Universität Graz
Institut für Mathematik
Steyrergasse 30
A-8010 Graz, AUSTRIA

Electronic mail: `rote@ftug.dnet.tu-graz.ac.at`.