Comput Visual Sci (2010) 13:83–97
DOI 10.1007/s00791-009-0126-4

REGULAR ARTICLE

Uniform tree approximation by global optimization techniques

Bernardo Llanas · Francisco Javier Sáinz

Received: 10 January 2008 / Accepted: 7 October 2008 / Published online: 14 November 2009

Abstract In this paper we present adaptive algorithms for solving the uniform continuous piecewise affine approximation problem (UCPA) in the case of Lipschitz or convex functions. The algorithms are based on the tree approximation (adaptive splitting) procedure. The uniform convergence is achieved by means of global optimization techniques for obtaining tight upper bounds of the local error estimate (splitting criterion). We give numerical results in the case of the function distance to 2D and 3D geometric bodies. The resulting trees can retrieve the values of the target function in a fast way.

Keywords Tree approximation · Precomputed maps · Uniform approximation · Branch and bound · D.C. optimization

1 Introduction

The analytical expression of most functions appearing in practical applications is unknown a priori. These functions are frequently given by

- Data provided by sensors and measurement devices.
- Data provided by the computer execution of a programmed algorithm.

In simulation applications we need to reproduce and interpolate the above data in a fast way. This can be done using a suitable approximant. An ideal approximant should verify the following conditions:

- C1) Simplicity. The approximant should consist of not too many elementary functions.
- C2) Precision. The approximation to the target function should be uniform and accurate.
- C3) Fast computation. The retrieval of data should be faster than that provided by the original data generator.

Piecewise affine (PA) functions satisfy condition C1 if the number of subsets in the partition is not excessively large. Moreover, if a PA-function is invertible, it can be inverted in a closed form. Thus PA-approximation is ideal for applications requiring the approximation of a function and its inverse. For example, the field of robotics is rife with cases where changes of coordinates play a key role:

- In mobile robot navigation [17,29].
- In the representation of gaits [28].

PA-functions have also been extensively used in computer graphics [27].

We give below a formal definition of uniform continuous PA-approximation.

Let $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d\}$ be a set of $d+1$ vectors in $\mathbb{R}^d$. This set is called affinely independent if the vectors $\{\mathbf{v}_1 - \mathbf{v}_0, \mathbf{v}_2 - \mathbf{v}_0, \ldots, \mathbf{v}_d - \mathbf{v}_0\}$ are linearly independent. Suppose now that $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d\}$ is an affinely independent subset of $\mathbb{R}^d$. The d-simplex T generated by $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d\}$, denoted by $\langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$, is defined to be the convex hull of the vectors $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d$. We denote by $|T|$ the diameter of a d-simplex T.

We denote by $\Pi_1(\mathbb{R}^d)$ the space of polynomials of degree less than or equal to 1, that is

$$\Pi_1(\mathbb{R}^d) = \{a_0 + a_1 x_1 + \cdots + a_d x_d : a_i \in \mathbb{R}, \quad i = 0, \ldots, d\}.$$

Communicated by G. Wittum.

B. Llanas (✉) · F. J. Sáinz
Departamento de Matemática Aplicada, ETSI de Caminos, Universidad Politécnica de Madrid, C/Profesor Aranguren s/n, 28040 Madrid, Spain
e-mail: ma07@caminos.upm.es

Consider the d-interval $I = [\mathbf{b}, \mathbf{c}]$, where $\mathbf{b}, \mathbf{c} \in \mathbb{R}^d$. We can find n (closed) simplices T_i such that

$$\mathring{T}_i \cap \mathring{T}_j = \varnothing, \quad i \neq j,$$
$$\bigcup_{i=1}^{n} T_i = I,$$

where $\mathring{T}$ denotes the interior of the set T (for example, see [4,24]). We call $\Delta \equiv \{T_i\}_{i=1}^n$ a partition of I. The set of partitions of I is denoted by $\mathcal{P}(I)$. We have the following result

Proposition 1 ([12]) *Let $T = \langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$ be a d-simplex and let f be a continuous function from T to $\mathbb{R}$, then there exists a unique affine map $I_T(f)$ such that*

$$I_T(f)(\mathbf{v}_j) = f(\mathbf{v}_j), \quad j = 0, 1, \ldots, d.$$

Let S be a compact set in $\mathbb{R}^d$. Let $C(S)$ be the class of functions g with domain S and range in $\mathbb{R}$ such that g is continuous on S. The customary norm for $C(S)$ is given by

$$\| g \|_{C(S)} = \max_{\mathbf{x} \in S} |g(\mathbf{x})|.$$

From the above result, given $f \in C(T)$ we can define the local approximation operator

$$I_T : C(T) \to \Pi_1(\mathbb{R}^d)$$
$$f \to I_T(f).$$

The affine map $I_T(f)$ has the following properties

Proposition 2 *Let $T = \langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$ be a d-simplex contained in I and $f \in C(I)$, then*

(a) *If $\mathbf{x} = \sum_{i=0}^d \lambda_i \mathbf{v}_i$, $\lambda_i \geq 0$, $i = 0, \ldots, d$ and $\sum_{i=0}^d \lambda_i = 1$, then*

$$I_T(f)(\mathbf{x}) \equiv \sum_{i=0}^{d} \lambda_i f(\mathbf{v}_i).$$

$(\lambda_0(\mathbf{x}), \ldots, \lambda_d(\mathbf{x}))$ are usually called the barycentric coordinates of $\mathbf{x}$. Each $\lambda_i(\mathbf{x}) \in \Pi_1$ is uniquely determined by $\lambda_i(\mathbf{v}_j) = \delta_{ij}$.

(b) *Given $\epsilon > 0$ there exists a number $\delta > 0$ such that for all d-simplices T contained in I with $|T| < \delta$ we have $\| f - I_T(f) \|_{C(T)} < \epsilon$.*

Proof *a*) follows from Proposition 1. *b*) is a consequence of *a*) and the uniform continuity of f on I. □

Let Δ be a partition of I and $f \in C(I)$. We define the approximant

$$s = \sum_{T_i \in \Delta} I_{T_i}(f) \chi_{T_i},$$

where χ_{T_i} is the characteristic function of the set T_i. $s \in C(I)$ if the partition (triangulation) is conforming and is piecewise continuous, otherwise. The partitions generated by the algorithms described in this paper are nonconforming.

We try to find an algorithm to solve the following uniform continuous PA-approximation (UCPA) problem:

Given $\epsilon > 0$ and $f \in C(I)$, find $\Delta \in \mathcal{P}(I)$ such that

$$\| f - s \|_{L^\infty(I)} = \max_{T_i \in \Delta} \| f - I_{T_i}(f) \|_{C(T_i)} < \epsilon.$$

We call such Δ ϵ-PA tessellation. We add the following condition

The tessellation should be of moderate size (C1).

Some usual algorithms for PA-approximation do not accomplish these requirements, for example:

- **Uniform refinement procedures**. These methods, based on Proposition 2b), are non adaptive, that is, regions in which the function is affine are subdivided in the same way that strong nonlinear regions. This implies tessellations with a huge number of simplices. Therefore condition C1 is not satisfied.
- **Moving mesh methods**. These algorithms start with an initial mesh and move the vertices with modification of the combinatorial structure of the initial grid [18] or keeping such structure [13]. Although these algorithm are adaptive, the number of cells in the initial partition is taken heuristically. Therefore, these methods do not guarantee uniform convergence.

A procedure for solving the UCPA problem with tessellations of moderate size, consists of using **tree approximation** or (**adaptive splitting**) algorithms. These algorithms consider:

- A nonnegative function $U(T)$ which gives an upper bound of $\| f - I_T(f) \|_{C(T)}$.
- An initial partition Δ_1 of I.
- An algorithm R for dividing any simplex into two or more simplices.

Given $\epsilon > 0$ we say that a simplex T is *good* if $U(T) < \epsilon$; otherwise T is called *bad*.

The algorithm proceeds in the following manner:

- The good simplices in the initial partition Δ_1 are put into the set $\mathcal{G}_1$ and the bad simplices are put into the set $\mathcal{B}_1$.
- At each step j we have a set of good simplices $\mathcal{G}_j$ and a set of bad simplices $\mathcal{B}_j$. We divide the simplices in $\mathcal{B}_j$ into their children according to R and test whether these

children T are good or bad to obtain the sets $\mathcal{G}_{j+1}$ and $\mathcal{B}_{j+1}$.
– The algorithm stops if $\mathcal{B}_j = \varnothing$. Then $\mathcal{G}_j$ is the searched partition Δ. If the stopping criterion is not satisfied, the process is repeated.

This algorithm always converges due to Proposition 2.b.

Piecewise polynomial approximation by adaptive splitting was introduced by Birman and Solomjak [3] in the context of Sobolev spaces and polynomials of degree n. This method has been studied from a theoretical point of view in, for example, [2,3,7–9]. In practice, the major difficulty resides in computing the local error estimate

$$E(T) = \max_{\mathbf{x}\in T} |f(\mathbf{x}) - I_T(f)(\mathbf{x})|. \tag{1}$$

In this paper we give efficient algorithms for finding tight upper bounds on $E(T)$. In this way tree approximation can be implemented for obtaining uniform continuous PA-approximants (condition C2). The resulting data structures are binary search trees. For binary search trees the *average* depth over all nodes in the tree is $O(\log(M))$ where M is the total number of nodes [1, p. 105]. Therefore, the *average* retrieval time will be also $O(\log(M))$ (condition C3).

This paper is organized as follows, Sect. 2 gathers analytical upper bounds on the local error function $E(T)$. Section 3 provides two methods based on global optimization techniques to obtain tight upper bounds on $E(T)$. Section 4 outlines the algorithm to generate the approximation tree. Section 5 shows numerical experiments using as target function the distance from a point to a geometric figure. Section 6 provides some concluding remarks.

2 Upper bounds of analytical type

From now on, $\| \cdot \|$ denotes the Euclidean norm in $\mathbb{R}^d$. In the case of smooth functions, we have the following upper bound on $E(T)$

Theorem 1 ([30,32]) *Let T be a d-simplex in $\mathbb{R}^d$. Let $f \in C^2(T)$, then*

$$|f(\mathbf{x}) - I_T(f)(\mathbf{x})| \le \frac{d}{4(d+1)}|T|^2 |f|_{2,\infty,T}, \quad \forall \mathbf{x} \in T, \tag{2}$$

where $|T|$ is the diameter of T and $|f|_{2,\infty,T}$ is the seminorm on $C^2(T)$ defined by

$$|f|_{2,\infty,T} \equiv \| \, |D^2 f| \, \|_{L^\infty(T)},$$

where

$$|D^2 f|(\mathbf{x}) \equiv \sup_{\substack{\mathbf{u}_1,\mathbf{u}_2\in\mathbb{R}^2 \\ \|\mathbf{u}_i\|\le 1}} |D_{\mathbf{u}_1} D_{\mathbf{u}_2} f(\mathbf{x})|.$$

The main drawback of the Subbotin–Waldron inequality (2) is that it can be applied only to C^2 functions. Nevertheless, most functions which appear in practice are non-smooth. For example, the distance to a convex geometric figure (test functions used in this paper), is not differentiable at the boundary of the figure. In the case of polygons and polyhedra, it is not of class C^2 on some regions outside the body.

The above theorem can be applied with some modifications to the distance from a point to a d-sphere, which is of class C^2 on regions that do not contain the boundary. Furthermore, its second order partial derivatives can be computed easily. The following result shows how to apply the above inequality in this case.

Proposition 3 (a) *Let c be a circle centered at the origin with radius R. Let $d_c(\mathbf{x})$ be the distance from a point $\mathbf{x}$ to c. If $T = \langle \mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2 \rangle$, where $\| \mathbf{v}_i \| > R$, $i = 0, 1, 2$, then*

$$|d_c|_{2,\infty,T} \le \frac{1}{r_o},$$

where r_o is the distance from the origin to the triangle T.

(b) *Let s be a 3-sphere centered at the origin with radius R. Let $d_s(\mathbf{x})$ be the distance from a point $\mathbf{x}$ to s. If $T = \langle \mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \rangle$, where $\| \mathbf{v}_i \| > R$, $i = 0, 1, 2, 3$, then*

$$|d_s|_{2,\infty,T} \le \frac{\sqrt{2}}{r_o},$$

where r_o is the distance from the origin to the tetrahedron T.

Proof We give a proof of the last inequality. Part (a) can be proved in a similar way. We first recall the definition of the Frobenius norm of a $m \times n$ matrix A

$$\| A \|_F \equiv \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{1/2}.$$

$D_{\mathbf{v}} D_{\mathbf{u}} \mathrm{d}_B(\mathbf{x}) = \mathbf{u}^t H \mathbf{v}$, where H is the Hessian matriz of d_B. If $\| \mathbf{u} \| \le 1$ and $\| \mathbf{v} \| \le 1$, then

$$|D_{\mathbf{v}} D_{\mathbf{u}} \mathrm{d}_B(\mathbf{x})| \le \| \mathbf{u} \| \| H \|_F \| \mathbf{v} \| \le \| H \|_F .$$

Since $\mathrm{d}_s = r - R$, where $r = (\sum_{i=1}^{3} x_i^2)^{1/2}$, we have

$$D_{ii} \mathrm{d}_s = \frac{x_j^2 + x_k^2}{r^3}, \quad j, k \ne i, \quad i = 1, 2, 3,$$

and

$$D_{ij} \mathrm{d}_s = -\frac{x_i x_j}{r^3}, \quad i \ne j, \quad i, j = 1, 2, 3,$$

then

$$\| H \|_F = \frac{\sqrt{2}}{r}.$$

□

Even in the case of a d-sphere, when a simplex T intersects its boundary, the distance is non-differentiable on T. Therefore, we need upper bounds on $E(T)$ in the case of non-smooth functions. The following inequality considers the case where f is convex.

Proposition 4 *Let f be a convex, continuous and non negative function defined on $T = \langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$. If we define*

$$g(\mathbf{x}) \equiv I_T(f)(\mathbf{x}) - f(\mathbf{x})$$

and

$$M \equiv \max_{0 \le i \le d} \{f(\mathbf{v}_i)\},$$

then

$$0 \le g(\mathbf{x}) \le M, \quad \forall \mathbf{x} \in T. \tag{3}$$

Proof The left-hand side of inequality (3) is a consequence of the convexity of f. Since f is non-negative, we have

$$-f(\mathbf{x}) \le 0, \quad \mathbf{x} \in T, \tag{4}$$

since $I_T(f)$ is convex it attains its maximum value at a vertex of T

$$I_T(f)(\mathbf{x}) \le M. \tag{5}$$

If we add (4) and (5), the result follows. □

The above inequality is effective only for small simplices near (or intersecting) the boundary of the d-sphere. It will be used to complement the Subbotin–Waldron inequality.

Consider now the case of Lipschitz functions. Let S be a d-simplex in $\mathbb{R}^d$, let $L \in [0, \infty)$. $l : S \to \mathbb{R}$ is L-Lipschitz on S if

$$|l(\mathbf{x}) - l(\mathbf{x}')| \le L \parallel \mathbf{x} - \mathbf{x}' \parallel, \quad \mathbf{x}, \mathbf{x}' \in S. \tag{6}$$

l is Lipschitz on S if

$$\mathrm{Lip}(l) \equiv \inf\{L \in [0, \infty) : l \; is \; L - Lipschitz \; on \; S\} < \infty.$$

The function distance d to an arbitrary nonempty, closed set in $\mathbb{R}^d$ is Lipschitz on $\mathbb{R}^d$ with Lip(d) = 1 [6, p. 50].

We have the following result

Proposition 5 ([26]) *Let $S = \langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$ be a d-simplex in $\mathbb{R}^d$ and let l be L-Lipschitz on S, then*

$$l(\mathbf{x}) \le \frac{1}{d+1} \times \left(\sum_{j=0}^{d} l(\mathbf{v}_j) + L \max_{0 \le i \le d} \sum_{j=0}^{d} \parallel \mathbf{v}_i - \mathbf{v}_j \parallel \right), \quad \forall \mathbf{x} \in S. \tag{7}$$

If f is a L-Lipschitz function, we can apply this inequality to problem (1). Define $h(\mathbf{x}) \equiv |f(\mathbf{x}) - I_S(f)(\mathbf{x})|$. Since $I_S(f)(\mathbf{x}) = \sum_{i=1}^{d} \alpha_i x_i + \alpha_0$, $I_S(f)$ is a Lipschitz function

$$|I_S(f)(\mathbf{x}) - I_S(f)(\mathbf{x}')| = \left| \sum_{i=1}^{d} \alpha_i (x_i - x_i') \right| \le \left(\sum_{i=1}^{d} \alpha_i^2 \right)^{1/2} \parallel \mathbf{x} - \mathbf{x}' \parallel . \tag{8}$$

From (6) and (8) $h(\mathbf{x})$ is L'-Lipschitz on S with $L' \equiv L + (\sum_{i=1}^{d} \alpha_i^2)^{1/2}$. From (7) we have

$$h(\mathbf{x}) \le \frac{1}{d+1} \times \left(\sum_{j=0}^{d} h(\mathbf{v}_j) + L' \max_{0 \le i \le d} \sum_{j=0}^{d} \parallel \mathbf{v}_i - \mathbf{v}_j \parallel \right), \quad \mathbf{x} \in S.$$

Define

$$\mu(S) \equiv \frac{1}{d+1} \left(\sum_{j=0}^{d} h(\mathbf{v}_j) + L' \max_{0 \le i \le d} \sum_{j=0}^{d} \parallel \mathbf{v}_i - \mathbf{v}_j \parallel \right). \tag{9}$$

$\mu(S)$ is never a tight bound because it is only of order $|S|$. Therefore, its direct use is inefficient. Further on, it will be employed as a basic tool for subdivision algorithms.

From the above results, it seems clear that in the case of non-smooth continuous functions it is necessary to design new procedures for obtaining tight upper bounds on $E(T)$. In the next section we provide several such methods based on global optimization techniques.

3 Tight upper bounds by global optimization techniques

In this section we introduce techniques for finding upper bounds on $E(T)$ lower than those provided by the analytical methods described in the previous section.

3.1 Truncated simplicial branch and bound method

Branch and bound is a major tool for solving hard optimization problems. A branch and bound algorithm consists of several steps:

- *Branching*: Creation of one or more new subproblems from an existing subproblem.
- *Bounding*: This process is applied to each subproblem.
- *Prunning*: Elimination of the subproblems which cannot lead to an optimal solution.
- *Retracting*: Moving backward from a higher level to a lower level.

The branch and bound approach to optimization problems was developed initially within the context of solving linear programs with integer decision variables [19]. The first application of this methodology to the optimization of multivariate Lipschitz functions appeared in [25].

Below we describe the *truncated branch and bound algorithm* which is a slight modification of the *classical simplicial branch and bound method* [26].

Let $h(\mathbf{x})$ be the function defined in Sect. 2 above. In order to describe the algorithm, we need some previous definitions.

We denote by $v(S)$ the set of vertices of any d-simplex S. At each iteration cycle $k = 1, 2, 3, \ldots$ we have the following elements

- A current list of adaptively generated, disjoint (with respect to their interior) subsimplices of T

$$\mathcal{P}_k = \{T_1, \ldots, T_i\}, \quad i = i(k),$$

that jointly cover the actual search region of interest.
- An estimate of the global optimizer $z_k^* = \max_{\mathbf{v}\in\mathcal{V}} h(\mathbf{v})$ where $\mathcal{V} = \cup_{j=1}^{i} v(T_j)$. z_k^* is a lower bound on $E(T)$.
- An upper bound $\mu(T_j)$, obtained from (9), for each $T_j \in \mathcal{P}_k$.

Then we can state

Truncated simplicial branch and bound method

Let $T = \langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$ be the d-simplex under consideration in the process of tessellation.

Read $Niter$ (Number of iterations).
Make $\mathcal{P}_0 = T$ and $z_0^* = \max_{0\le j\le d}\{h(\mathbf{v}_j)\}$.
Compute $\mu(T)$ (*bounding*).
for $k = 0, Niter$
{
-Build the set $\mathcal{R}_k = \mathcal{P}_k - \{T_j : \mu(T_j) < z_k^*\}$ (*prunning*).
-Select a subsimplex $T_t \in \mathcal{R}_k$ such that $\mu(T_t) = \max_{T_r\in\mathcal{R}_k} \mu(T_r)$ (*retracting*).
-Divide T_t into two subsimplices by bisecting the largest edge of T_t. $T_t = \cup_{j=1}^{2} T_{t_j}$ (*branching*).
-Build the set $\mathcal{P}_{k+1} = (\mathcal{R}_k - T_t) \cup (\cup_{j=1}^{2} T_{t_j})$.
-Evaluate h on the new vertices and derive a new estimate of the global optimizer z_{k+1}^*.
-Compute $\mu(T_j)$ for each $T_j \in \mathcal{P}_{k+1}$ (*bounding*).
}
write: An upper bound of h on T is
$\mu = \max_{T_j\in\mathcal{P}_{Niter+1}} \mu(T_j)$.
end

Note. The classical simplicial branch and bound algorithm tries to find the maximum of h within an error ϵ.

The **for** loop is replaced by
do {
Block of sentences equal to those
included in the above **for** loop.
Make $\mu = \max_{T_j\in\mathcal{P}_{k+1}} \mu(T_j)$.
} **while**$(\mu - z_{k+1}^* > \epsilon)$
and the last sentence by **write** The maximum is z_{k+1}^*, respectively.

The classical branch and bound is not suitable for obtaining upper bounds on $E(T)$ because the number of iterations is unknown a priori, and can be very large for a given ϵ. We call TSBBA the adaptive algorithm based on the truncated simplicial branch and bound method.

3.2 Successive subdivision D.C. method

If $f(\mathbf{x})$ is a convex function defined on a d-simplex T then

$$f(\mathbf{x}) - I_T(f)(\mathbf{x}) \le 0, \quad \forall \mathbf{x} \in T.$$

Therefore, $h(\mathbf{x}) = I_T(f)(\mathbf{x}) - f(\mathbf{x}) \ge 0$ and (1) can be written as

$$E(T) = \max_{\mathbf{x}\in T} h(\mathbf{x}) = -\min_{\mathbf{x}\in T}(f(\mathbf{x}) - I_T(f)(\mathbf{x})).$$

The problem

$$\min_{\mathbf{x}\in T}(f(\mathbf{x}) - I_T(f)(\mathbf{x})). \tag{10}$$

is a difference of two convex functions (D.C.) optimization problem.

Most algorithms for D.C. optimization are iterative (see, e.g., [31] for a survey), therefore, they are not suited to be included into an adaptive splitting algorithm. In this section we use some preparatory results of an iterative algorithm (prismatic branch and bound [16]) for obtaining methods easy to implement within an adaptive splitting approach.

By introducing an additional variable t, (10) can be converted into the equivalent problem

$$\min_{\substack{\mathbf{x}\in T \\ f(\mathbf{x})-t\le 0}} (t - I_T(f)(\mathbf{x})). \tag{11}$$

with concave objective function and convex feasible set.

In effect, it is clear that

$$\min_{\substack{\mathbf{x}\in T \\ f(\mathbf{x})-t\le 0}} (t - I_T(f)(\mathbf{x})) = \min_{\mathbf{x}\in T}(f(\mathbf{x}) - I_T(f)(\mathbf{x})).$$

From this equality the following result is obtained

Proposition 6 *If* $\mathbf{x}^*$ *is a solution of* (10) *and* $t^* = f(\mathbf{x}^*)$, *then* $(\mathbf{x}^*, t^*)$ *is a solution of* (11). *Conversely, if* $(\mathbf{x}^*, t^*)$ *is a solution of* (11), *then* $\mathbf{x}^*$ *is a solution of* (10).

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function and let $\mathbf{z} \in \mathbb{R}^d$, the subdifferential of f at $\mathbf{z}$ is given by

$$\partial f(\mathbf{z}) \equiv \{\mathbf{v} \in \mathbb{R}^d : f(\mathbf{z}) + (\mathbf{x} - \mathbf{z}).\mathbf{v} \leq f(\mathbf{x})\}, \quad \forall \mathbf{x} \in \mathbb{R}^d.$$

$\mathbf{s} \in \partial f(\mathbf{z})$ is called a subgradient of f at $\mathbf{z}$. We use the following definition [16, p. 220].

Let f be a convex function on $\mathbb{R}^d$ and let M be a finite set of points in $\mathbb{R}^d$. Then the function

$$f_M(\mathbf{x}) \equiv \max_{\mathbf{z} \in M}((\mathbf{x} - \mathbf{z}).\mathbf{s} + f(\mathbf{z})),$$

where $\mathbf{s}$ is a subgradient of f at $\mathbf{z}$, has the following properties

1. $f_M(\mathbf{x})$ is piecewise affine and convex on $\mathbb{R}^d$.
2. $f_M(\mathbf{x}) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^d$.
3. $f_M(\mathbf{z}) = f(\mathbf{z})$ for all $\mathbf{z} \in M$.
4. If $M_1 \subseteq M_2$, and we take the same subgradient in points in $M_1 \cap M_2$ as in points in M_1, then $f_{M_1}(\mathbf{x}) \leq f_{M_2}(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^d$.

(11) can be written as

$$\min_{(\mathbf{x},t) \in F}(t - I_T(f)(\mathbf{x})),$$

where

$$F \equiv \{(\mathbf{x}, t) \in \mathbb{R}^{d+1} : \mathbf{x} \in T, \quad f(\mathbf{x}) - t \leq 0\}.$$

Define

$$P_M \equiv \{(\mathbf{x}, t) \in \mathbb{R}^{d+1} : \mathbf{x} \in T, \quad f_M(\mathbf{x}) - t \leq 0\},$$

where M is a finite subset of T. Then $P_M \supseteq F$, and

$$\min_{(\mathbf{x},t) \in P_M}(t - I_T(f)(\mathbf{x})) \leq \min_{(\mathbf{x},t) \in F}(t - I_T(f)(\mathbf{x})),$$

hence

$$-\min_{(\mathbf{x},t) \in P_M}(t - I_T(f)(\mathbf{x})) \tag{12}$$

is an upper bound on $E(T)$. If we consider two finite sets M_1 and M_2 and f_{M_1} and f_{M_2} satisfying the conditions in Property 4, we have

$$\min_{(\mathbf{x},t) \in P_{M_1}}(t - I_T(f)(\mathbf{x})) \leq \min_{(\mathbf{x},t) \in P_{M_2}}(t - I_T(f)(\mathbf{x})),$$

This inequality gives an incremental procedure for improving the bound, but it depends on the position of the initial and successive points and on the choice of the subgradients. Even in the best case, the resulting procedure would be iterative.

An alternative heuristic approach consists of considering a set M having many points uniformly distributed on T. A drawback of this method is the need of solving LP (linear programming) problems (12) having a large set of linear restrictions. This can lead to numerical difficulties when standard algorithms, such as the simplex method, are used for solving them (In an adaptive splitting procedure such LP problem must be solved many times).

Further on, we detail the *successive subdivision method*, that avoids large LP problems. Below we prove that when f is the function $\mathrm{d}_{\mathcal{P}}$ (distance to a polytope $\mathcal{P}$ in $\mathbb{R}^d$ $d = 2, 3$), we can compute effectively P_M.

Let d_C the function distance to a closed, convex set C in $\mathbb{R}^d$. The subdifferential of d_C is given by

$$\partial \mathrm{d}_C(\mathbf{z}) = \begin{cases} N_C(\mathbf{z}) \cap B_1 & \text{if } \mathbf{z} \in C \\ (\mathbf{z} - \pi_C(\mathbf{z}))/ \parallel \mathbf{z} - \pi_C(\mathbf{z}) \parallel & \text{if } \mathbf{z} \notin C \end{cases}$$

where $\pi_C(\mathbf{z})$ denotes the projection of $\mathbf{z}$ on C, $N_C(\mathbf{z})$ the outward normal cone of C at $\mathbf{z}$ and B_1 the Euclidean unit ball centered at the origin [15, p. 259].

If C is the convex polyhedron defined by

$$\mathcal{P} \equiv \{\mathbf{x} \in \mathbb{R}^3 : \mathbf{a}_j.\mathbf{x} \leq b_j, \ j = 1, \ldots, NC\}$$

and we define the set of active constraints at $\mathbf{z}$ by

$$J(\mathbf{z}) \equiv \{j : \mathbf{a}_j.\mathbf{z} = b_j\},$$

then

$$N_{\mathcal{P}}(\mathbf{z}) = \left\{ \sum_{j \in J(\mathbf{z})} \alpha_j \mathbf{a}_j : \alpha_j \geq 0 \right\},$$

see for example, [15, p. 138]. From the above expressions we can calculate subgradients of $d_{\mathcal{P}}$ at $\mathbf{z}$. We consider $d = 3$ (the treatment of $d = 2$ is similar). If T is the tetrahedron $\langle \mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \rangle$, f is the distance to $\mathcal{P}$, represented by $\mathrm{d}_{\mathcal{P}}$ and M is the set of vertices of T, we can give a precise description of P_M.

Every face of the tetrahedron can be represented by the 3-combination $\{i, j, k\}$ of $\{0, 1, 2, 3\}$.

Let τ denote the set of 3-combinations of $\{0, 1, 2, 3\}$.

The equation of the plane containing the face $\{i, j, k\}$ is given by

$$\mathbf{n}_{ijk}.(\mathbf{x} - \mathbf{v}_i) = 0,$$

where

$$\mathbf{n}_{ijk} \equiv (\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i).$$

Define

$$E_{ijk} \equiv \mathbf{n}_{ijk}.(\mathbf{v}_l - \mathbf{v}_i) \quad \text{where } l \notin \{i, j, k\},$$
$$e_{ijk} \equiv E_{ijk}/|E_{ijk}|.$$

Points $\mathbf{x}$ inside T satisfy

$$\mathbf{n}_{ijk}.(\mathbf{x} - \mathbf{v}_i)e_{ijk} \geq 0 \quad \text{for all } \{ijk\} \in \tau.$$

Consequently, P_M is the set of points $(\mathbf{x}, t) \in \mathbb{R}^4$ such that

$$(\mathbf{x} - \mathbf{v}_i).\mathrm{sd}_{\mathcal{P}}(\mathbf{v}_i) + \mathrm{d}_{\mathcal{P}}(\mathbf{v}_i) - t \leq 0, \quad i = 0, 1, 2, 3,$$
$$\mathbf{n}_{ijk}.(\mathbf{x} - \mathbf{v}_i)e_{ijk} \geq 0, \quad \{ijk\} \in \tau.$$

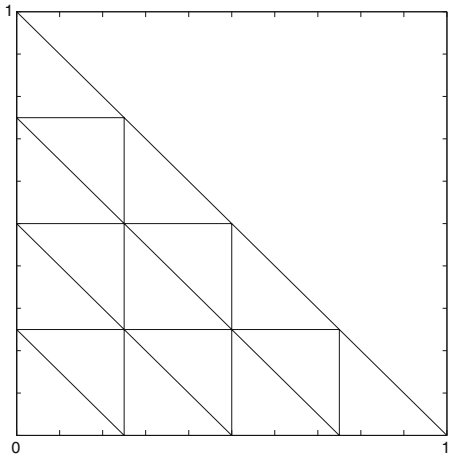

Fig. 1 2D mesh for $m = 4$

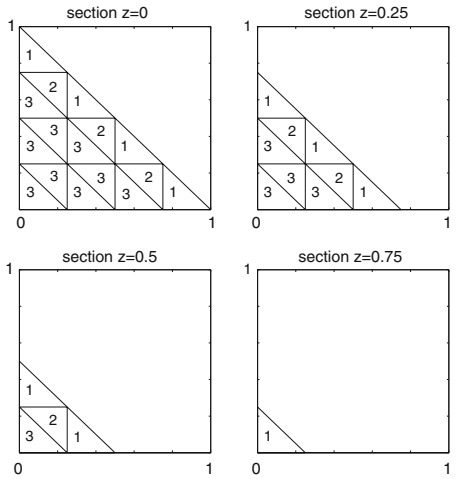

Fig. 2 Generation of subtetrahedra for $m = 4$

Therefore, (12) is a linear programming problem with 6 constraints in 2D and 8 constraints in 3D. It can be solved by several procedures, for example, the simplex algorithm.

The idea behind the *successive subdivision D.C. method* is the following. For a given d-simplex T, we can reduce the upper bound on $E(T)$ by subdividing T into a set of subsimplices and applying the above method to each one of them. Then, an upper bound on T will be the maximum of the upper bounds found for the subsimplices. If we repeat this process over a fixed number of partitions of T, we can choose the upper bound on T as the minimum of the bounds obtained for each partition.

To express concretely the algorithm, we detail the subdivision process used for 2-simplices (triangles) and 3-simplices (tetrahedra).

The subdivision process is made by means of the algebraic (transport) method.

In the plane we triangulate the standard triangle

$$S = \langle (0, 0), (1, 0), (0, 1) \rangle,$$

in the following way

- Every edge is subdivided into m equal parts.
- The points of the subdivision are connected by straight lines as shown in Fig. 1.

In this way, we subdivide the standard triangle into m^2 subtriangles. Then, given an arbitrary triangle $T = \langle \mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2 \rangle$ we export the mesh of S into T by means of the application

$$\mathbf{w} = (1 - x - y)\mathbf{v}_0 + x\mathbf{v}_1 + y\mathbf{v}_2.$$

In three dimensions, we subdivide the standard tetrahedron

$$S = \langle (0, 0, 0), (1, 0, 0), (0, 1, 0), (0, 0, 1) \rangle$$

into subtetrahedra in the following way

- Every edge is subdivided into m equal parts.
- We intersect S with the planes $x = k/m$, $y = k/m$, $z = k/m$ and $x + y = k/m$, $k = 0, \ldots, m$.
- Every triangle in the sections $z = k/m$, $k = 0, \ldots, m$ (Fig. 2) is used to generate 1, 2 or 3 subtetrahedra according to the method described in [11].

In this way, we subdivide the standard tetrahedron into m^3 subtetrahedra. Then, given an arbitrary tetrahedron $T = \langle \mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3 \rangle$ we export the mesh of S into T by means of the application

$$\mathbf{w} = (1 - x - y - z)\mathbf{v}_0 + x\mathbf{v}_1 + y\mathbf{v}_2 + z\mathbf{v}_3.$$

Let $T = \langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$ be the simplex under consideration in the process of tessellation.

Denote by

$$T_k^m \equiv \langle \mathbf{v}_{k0}^m, \mathbf{v}_{k1}^m, \ldots, \mathbf{v}_{kd}^m \rangle, \quad 1 \le k \le m^d,$$

the elements of the mesh with m subdivisions by edge. Define P_k^m as the set

$$\{(\mathbf{x}, t) \in \mathbb{R}^{d+1} : \mathbf{x} \in T_k^m \text{ and } (\mathbf{x} - \mathbf{v}_{ki}^m) s \mathrm{d}_{\mathcal{P}}(\mathbf{v}_{ki}^m) + \mathrm{d}_{\mathcal{P}}(\mathbf{v}_{ki}^m) - t \le 0, \quad i = 0, \ldots, d\}.$$

Using the above notation and denoting by ub the searched upper bound, we can state

Successive subdivision difference of two convex functions method

Read N (Maximum number of subdivisions of the edges)
$ub = 10^6$;
for $m = 2, N$
{
  $ub^m = 0$;
  for $k = 1, m^d$
  {
    $ub_k^m = -\min_{(\mathbf{x},t)\in P_k^m}(t - a_{T_k^m}(f)(\mathbf{x}))$
    if $(ub_k^m > ub^m)$ $ub^m = ub_k^m$;
  }
  if $(ub^m < ub)$ $ub = ub^m$;
}
end

This algorithm explores uniformly the domain and it only needs to solve a small LP problem of fixed size at each iteration. We call SSDCA the adaptive algorithm based on the successive subdivision difference of two convex functions method.

4 Algorithm to generate the approximation tree

We assume that the initial domain $\mathcal{D}$ is the union of several d-simplices with pairwise disjoint interiors. The *adaptive splitting* algorithm (Sect. 1) is applied to each one of them consecutively. The algorithm splits recursively each subsimplex into two subsimplices by dividing its largest edge according to the following method. Given two positive parameters ϵ and ML, the *split criterion* is defined as

$$SPC(T) \equiv \{UB(E(T)) > \epsilon \text{ and } MAXL > ML\},$$

where $UB(E(T))$ is the upper bound computed according to the methods specified in the above sections and $MAXL$ denotes the length of the largest edge of the simplex. We impose a minimum value ML for the length of the subsimplices edges. In the experiments (Sect. 5), ML has been a very small number and it has never been reached.

The resulting data structure is a binary search tree. A C-like pseudocode of the algorithm to generate the approximation tree can be found in [23].

5 Computer experiments

In this section we test the above methods with the problem of approximating the function Euclidean distance to a geometric body. Its fast computation is necessary in robotic applications (obstacle avoidance, path planning, etc) and in VR-simulations. In pattern recognition this function allows to compute the Hausdorff distance between polytopes [22] (Hausdorff distance is a similarity measure between bodies). The experimental environment has been the following:

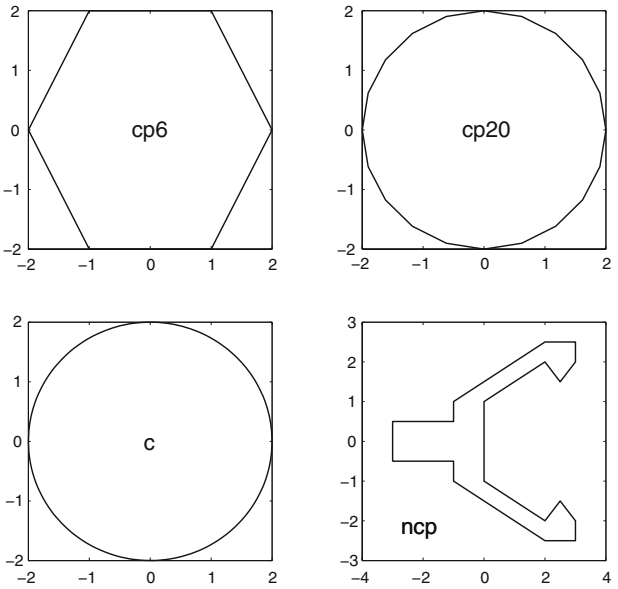

Fig. 3 2D experiments

- CPU Pentium IV 2 GHz.
- RAM 1 GB.
- Operating system Windows XP.
- Running software Microsoft Visual C 6.0.

5.1 Experiments in two dimensions

The domain of the functions to be approximated has been the square $[-4, 4]^2$ containing each figure. This square is further divided into two triangles by its diagonal. Every algorithm has been applied to each triangle consecutively.

2D experiments are shown in Fig. 3. We use the following notation

- cp6: convex hexagon.
- cp20: convex polygon with 20 sides.
- c: circle of radius 2.
- ncp: nonconvex polygon with 14 sides.

In two dimensions we have used the following algorithms

- TSBBA: algorithm based on the truncated simplicial branch and bound method (Sect. 3).
- SSDCA: algorithm based on the successive subdivision difference of convex functions method (Sect. 3).
- SUWA1 and SUWA2: algorithms based on the Subbotin–Waldron inequality (2).

All these algorithms use the scheme for tree generation given in Sect. 4. TSBBA uses a doubly linked list for implementig the branch and bound process. SSDCA solves the linear programming problem (12) by means of the two-phase simplex method (see, e.g. [5]). SSDCA has been applied with maximum number of subdivisions of the edges $N = 5$ in all 2D experiments. The distance from a point to a convex polygon has been computed using the LSABF algorithm [21].

The algorithms have been implemented in different programs having the following supplementary features. Define the average error

$$\bar{E}(T) \equiv \frac{\int_T |f(\mathbf{x}) - I_T(f)(\mathbf{x})| d\mathbf{x}}{\mathrm{v}(T)}, \tag{13}$$

where $\mathrm{v}(T)$ denotes the volume of the d-simplex T. We call $\bar{E}_5(T)$ and $\bar{E}_7(T)$ the result of evaluating (13) by means of the Grundmann and Möller's cubature rules of degree 5 and 7, respectively [14]. The upper bound $UB(E(T))$ in the case of polygons has been determined by the following algorithm

if (A. T is inside the polygon)
  $UB(E(T)) = 0$;
else if (B. T is contained within an outward infinite rectangle normal to some edge)
  $UB(E(T)) = 0$;
else if (C. $|\bar{E}_5(T) - \bar{E}_7(T)| < \frac{E_1}{10}$ && $\bar{E}_7(T) < E_1$)
{
  $UB(E(T))$ is determined by the corresponding *Global optimization algorithm*
}
else
  $UB(E(T)) = 1000$;

In the case of convex figures condition A is a test for checking if the triangle T lies inside the figure ($E(T) = 0$). In the nonconvex case we suppose that the nonconvex polygon is the union of convex polygons with pairwise disjoint interiors (components). We check if the triangle T lies inside a component polygon.

In the case of convex figures condition B is a test for checking if the three vertices of the triangle are projected on the same edge of the polygon. In the nonconvex case we check if the three vertices of the triangle are projected on the same edge of the same component polygon without interference from the remainder components. In both cases the distance is affine over T, therefore, $E(T) = 0$. Condition C is a test for applying global optimization algorithms only to feasible triangles. The first part of this test tries to guarantee the accuracy of the approximated integrals (see [23] for details).The *Global optimization algorithms* are TSBBA and SSDCA in the case of convex polygons and TSBBA in the case of nonconvex polygons.

For the circle, the algorithm for determining $UB(E(T))$ in the case of TSBBA and SSDCA is the same that the previously stated but removing condition B. In the case of applying the Subbotin–Waldron inequality (SUWA1, SUWA2), the procedure is the following

if (A. T is inside the circle)
  $UB(E(T)) = 0$;
else if (B. T intersects the boundary of the circle)
  $UB(E(T))$ is determined by inequality (3)
  in the case of SUWA1 and by formula (9)
  in the case of SUWA2.
else
  $UB(E(T))$ is determined by
  the Subbotin–Waldron inequality (2);

The results obtained by applying TSBBA ($\epsilon = 0.05$, $ML = 0.001$) to the function distance, are shown in Fig. 4 as graphs of number of iterations versus number of triangles in the tessellation. The number of triangles decreases as the number of iterations increases reaching a stationary value which corresponds to the best tessellation which can be obtained using this method.

From now on, we call ϵ precision. In the figures we use the reciprocal of the precision K = $1/\epsilon$. The compared performances of TSBBA, SSDCA applied to the function distance to the polygons and the circle and SUWA1 and SUWA2 applied to the function distance to the circle, are shown in

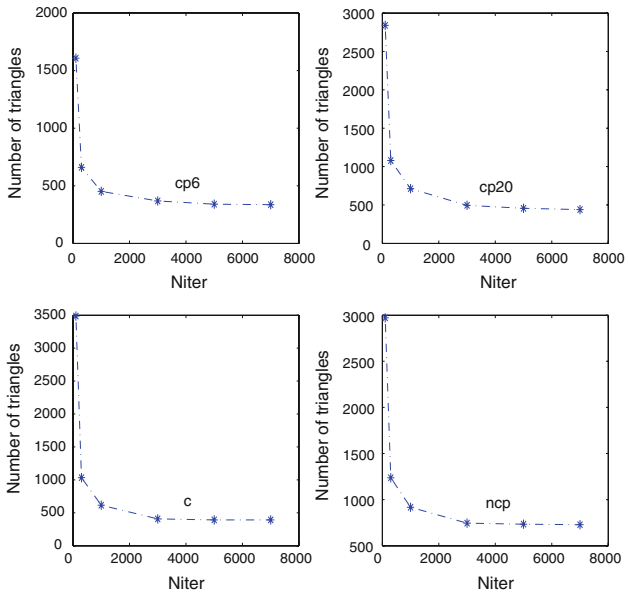

Fig. 4 Stationary behavior of TSBBA

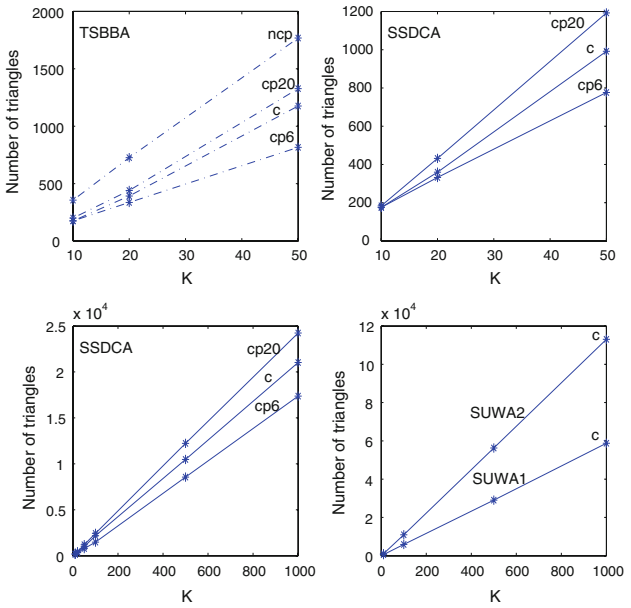

Fig. 5 2D performance of the proposed algorithms

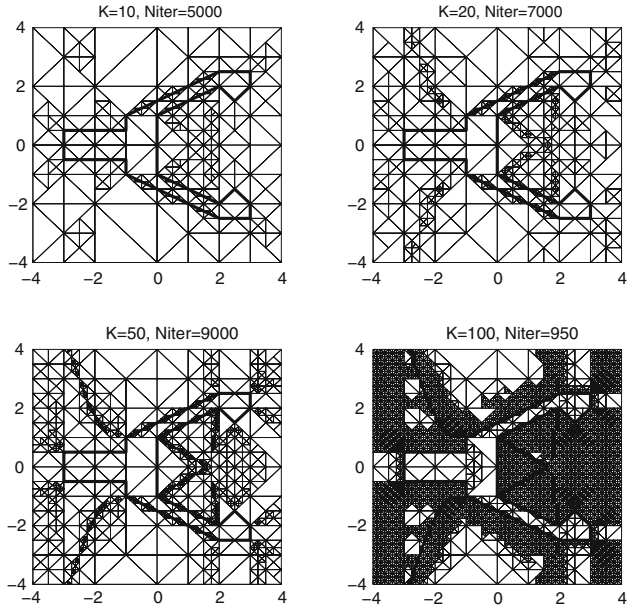

Fig. 6 2D tessellations generated by TSBBA

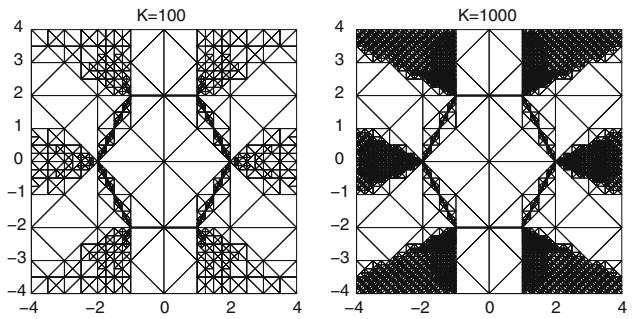

Fig. 7 2D tessellations generated by SSDCA

Table 1 Performance comparison between TSBBA and SSDCA

	cp6	cp20	c
TSBBA	816	1328	1176
SSDCA	776	1192	992

Fig. 5 as graphs of reciprocal of the precision versus the number of triangles in the tessellation. In all cases considered, the behavior has been almost linear. The numbers of triangles corresponding to K = 50 are shown in Table 1. TSBBA has been applied with $Niter = 9000$ in the three examples.

Figure 6 shows the tessellations obtained by TSBBA applied to the distance to ncp, corresponding to different precisions (the data corresponding to K = 100 have been obtained with the non-stationary value $Niter = 950$).

Figure 7 shows the graphs of the tessellations obtained by applying SSDCA to the function distance to the hexagon cp6. The number of triangles represented in these graphs is

- SSDCA with K = 100: 1480 triangles.
- SSDCA with K = 1,000: 17372 triangles.

From Figs. 4, 5 and Table 1 we can conclude that

- SSDCA is the most effective procedure in the case of convex figures (polygons and circle). It obtains greater values of K than TSBBA. Moreover, the number of triangles is lower than that obtained with TSBBA, SUWA1 and SUWA2.
- TSBBA gives less values of K and requires more computational resources than the other methods, but it can be applied for obtaining ϵ-PA tessellations of the function distance to any figure, including non-convex ones.

The results about the runtime of TSBBA, SSDCA, SUWA1 and SUWA2 are shown in Fig. 8 as graphs of reciprocal of the precision versus compute time.

We can conclude from these experiments that

- SUWA1 and SUWA2 are the fastest algorithms.
- SSDCA shows an acceptable compute time.
- TSBBA is the slowest of all algorithms because its stationary behavior is reached after many iterations.

We have studied the increment in the number of triangles in the tessellations as the domain of the function distance

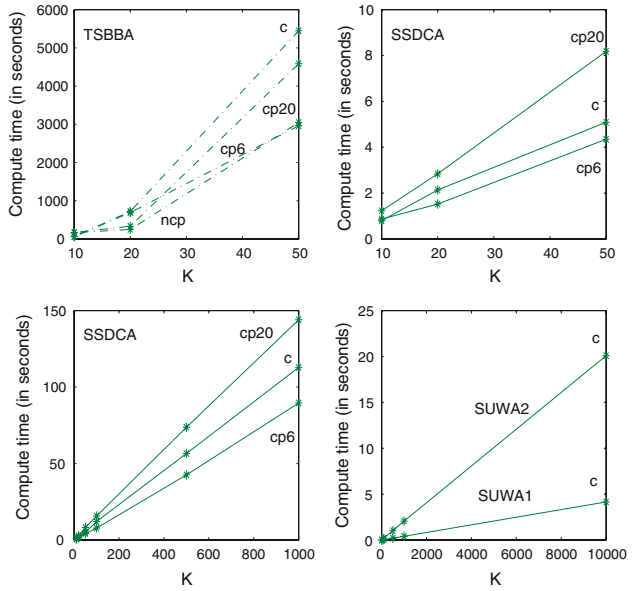

Fig. 8 2D time behavior of the proposed algorithms

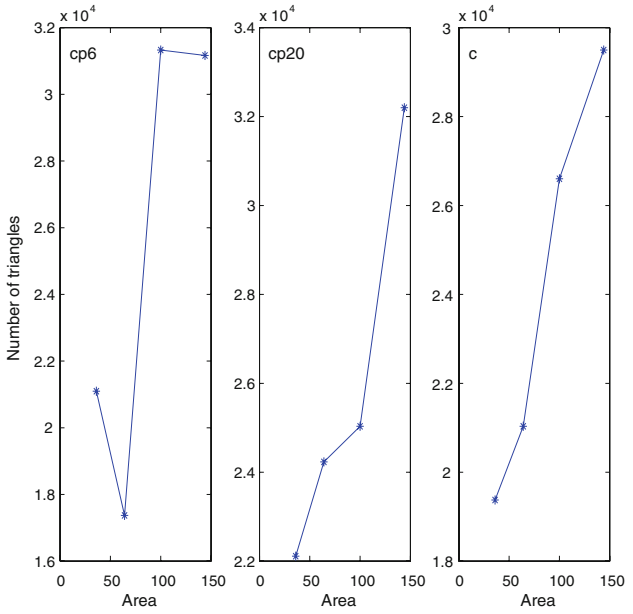

Fig. 9 Domain variation experiments

increases. The results obtained by applying SSDCA to the distances to cp6, cp20 and c are shown in Fig. 9 as graphs of area versus number of triangles. In all cases, the error and minimum edge length are fixed: $\epsilon = 10^{-3}$ and $ML = 10^{-4}$. We have considered the domains: $[-3, 3]^2, [-4, 4]^2, [-5, 5]^2$ and $[-6, 6]^2$. In the case of polygons with a small number of vertices, a little change in the size of the domain implies a big change in the number of triangles in the tessellation. This instability accounts for the results obtained for cp6.

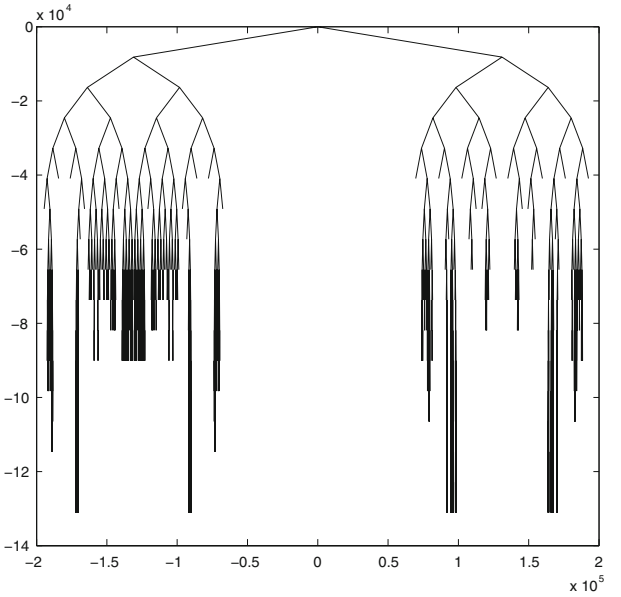

Fig. 10 Approximation tree

Figure 10 shows the approximation tree generated by SSDCA in the case of the distance to the hexagon cp6 with $\epsilon = 0.01$ and $ML = 0.001$ (1480 triangles). The initial domain has been the triangle with vertices $(-4, -4)$, $(4, -4)$ and $(4, 4)$.

5.2 Experiments in three dimensions

In this subsection we consider the function distance to a 3D geometric body. The domain of this function has been the cube $[-4, 4]^3$ containing each of the considered bodies. Due to RAM requirements, we have decomposed the cube into twelve tetrahedra by means of its center and the diagonals of its faces. The algorithms have been applied to each tetrahedron consecutively.

In some 3D experiments we have used ellipsoidal polyhedra, that is, those inscribed in the ellipsoid

$$x^2/a^2 + y^2/b^2 + (z - c)^2/c^2 = 1.$$

The ellipsoid has been subdivided axially (z axis) into m parts and radially (x–y plane) into n parts. The resulting polyhedron has $(m - 2)n + 2$ vertices and $(m - 1)n$ faces. We denote by $\mathcal{E}(n, m, a, b, c)$ any polyhedron of this type.

3D experiments are shown in Fig. 11. We use the following notation

- cpl4: tetrahedron.
- cpl6: cube.
- cpl20: icosahedron.
- s: sphere of radius 1.5 centered at the origin.
- epl1: $\mathcal{E}(10, 10, 1, 1, 1.5)$ (82 vertices and 90 faces).

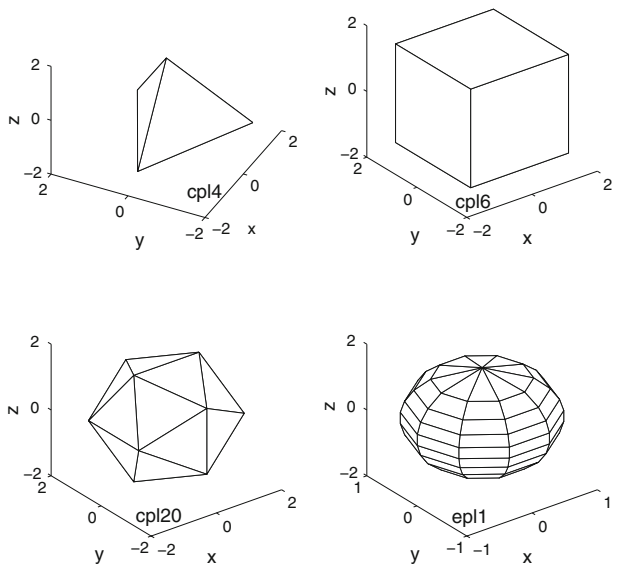

Fig. 11 3D experiments

– epl2: $\mathcal{E}(20, 26, 1, 1, 1.5)$ (482 vertices and 500 faces).
– epl3: $\mathcal{E}(25, 33, 1, 1, 1.5)$ (777 vertices and 800 faces).

The specific features of the bodies with less vertices are described below.

– The vertices of the tetrahedron are

$(1.5, -1.5, -1.5)$	$(1.5, 1.5, -1.5)$
$(-1.5, 0, -1.5)$	$(0, 0, 1.5)$

– The vertices of the cube are

$(1.5, -1.5, -1.5)$	$(1.5, 1.5, -1.5)$
$(-1.5, 1.5, -1.5)$	$(-1.5, -1.5, -1.5)$
$(1.5, -1.5, 1.5)$	$(1.5, 1.5, 1.5)$
$(-1.5, 1.5, 1.5)$	$(-1.5, -1.5, 1.5)$

– The vertices of the icosahedron are

$(1, 0, \tau)$	$(-1, 0, \tau)$	$(0, -\tau, 1)$
$(\tau, -1, 0)$	$(\tau, 1, 0)$	$(0, \tau, 1)$
$(-\tau, -1, 0)$	$(-\tau, 1, 0)$	$(0, \tau, -1)$
$(-1, 0, -\tau)$	$(1, 0, -\tau)$	$(0, -\tau, -1)$

where $\tau = 1.618034$.

We have used SSDCA and SUWA1 adapted to three dimensions. The distances from a point to a polyhedron have been computed using the LSABF algorithm [21]. SSDCA determines $UB(E(T))$ in the case of polyhedra by the following algorithm

if (A. T is inside the polyhedron)
  $UB(E(T)) = 0$;
else if (B. $max(\mathrm{d}_{\mathcal{P}}(\mathbf{a}), \mathrm{d}_{\mathcal{P}}(\mathbf{b}), \mathrm{d}_{\mathcal{P}}(\mathbf{c}), \mathrm{d}_{\mathcal{P}}(\mathbf{d})) < E1$)
  $UB(E(T))$ is determined by inequality (3);
else if (C. T is contained within
an outward infinite prism normal
to some face of the polyhedron)
  $UB(E(T)) = 0$;
else if (D. $|\bar{E}_5(T) - \bar{E}_7(T)| < \frac{E_1}{10}$ && $\bar{E}_7(T) < E_1$)
{
  $UB(E(T))$ is determined by SSDCA;
}
else
  $UB(E(T)) = 1000$;

Condition A is a test for checking if the tetrahedron lies inside the polyhedron ($E(T) = 0$).

The two-phase simplex method can become unbounded if the tetrahedra are small and they lie near the boundary of the polyhedron (or sphere). Condition B is a test for detecting this situation and using inequality (3) instead of SSDCA.

Condition C is a test for checking if the four vertices of the tetrahedron are projected on the same face of the polyhedron (the distance is affine over T, therefore, $E(T) = 0$). Condition D is a test for applying global optimization algorithms only to feasible triangles (see the 2D case).

In the case of the sphere, the algorithm for determining $UB(E(T))$ using SSDCA is identical to the previous one but removing test C. The method used by SUWA1 (Subbotin–Waldron) is the following

if (A. T is inside the sphere)
  $UB(E(T)) = 0$;
else if (B. T intersects the boundary of the sphere)
  $UB(E(T))$ is determined by inequality (3);
else
  $UB(E(T))$ is determined
  by the Subbotin–Waldron inequality (2);

In all 3D experiments we have applied SSDCA with maximum number of subdivisions $N = 1$.

The results obtained by applying SSDCA to the function distance to the 3D bodies defined above are shown as graphs of reciprocal of the precision versus number of tetrahedra and reciprocal of the precision versus compute time in Fig. 12. In the case of elliptic polyhedra, the number of tetrahedra is almost independent of the combinatorial complexity (i.e., the number of vertices, edges and faces) of the polyhedron

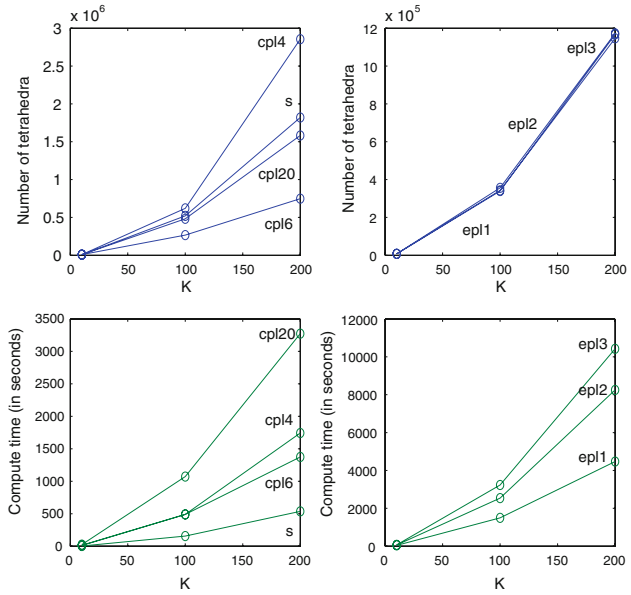

Fig. 12 3D performance of SSDCA

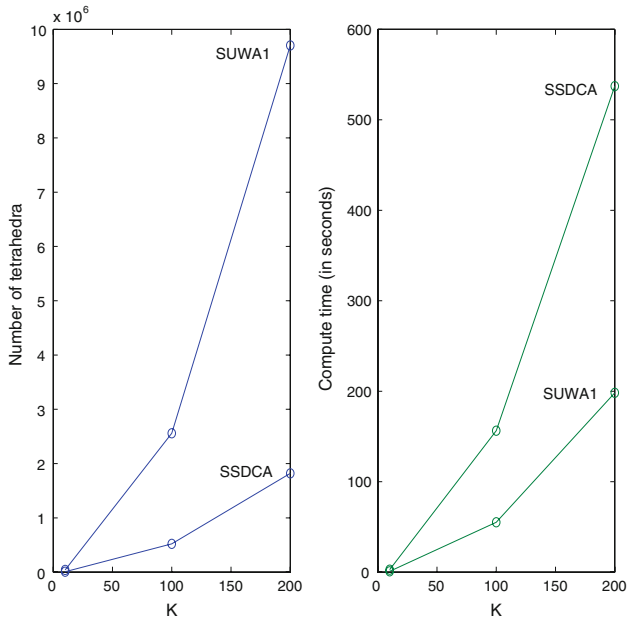

Fig. 13 Comparison between SSDCA and SUWA1 in the case of the function distance to a sphere

considered. The compute time of SSCDA depends on two factors: the number of tetrahedra in the tessellation and the compute time of the distance from a point to the corresponding body (in classical algorithms such as LSABF, this compute time depends on the combinatorial complexity of the polyhedron considered).

The results obtained with SUWA1 applied to the distance to a sphere were compared with those obtained with SSDCA as shown in Fig. 13. SSDCA obtains tessellations with a number of tetrahedra lower than those obtained with SUWA1.

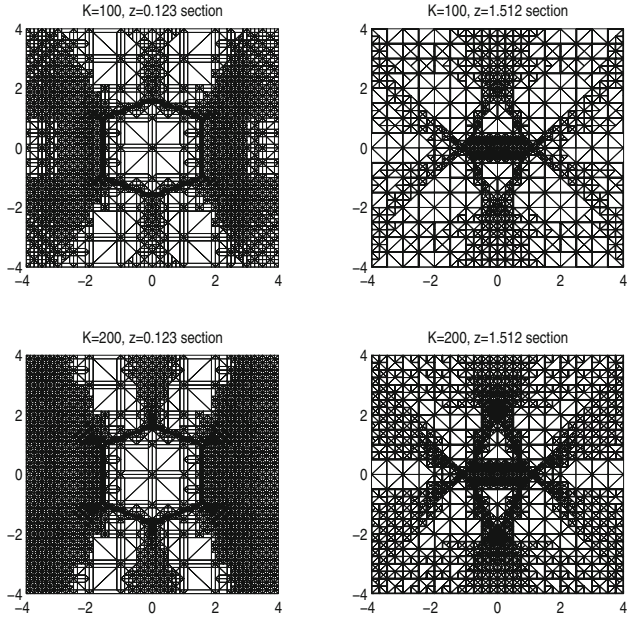

Fig. 14 Intersections of 3D tessellations with *horizontal* planes

The compute time of SSDCA, although greater than the corresponding to SUWA1, is relatively moderate. Therefore, we can conclude that SSDCA is more efficient than SUWA1 in the case of spheres.

We have applied SSDCA to the function distance to the icosahedron cpl20 with different precisions. Figure 14 shows the intersections of the corresponding tessellations with the planes $z = 0.123$ and $z = 1.512$.

We have studied the increment in the number of tetrahedra in the tessellations as the domain of the function distance increases.

The results obtained by applying SSDCA to the distances to the bodies cpl4, cpl20 and epl3 are shown in Fig. 15 as graphs of volume versus number of tetrahedra. In all cases, we have set $\epsilon = 10^{-2}$ and $ML = 10^{-3}$. We have considered the domains: $[-3, 3]^3$, $[-4, 4]^3$, $[-5, 5]^3$ and $[-6, 6]^3$. In the case of polyhedra with a small number of vertices, a little change in the size of the domain implies a big change in the number of tetrahedra in the tessellation. This instability accounts for the results obtained for cpl4.

6 Concluding remarks

A method for approximating functions by piecewise polynomials on partitions of a given domain is nonlinear partitioning. In this procedure the partition is generated by refining some but not all cells. In numerical implementations we need a way to decide when to refine a cell or not. An adaptive algorithm provides such a strategy typically by using local error estimates that monitor the error between the target function

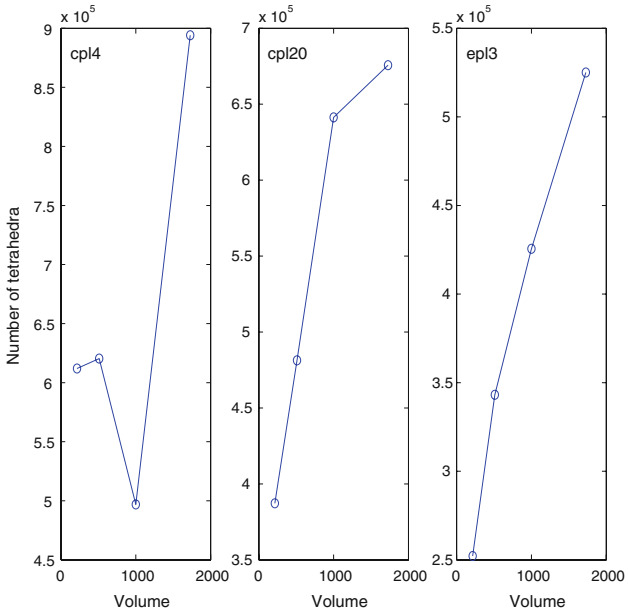

Fig. 15 3D domain variation experiments for SSDCA

and the current approximation on a given cell. Constructing good error estimates in the given numerical setting is usually the main challenge in adaptive approximation [10].

In this paper we present adaptive algorithms for solving the UCPA problem with tessellations of moderate size, in the case of Lipschitz or convex functions. These algorithms compute the local error estimate using two procedures based on global optimization techniques:

- TSBBA performs a truncated branch and bound process for every cell T.
- SSDCA subdivides T into subcells and applies the difference of convex functions minimization method, described in Sect. 3, to each one of them.

Numerical experiments on the function distance d from a point to a geometric body show that TSBBA can approach the distance to convex or nonconvex bodies uniformly but their memory and compute time requirements are large. The possibility of applying analytical bounds based on second derivatives (SUWA) and using the algorithms described in Sect. 3 and inequality (3) for those subsimplices T with $\mathrm{d} \notin C^2(T)$ presents the difficulty of determining such subsimplices. For example, in the case of convex polyhedra, $\mathrm{d} \notin C^2(T)$ when T intersects the surface of separation between the Voronoi zones of the different features (vertices, edges, faces). Therefore, the corresponding intersection tests will become cumbersome for complex polyhedra. In the case of nonconvex polyhedra, $\mathrm{d} \notin C^1(T)$ when T intersects the surface of separation between Voronoi zones of the different features. In this case, Voronoi zones can be very difficult to determine beforehand. In both cases $\mathrm{d} \notin C^1(T)$ when T intersects the boundary of the polyhedron, therefore, tests of intersection with the boundary are also necessary.

SSDCA cannot be applied to approximate the distance to nonconvex bodies, but its RAM memory and compute time requirements are moderate. This allows to obtain precise distances, even in 3D.

When SSDCA is applied to the distance to a sphere, its performance is better than that corresponding to the algorithm based on the Subottin-Waldron inequality (SUWA1).

Figure 12 shows that the number of tetrahedra in the tessellations for the distance to convex polyhedra with very different combinatorial complexity, is almost identical. This implies that the average retrieval time of the corresponding precomputed maps will be similar. By contrast, fast algorithms for computing the distance [20,21] present a compute time increment as the combinatorial complexity of the polyhedra increases.

To sum up, we have presented some algorithms with which one can obtain tree approximants satisfying conditions C1, C2 and C3 in the case of moderately nonlinear Lipschitz or convex functions.

Acknowledgments The authors thank the anonymous referees for their valuable comments and suggestions which improved the original manuscript of this paper.

References

1. Allen, M.: Data Structures and Algorithm Analysis in C. The Benjamin/Cummings, Redwood City (1993)
2. Binev, P., De Vore, R.: Fast computation in adaptive tree approximation. Numer. Math. **97**, 193–217 (2004)
3. Birman, M.S., Solomjak, M.Z.: Piecewise-polynomial approximations of functions of classes W_p^α. Sb. Math. **2**, 295–317 (1967)
4. Bliss, A., Su, F.E.: Lower bounds for simplicial covers and triangulation of cubes. Discrete Comput. Geom. **33**, 669–686 (2005)
5. Chong, E.K.P., Zak, S.H.: An Introduction to Optimization. Wiley, New York (2001)
6. Clarke, F.H.: Optimization and Nonsmooth Analysis. Wiley, New York (1983)
7. Dahmen, W.: Adaptive approximation by multivariate smooth splines. J. Approx. Theory **36**, 119–140 (1982)
8. de Boor, C., Rice, J.R.: An adaptive algorithm for multivariate approximation giving optimal convergence rates. J. Approx. Theory **25**, 337–359 (1979)
9. De Vore, R.A., Yu, X.M.: Degree of adaptive approximation. Math. Comp. **55**, 625–635 (1990)
10. De Vore, R.A.: Optimal computation. In: Proceedings of the International Congress of Mathematicians, Madrid, Spain (2006). Eur. Math. Soc., pp. 187–215 (2007)
11. George, P.L.: Génération automatique de maillages: applications aux methodes d'éléments finis. Masson, Paris (1990)

12. Gamelin, Th.W., Greene, R.E.: Introduction to Topology. Dover, New York (1999)
13. Groff, R.E., Khargonekar, P.D., Koditschek, D.E.: A local convergence proof for the MINVAR algorithm for computing continuous piecewise linear approximations. SIAM J. Numer. Anal. **41**, 983–1007 (2003)
14. Grundmann, A., Möller, H.M.: Invariant integration formulas for the n-simplex by combinatorial methods. SIAM J. Numer. Anal. **15**, 282–290 (1978)
15. Hiriart-Urruty, J.B., Lemaréchal, C.: Convex Analysis and Minimization Algorithms I. Springer, Berlin (1993)
16. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. Kluwer, Dordrecht (1995)
17. Koditschek, D.E., Rimon, E.: Robot navigation functions on manifolds with boundary. Adv. Appl. Math. **11**, 412–442 (1990)
18. Kreylos, O., Hamann, B.: On simulated annealing and the construction of linear spline approximations for scattered data. IEEE Trans. Vis. Comput. Graph. **7**, 17–31 (2001)
19. Land, A.H., Doig, A.G.: An automatic method for solving discrete programming problems. Econometrica **28**, 497–520 (1960)
20. Lin, M.: Efficient collision detection for animation and robotics. Ph.D. Thesis, University of California at Berkeley (1993)
21. Llanas, B., Fernández de Sevilla, M., Feliú, V.: An iterative algorithm for finding a nearest pair of points in two convex subsets of $\mathbb{R}^n$. Comput. Math. Appl. **40**, 971–983 (2000)
22. Llanas, B.: Efficient computation of the Hausdorff distance between polytopes by exterior random covering. Comput. Optim. Appl. **30**, 161–194 (2005)
23. Llanas, B., Sáinz, F.J.: Fast training of neural trees by adaptive splitting based on cubature. Neurocomputing **71**, 3387–3408 (2008)
24. Orden, D., Santos, F.: Asymptotically efficient triangulations of the d-cube. Discrete Comput. Geom. **30**, 509–528 (2003)
25. Pinter, J.D.: Extended univariate algorithms for n-dimensional global optimization. Computing **36**, 91–103 (1986)
26. Pinter, J.D.: Global Optimization in Action. Kluwer, Dordrecht (1996)
27. Petrovic, V., Kuester, F.: Optimized construction of linear approximations to image data. In: Proceedings of the 11th Pacific conference on Computer Graphics and Applications (PG'03), IEEE (2003)
28. Raibert, M.H.: Legged Robots that Balance. MIT Press, Cambridge (1986)
29. Rimon, E., Koditschek, D.E.: The construction of analytic diffeomorphisms for exact robot navigation on star worlds. Trans. Amer. Math. Soc. **327**, 71–115 (1991)
30. Subbotin, Y.N.: Error of the approximation by interpolation polynomials of small degrees in n-simplices. Math. Notes **48**, 1030–1037 (1990)
31. Tuy, H.: D.C. optimization: theory, methods and algorithms. In: Horst, R., Pardalos, P.M. (eds.) Handbook of Global Optimization, pp. 149–216. Kluwer, Dordrecht (1995)
32. Waldron, S.: The error in linear interpolation at the vertices of a simplex. SIAM J. Numer. Anal. **35**, 1191–1200 (1998)