COLORADO SCHOOL OF MINES
EARTH • ENERGY • ENVIRONMENT

Division of Economics and Business
Working Paper Series

Continuous Piecewise Linear δ-Approximations for MINLP Problems. I. Minimal Breakpoint Systems for Univariate Functions

Steffen Rebennack
Josef Kallrath

Working Paper 2012-12
http://econbus.mines.edu/working-papers/wp201212.pdf

Colorado School of Mines
Division of Economics and Business
1500 Illinois Street
Golden, CO 80401

October 2012

Colorado School of Mines
Division of Economics and Business
Working Paper No. **2012-12**
October 2012

Title:
Continuous Piecewise Linear δ-Approximations for MINLP Problems. I. Minimal Breakpoint Systems for Univariate Functions

Author(s):
Steffen Rebennack
Division of Economics and Business
Colorado School of Mines
Golden, CO 80401-1887
srebenna@mines.edu

Josef Kallrath
Department of Astronomy
University of Florida
kallrath@astro.ufl.edu

ABSTRACT

For univariate functions, we compute optimal breakpoint systems subject to the condition that the piecewise linear approximation (or, under- and overestimator) never deviates more than a given δ-tolerance from the original function, over a given finite interval. The linear approximators, under- and overestimators involve shift variables at the breakpoints leading to a small number of breakpoints while still ensuring continuity over the full interval.

We develop two mixed integer non-linear programming models: one which yields the minimal number of breakpoints, and another in which, for a fixed number of breakpoints, their values are computed. Alternatively, we use two heuristics in which we compute the breakpoints subsequently, solving small mixed integer non-linear programming problems, with significantly fewer variables.

The optimal breakpoints for the nonlinear functions can be used in the mixed integer linear programming problem replacement of the original non-linear programming problem or the mixed integer non-linear programming problem. Due to the δ-limited discretization error and the minimal number of breakpoints, the solution of the mixed integer linear programming problem can be obtained in reasonable time and serves a good approximation to the global optimum, which can be fed into a local non-linear programming or mixed integer non-linear programming solver for the final refinement.

Keywords: global optimization, nonlinear programming, mixed integer nonlinear programming, nonconvex optimization, overestimator, underestimator, inner approximation, outer approximation.

1 Introduction

We are interested in the following nonlinear optimization problem:

$$\begin{aligned} \text{NOP}: \min \quad & f(x) && (1)\\ \text{s.t.} \quad & g(x) = 0 && (2)\\ & h(x) \leq 0 && (3)\\ & x \in \mathbb{D} && (4) \end{aligned}$$

with lower and upper bounds, X_- and X_+, on x and

$$\begin{aligned} D_1: \quad & \mathbb{D} := [X_-, X_+]^n \subset \mathbb{R}^n, \text{ or}\\ D_2: \quad & \mathbb{D} := [X_-, X_+]^n \subset \mathbb{R}^{n_1} \times \{0,1\}^{n_2}, \text{ or}\\ D_3: \quad & \mathbb{D} := [X_-, X_+] \subset \mathbb{R}, \text{ or}\\ D_4: \quad & \mathbb{D} := [X_-, X_+]^2 \subset \mathbb{R}^2 \end{aligned}$$

and $f : \mathbb{D} \to \mathbb{R}$, $g : \mathbb{D} \to \mathbb{R}^{m_1}$, $h : \mathbb{D} \to \mathbb{R}^{m_2}$ being continuous functions as well as $n_1 + n_2 = n$. We also allow for special cases when the number of equality constraints, m_1, is zero or the number of inequality constraints, m_2, is zero. We are particularly interested in those cases when problem (1)-(4) is a non-convex optimization problem, for example, $f(x)$ is a non-convex function in x or constraint set (2)-(3) defines a non-convex feasible region.

In this paper, we mainly discuss one dimensional problems, *i.e.*, case D_3, while higher dimensional problems (case D_4) are treated in Rebennack & Kallrath (2012, [31]). Notice that the case of mixed continuous and binary variables, D_2, is a special case of D_1, because by adding non-convex terms to the objective (1), one can enforce integrability on the decision variables, *cf.* Horst, Pardalos & Thoai (2000, [20]). However, this transformation is more of theoretical value because these non-convex terms require constants that are difficult to determine. Obviously, D_1 includes the cases D_3 and D_4.

We are not interested in solving problem (1)-(4) to global optimality because we assume that there are global solvers available, which can solve this optimization problem (or variations of it) to global optimality. Rather, we are interested in finding linear ε-approximations of the optimization problem (1)-(4) in the following sense:

Definition 1 (ε-approximated problem) Let x^* be a globally optimal solution to (1)-(4) and $|\cdot|$ denote the absolute value function. We call an optimization problem an *ε-approximated problem,* if for any optimal solution y^* of the ε-approximated problem, y^* satisfies the following properties:

$$\begin{aligned} P_1: \quad & |g_i(y^*)| \leq \varepsilon \quad , \qquad i = 1, \ldots, m_1\\ P_2: \quad & h_j(y^*) \leq \varepsilon \quad , \qquad j = 1, \ldots, m_2\\ P_3: \quad & |f(x^*) - f(y^*)| \leq \varepsilon \quad . \end{aligned}$$

Linear ε-approximations satisfying properties P_1-P_3 often require non-convex and non-concave piecewise linear functions, which can be expressed via linear functions and breakpoints. Thus, one obtains a MILP, approximating the nonlinear optimization problem (1)-(4) via such linear ε-approximations (for more details, see Section 3). The size of the resulting MILP depends crucially on the number of breakpoints and one can expect that the MILP has significantly more variables than the original MINLP/NLP (1)-(4).

The definition of the ε-approximated problem assumes that both the original nonlinear optimization problem as well as the approximated problem are feasible. However, to safely detect infeasibility of the nonlinear optimization problem (1)-(4) is an interesting problem by itself. Particularly, one might ask for the additional property on an ε-approximated problem

$$P_4: \quad \text{if the } \varepsilon\text{-approximated problem is infeasible,}$$
$$\text{then the global optimization problem (1)-(4) is also infeasible.}$$

If an optimization problem is infeasible, then we use the convention for minimization problems that the "optimal" objective function value is infinity, *e.g.*, $f(x^*) = \infty$.

We note that feasible solutions to the ε-approximated problem might not be feasible to the NOP. This relaxation enables us to obtain valid lower bounds on the optimal objective function value of NOP. Upper bounds (*e.g.*, feasible solutions to NOP) can be obtained using local (NLP) solvers, given the computed solution as a starting point. For further details, please refer to Section 3.

Finding mixed-integer linear ε-approximated representations is of particular interest, when problem (1)-(4) is embedded into a much larger optimization problem, typically a MILP. By including the nonlinear optimization problem, one obtains a large-scale MINLP, which tends to be very difficult to solve to global optimality. By reformulating the nonlinear problem as a MILP, one obtains a large-scale MILP formulation of the original problem. Such MILPs can then be solved using commercial solvers like CPLEX, Gurobi, or Xpress. Furthermore, the obtained solutions can be fed into a local MINLP solver for the final refinement.

We mention two potential applications fitting into this framework: (1) supply network problems and (2) power system optimization problems. (1) Typical supply network problems, which gave the primary motivation for the ε-approximated representations are those production planning and distributions problems with additional design aspects described by Kallrath (2002, [21]) or Kallrath & Maindl (2006, Chap. 8, [22]). (2) Power system optimization problems in the short or mid-term, *i.e.*, such as unit commitment or economics dispatch problems, often involve the scheduling generation of large, real power systems. Such models are typically formulated using MILP techniques. However, as soon as gas-networks or the electricity grid play a crucial role, MINLP techniques are required. Modeling of gas or electricity networks involve (highly) non-convex constraint systems. As such, nonlinear optimization models need to be solved to global optimality within the framework of the, potentially, much larger generation scheduling problems.

The contributions of this paper are various methods to systematically construct optimal (or "good") breakpoint systems, satisfying properties P_1-P_4, for functions depending on one variable. Due to the ε-limited discretization error and the minimal number of breakpoints, the solution of the MILP problem can be obtained in reasonable time (though the approximated problem is still NP-hard) and serves as a good approximation to the global optimum. More specifically:

1. We develop algorithms which allow to compute the *proven* minimal number of breakpoints required to piecewise linearly and continuously approximate any continuous function over a compactum (the methodology works also if the function has finitely many discontinuities).
2. For a given number of breakpoints, we develop an algorithm which can compute the tightest possible piecewise linear and continuous approximator; tightest in the sense of minimizing the largest deviation between the approximator and the function.
3. For a given $\delta > 0$, we can compute piecewise linear and continuous δ-under- and δ-overestimators.
4. We prove existence of δ-approximators, δ-under- and δ-overestimators as well as the finite convergence of all developed algorithms.

This paper is continued by a second paper with discussions on bivariate functions and transformations of multivariate functions to lower dimensional functions, see Rebennack & Kallrath (2012, [31]).

The remainder of the paper is organized as follows: Section 2 provides a review of relevant literature. The discussion on approximator systems (Section 3) motivates the construction of approximator, over- and underestimators for single functions. The construction of these approximations are discussed for univariate functions (Section 4). Finally, we conclude with a discussion in Section 5.

2 Literature Review

Within the mathematical programming community, approximating or interpolating nonlinear functions by piecewise linear function is closely related to *special ordered sets*. Tomlin (1988,[36]) is a good resource on the historical milestones of the concept of special ordered sets (of type 1, SOS-1, and of type 2, SOS-2; originally named S1 and S2 sets) explicitly introduced by Beale and Tomlin (1970, [2]), but already used earlier by Beale (1963,[1]) to deal with piecewise linear functions, or nonlinear functions approximated by piecewise linear functions. Per definition, at most one of the (usually, non-negative) variables λ_i of a special ordered set of type 1 can have a nonzero value. In standard SOS-1 sets, one exploits monotonicity with respect to the ordering of the set elements to separate subsets of the set members, and often a convexity constraint, $\sum_b \lambda_b = 1$, is present. The special order is usually reflected in a reference row, *e.g.*, $c = \sum_b C_b \lambda_b$, where the SOS-1 set is used to select the best capacity. Branching on SOS-1 sets works efficiently if the discrete capacities, C_b, are a monotonously increasing function of the index b. This example is discussed in great detail by Kallrath and Wilson (1997, Section 6.7.1, [23]).

Beale and Forrest (1976, [3]) present the idea of linear approximations to compute the global minimum of nonconvex nonlinear functions using non-negative variables λ_b forming a SOS-2 set. These variables are subject to the condition that at most two of them can have non-zero values and the two non-negative variables can only occur for adjacent indices. Beale and Forrest develop efficient branching schemes to exploit this structure. If used for interpolation, the convexity constraint

$$\sum_b \lambda_b = 1$$

is added. Since 1976, various contributions elaborated on the usage of SOS-2, among them Farias et al. (2000, [9], 2008, [10]) optimizing a discontinuous separable piecewise linear function, Leyffer et al. (2008, [25]) constructing a Branch-and-Refine algorithm for mixed integer nonconvex global optimization, or Vielma et al. (2009, [37]) developing a unifying framework and extensions to mixed-integer models for nonseparable piecewise-linear optimization problems, and the recent work by Vielma & Nemhauser (2011, [38]) using significantly fewer binary variables growing only logarithmically in the number of breakpoints.

Given these latest developments, one might argue that the number of breakpoints is not so critical anymore. While in many cases this may be true for well behaved functions, for large intervals and expressions involving trigonometric functions or functions with many local extrema it still may be crucial to keep the number of breakpoints as small as possible if piecewise linear approximations are embedded in otherwise large MILP models.

Let us at this point recall what we have in mind: We aim for tight approximators (not only interpolators) with a guaranteed accuracy by exploiting the placements of breakpoints as a degree of freedom. The framework by Vielma & Nemhauser profits from tight approximators greatly: For the same number of breakpoints and constraints, we can expect to have (better) bounds for NOP when using tight approximators.

All publications listed above use a *given* set of breakpoints, b, with known arguments, X_b, and function values $F_b := f(X_b)$ allowing to *interpolate* both function arguments

$$x = \sum_b X_b \lambda_b \qquad \text{and function values} \qquad f(x) = \sum_b F_b \lambda_b \quad .$$

Misener and Floudas (2010, [29]) presented explicit, piecewise-linear formulations of two- or three dimensional functions based on simplices.

To compute optimal breakpoints or optimal triangulations, we need to solve mixed integer nonlinear programming (MINLP) problems and semi-infinite programming (SIP) problems to global optimality: The tremendous effort in these fields (MINLP and Global Optimization) has been reviewed by Floudas (2000, [13]), Tawarmalani and Sahinidis (2002, [34]), Tawarmalani and Sahinidis (2004, [35]), Grossmann (2002, [18]), and Floudas *et al*. (2005, [14]). The book *Frontiers in Global Optimization* edited by Floudas and Pardalos (2004, [15]) gives a good overview about trends and activities in the field, and Liberti and Maculan (2006, [26]) cover theory and implementations.

The semi-infinite programming (SIP) problems we are encountering have a finite number of optimization variables, but an infinite number of nonlinear, non-convex constraints. We approach them by discretization which leads to a finite number of constraints followed by a test involving the computation of the global maximum of the deviation function. If the test fails, we refine the grid close to the idea of Blankenship and Falk (1976, [5]). SIP has a rich field of literature. Therefore, we refer the reader to the surveys by Hettich and Kortanek (1993, [19]), or Lopez and Still (2007, [27]).

Finally, in the context of over- and underestimators, the notion of outer (or inner) approximation is a valuable concept in Mathematical Programming, and, especially, in solving MINLP problems, cf. Duran and Grossmann (1986, [11]), Fletcher and Leyffer (1994, [12]), Borchers and Mitchell (1997, [6]), Kesavan et al. (2004, [24]), or Bergamini et al. (2008, [4]).

However, unlike in the seminal work by McCormick (1976, [28]), we construct piecewise linear over- and underestimators which are only piecewise convex, but not necessarily globally convex. Similarly, in outer approximation one usually approximates the feasible region by tangents or tangent hyperplanes constructed by Taylor series expansions in points of the feasible region's boundary. Instead, in our approach, we use piecewise linear functions to piecewise approximate the feasible region from inside or outside. Note that our goal is to derive best approximations to NLP or MINLP problems to be solved in the framework of MILP formulations; we are not seeking the global optimum.

Rosen and Pardalos (1986, [32]) proposed the use of piecewise linear interpolators using equidistance breakpoints for concave quadratic minimization problems (see also Pardalos and Rosen, 1987, [30, Chapter 8]). They are able to derive a condition for the number of breakpoints needed in order to achieve a given error tolerance. By concavity, their interpolators are underestimators. To the best knowledge of the authors, this is the first work which allows for the computation of breakpoints for a given error tolerance. In this respect, our work differs in the following important points: (1) we distribute the breakpoints freely, (2) we allow shifts at the breakpoints, (3) we can treat general functions, and (4) we can compute the minimal number of breakpoints required to achieve a given accuracy.

A recent publication by Geißler et al. (2012, [17]) and slightly earlier the dissertation by Geißler (2011,[16]) come in some parts close to our ideas but differ in the following aspects. The authors do not target on computing optimal breakpoint systems (minimal in the number of breakpoints) and they only estimate the approximation error (or errors for over- and underestimating) for the general case of indefinite functions while we solve nonconvex NLP problems to global optimality leading to the tightest approximators. Their approach does not involve shift variables at the breakpoints which is an important degree of freedom leading to a smaller number of breakpoints and tighter approximations. Our approach is more general in this aspect because it can handle arbitrary, indefinite functions regardless of their curvature. Our only requirement is that the functions have a finite number of discontinuities over a compactum and is bounded (*e.g.*, no singularities). Figure 10 of their paper shows discontinuities in the over- or underestimators while our approach produces continuous ones.

3 General Properties of Approximator Systems

In this section we formalize approximation to under and overestimator functions and establish their existence under mild conditions (see Section 3.1). We formalize the tightness of an approximator in Section 3.2 and connect the approximators to the nonlinear optimization problem of the form (1)-(4) in Section 3.3.

3.1 Approximators, Under- and Overestimators

First, let us recall the definitions of piecewise linear functions in n dimensions and derive from there a definition of a support area. We call a function $\ell : \mathbb{D} = D_1 \to \mathbb{R}$ in n-dimensions *piecewise linear* if there exists a finite partition of $\mathbb{D}$, *i.e.*, $\bigcup_i \mathbb{D}^i \subseteq \mathbb{D}$ with $\mathbb{D}^i \subseteq \mathbb{R}^n$, such that for all i: ℓ is a linear function in $\mathbb{D}^i$ and $\mathbb{D}^i$ is connected. For a partition, we do not require the intersection of any two sets to be empty. However, we are particularly interested in partitions, in which the intersection of any two sets $\mathbb{D}^i$ and $\mathbb{D}^j$ yields a space of at most $n-1$ dimensions. (Such a partition exists if the function is piecewise linear, *cf.* to the proof of Theorem 1.) We call any of the $\mathbb{D}^i$ a *support area* for function f.

Definition 2 (δ-approximator) Let $f : \mathbb{D} = D_1 \to \mathbb{R}$ be a function in n dimensions and let scalar $\delta > 0$. A piecewise linear, continuous function $\ell : \mathbb{D} \to \mathbb{R}$ is called a δ-*approximator* for f, if the following property holds

$$\max_{x\in\mathbb{D}} |\ell(x) - f(x)| \leq \delta \quad . \tag{5}$$

Theorem 1 (Existence of δ-approximator) *Let $f : \mathbb{D} = D_1 \to \mathbb{R}$ be a continuous function in n dimensions and $\delta > 0$. Then there exists a δ-approximator function $\ell : \mathbb{D} \to \mathbb{R}$ for f with finitely many support areas.*

Proof We use the following definition for continuous functions: f is a continuous function in x_0 over $\mathbb{D}$, if and only if $\forall \eta > 0\ \exists \gamma = \gamma(\eta) > 0$:

$$\forall x \in \mathbb{D} \text{ with } x \in B_\gamma(x_0) \text{ the following holds: } f(x) \in B_\eta\big(f(x_0)\big) \quad ,$$

where we define the n dimensional open ball with center z using metric $\|\cdot\|$ as

$$B_\mu(z) := \big\{x;\, \|x-z\| < \mu\big\} \quad . \tag{6}$$

Construct the δ-approximator ℓ for f as follows (by induction).

In the first step, choose $x_0^1 := X_-$ with $\eta := \frac{\delta}{2}$ and select an appropriate γ_0. Now construct a hyperplane h_1 for $x \in B_{\gamma_0}\big(x_0^1\big)$ with $h_1(x) \in B_\eta\,\big(f(x_0^1)\big)$ (such a hyperplane exists because $\eta, \gamma_0 > 0$).

In the k-th step, piecewise hyperplanes h_l ($l = 1,\ldots,k-1$) in n dimensions are given on their unified domain $\bigcup_{l=1}^{k-1} B_{\gamma_l}\big(x_0^l\big) \cap \mathbb{D}$ along with the points x_0^l for $l = 1,\ldots,k-1$ and the condition $B_{\gamma_m}\big(x_0^m\big) \setminus \bigcup_{l=1,l\neq m}^{k-1} B_{\gamma_l}\big(x_0^l\big) \neq \emptyset$ for all $m = 1,\ldots,k-1$. Choose x_0^k from the boundary of $\bigcup_{l=1}^{k-1} B_{\gamma_l}\big(x_0^l\big)$ and let γ_0 be the corresponding value to $\eta = \frac{\delta}{2}$ and x_0^k. Assign $\gamma_k := \text{infimum}_{l=1,\ldots,k-1}\{\gamma_0, \|x_0^k - x_0^l\|\}$. Now construct a hyperplane h_k for $x \in B_{\gamma_k}\left(x_0^k\right) \setminus \bigcup_{l=1}^{k-1} B_{\gamma_l}\left(x_0^l\right) \cap \mathbb{D}$ such that $h_k(x) \in B_\eta\left(f\left(x_0^k\right)\right)$.

Stop when $\bigcup_l B_{\gamma_l}\left(x_0^l\right) \supset \mathbb{D}$ and define the function ℓ as the collection of the hyperplanes h_k.

Next, we have to show that the construction above stops after finitely many steps. The construction requires at most countably many steps, because for each given η, one can find a *rational* γ (since $\mathbb{Q}$ is dense in $\mathbb{R}$); thus, $[X_-, X_+]^n$ can be covered by at most countably many open balls B_γ. As $\mathbb{D}$ is a compact set and $\bigcup_l B_\gamma(x_0^l)$ is an open cover of $\mathbb{D}$, there exists a finite subcover of $\mathbb{D}$. By construction, removing any open ball $B_{\gamma_l}\left(x_0^l\right)$ from the cover destroys the cover (because point x_0^l is not included in the cover). In other words, ℓ can be constructed with finitely many support areas.

Now, we modify the construction above in such a way that ℓ is continuous on $\mathbb{D}$. Therefore, we construct piecewise linear, continuous functions, which are basically collections of hyperplanes in the n dimensional space which are "glued" together. For the first step, h_1 is continuous on $B_\gamma(x_0^1) \cap \mathbb{D}$. For the k-th step, use piecewise linear functions h_k such that the collection of h_i ($i = 1, \ldots, k$) defines a continuous function over $\bigcup_{l=1}^k B_\gamma(x_0^l) \cap \mathbb{D}$. Such a function h_k exists, because by induction, the collection of h_i with $i = 1, \ldots, k-1$ defines a continuous function implying that at most $k-1$ support areas are necessary for the construction of h_k.

By construction, ℓ satisfies (5). □

If the domain of f can be partitioned finitely, such that f is continuous and bounded on each support area, then one can construct a piecewise linear δ-approximator function $\ell : \mathbb{D} \to \mathbb{R}$ for f with finitely many support areas (apply Theorem 1 for each support area where f is continuous). However, one may not be able to impose continuity on the function ℓ. In one dimension, the requirement of a finite partition means that f has finitely many points of discontinuity and is bounded.

The existence of δ-approximator functions raises the question as to how (computationally) difficult they are to construct. The answer is sobering: for an arbitrary, continuous function f and an arbitrary scalar $\delta > 0$, it is *NP-hard* to check if a piecewise linear, continuous function ℓ satisfies (5), *i.e.*, to determine if there exists an $\tilde{x} \in \mathbb{D}$ such that $|\ell(\tilde{x}) - f(\tilde{x})| > \delta$ is *NP-complete*. This follows because solving

$$\max_{x \in \mathbb{D}} |\ell(x) - f(x)|$$

has the same complexity as finding the global maximum of function f itself. (The reduction can be strictly proven by choosing $\ell \equiv 0$.) Thus, to compute a δ-approximator for an arbitrary, continuous function is *NP-hard*. We will introduce various methods to construct such δ-approximator functions in the following sections.

Under- and Overestimator are of special interest when safe bounds for optimization problems are desired. We discuss this in Section 3.3. Their definitions are formalized s follows:

Definition 3 (δ-underestimator / δ-overestimator) We call a function $\ell : \mathbb{D} = D_1 \to \mathbb{R}$ a δ-*underestimator* of function $f : \mathbb{D} \to \mathbb{R}$, if condition (5) is satisfied along with

$$\ell(x) \leq f(x) \quad x \in \mathbb{D} \quad . \tag{7}$$

We call function ℓ a δ-*overestimator* of function f, if $-\ell$ is a δ-underestimator of $-f$.

The existence of ε-underestimator / ε-overestimator is immediate by Theorem 1 under the same assumptions for function f, by using $\delta = \frac{\varepsilon}{2}$ and shifting the constructed δ-approximator by δ down / up. This procedure sustains the minimality of a support area system for δ-approximator functions:

Corollary 1 *Let $\ell : \mathbb{D} \to \mathbb{R}$ be a δ-approximator for $f : \mathbb{D} \to \mathbb{R}$ with a minimal number of support areas and let $\varepsilon = 2\delta$. Then $\ell_-(x) = \ell(x) - \delta$ and $\ell_+(x) = \ell(x) + \delta$ define an ε-underestimator and an ε-overestimator, respectively, for f with a minimal number of support areas.*

Proof The proof is by contradiction. Assume that there is an ε-underestimator ℓ_-^* for f with less support areas than ℓ_- for f. Then, ℓ_-^* has also less support areas than δ-approximator ℓ. With $\ell^* := \ell_-^* + \frac{\varepsilon}{2}$, ℓ^* is δ-approximator for f with less support areas than ℓ, contradicting the minimality of the number of support areas of ℓ. □

3.2 Tightness of Approximator Systems

Next to the minimality of the number of support areas, we are interested in obtaining tight approximators, under- or overestimators. This leads to the following definition:

Definition 4 (tightness) A δ-approximator, δ-underestimator or δ-overestimator with B support areas for function f is called *tighter* than a ϑ-approximator, ϑ-underestimator or ϑ-overestimator, respectively, with B support areas for function f, if $\delta \leq \vartheta$. A δ-approximator, δ-underestimator or δ-overestimator with B support areas is called *tight* for $f(x)$, if there is no *tighter* ϑ-approximator, ϑ-underestimator or ϑ-overestimator for f.

Interestingly, tightness is preserved when shifting approximators to obtain under- or -overestimators:

Corollary 2 *Let $\ell : \mathbb{D} \to \mathbb{R}$ be a tight δ-approximator for $f : \mathbb{D} \to \mathbb{R}$ and let $\varepsilon = 2\delta$. Then $\ell_-(x) = \ell(x) - \delta$ and $\ell_+(x) = \ell(x) + \delta$ define a tight ε-underestimator and an ε-overestimator, respectively, for f with the same number of support areas.*

Proof The proof is by contradiction. Assume that there is a ϑ-underestimator ℓ_-^* for f which is tighter than ℓ_-, *i.e.*, $\vartheta < 2\delta$. Then, $\ell^* := \ell_-^* + \frac{\vartheta}{2}$, is a tighter $\frac{\vartheta}{2}$-approximator for f than ℓ because $\frac{\vartheta}{2} < \delta$, contradicting the tightness of ℓ. □

Note that we call a piecewise linear approximator ℓ tight for function f, if the *maximal* deviation of ℓ and f is *minimal*. Practically, however, we are also interested in minimizing the area between ℓ and f. Thus, ideally, one should compute

1. first, the minimum number of support areas B^* needed to obtain a given δ-approximation ,
2. second, find a tight ϑ-approximator with B^* support areas ($\vartheta \leq \delta$), and

3. third, compute a ϑ-approximator with B^* support areas which minimizes the area between the ϑ-approximator and f.

This applies also to under- and overestimators. In this paper, we treat only on the first and the second computational step of this three phase method.

3.3 Approximator Systems

The existence of δ-approximator functions allows us to apply them in the context of problem (1)-(4). One obtains the following

Theorem 2 *Given is problem (1)-(4) with $\mathbb{D}$ being either D_1 or D_2. With a function ℓ over D_1 satisfying (5) for $\delta := \frac{\varepsilon}{2}$, any optimal solution of*

$$\begin{aligned} \min \quad & \ell(y) && (8)\\ \text{s.t.} \quad & g(y) = 0 && (9)\\ & h(y) \le 0 && (10)\\ & y \in \mathbb{D} && (11) \end{aligned}$$

satisfies properties P_1-P_3. Furthermore, (8)-(11) satisfies P_4.

Proof Assume that (8)-(11) is infeasible. As the feasible region of (9)-(11) is identical to the feasible region of (2)-(4), one obtains P_4.

Now, assume that (8)-(11) is feasible. Then, there is nothing to show for P_1 and P_2.

For P_3, let y^* be an optimal solution to (8)-(11) and x^* be an optimal solution to (1)-(4). We use a proof by contradiction: assume that

$$f(x^*) \notin \overline{B}_\varepsilon\big(f(y^*)\big) \quad ,$$

where the closed ball is defined, consistently with (6), as

$$\overline{B}_\mu(z) := \big\{x; |x-z| \le \mu\big\} \quad .$$

By (5), we obtain

$$\ell(y^*) \in \overline{B}_\delta\big(f(y^*)\big) \quad ,$$
$$\ell(x^*) \in \overline{B}_\delta\big(f(x^*)\big) \quad .$$

By assumption, $\overline{B}_\delta\big(f(y^*)\big) \cap \overline{B}_\delta\big(f(x^*)\big) = \emptyset$ and because $f(x^*) \le f(y^*)$ due to the optimality of x^*, we have that $\ell(x^*) < \ell(y^*)$ which contradicts optimality of y^*. □

An interesting special case for Theorem 2 is when g and h are linear functions.

One might expect that $x^* = y^*$ in most cases. However, note that x^* does not even have to be "close" to y^* in general.

Theorem 2 shows that if the "difficult" nonlinear relations only occur in the objective function, then linear $\varepsilon-$approximations work fine to approximate the original nonlinear optimization problem. However, if nonlinear terms occur in difficult to handle constraints, and even more complicating, in equality constraints, we rather use linear under- and overestimators.

By using underestimators for the objective function for minimization problems, one obtains the following interesting result.

Proposition 1 *Theorem 2 still holds, if we replace ℓ with an ε-underestimator for f.*

Proof We have that

$$f(y^*) - \ell(y^*) \leq \varepsilon$$
$$f(x^*) \geq l(y^*)$$

which implies $f(y^*) - f(x^*) \leq \varepsilon$. Thus, P_3 holds. □

Proposition 1 allows a tighter approximation of the global optimization problem than Theorem 2 in the case of underestimators.

By relaxing the feasible region *(outer approximation)*, lower bounds on the optimal objective function of the original problem can be calculated, which may be useful to solve the problem approximately to global optimality. This is formalized in the next theorem:

Theorem 3 (outer approximation) *Given is problem (1)-(4) with $\mathbb{D}$ being either D_1 or D_2. With functions ℓ_-, g_{i-} being ε-underestimators for f and g_i, respectively, and g_{i+}, h_{j+} being ε-overestimators for g_i and h_j, respectively, any optimal solution y^* of*

$$\min \quad \ell_-(y) \tag{12}$$
$$\text{s.t.} \quad g_{i+}(y) \leq 0 \quad , \qquad i = 1, \ldots, m_1 \tag{13}$$
$$g_{i-}(y) \geq 0 \quad , \qquad i = 1, \ldots, m_1 \tag{14}$$
$$h_{j+}(y) \leq 0 \quad , \qquad j = 1, \ldots, m_2 \tag{15}$$
$$y \in \mathbb{D} \tag{16}$$

satisfies properties P_1, P_2 and $\ell_-(y^*) \leq f(x^*)$. Furthermore, (12)-(16) satisfies P_4.

Proof P_1 and P_2 hold by construction of (13)-(16). The condition $\ell_-(y^*) \leq f(x^*)$ and P_4 hold true because any feasible solution x to (1)-(4) is also feasible for (12)-(16).□

By Theorem 3, any optimal solution of (12)-(16) defines a (valid) lower bound to the original problem (1)-(4). An upper bound can be obtained, *e.g.*, by running a local solver (using the obtained solution y^*).

One may wonder, if it is possible to prove a stronger version of Theorem (3) along the lines of Theorem 2. To do so, consider the following one dimensional example for any positive scalar M:

$$\begin{aligned} \min \quad & M x_1 \\ \text{s.t.} \quad & -x_1^2 + \frac{\varepsilon}{2} \leq 0 \\ & x_1 \in \{0, 1\} \quad , \end{aligned}$$

with the (unique) optimal solution $x_1^* = 1$ and optimal objective function value $f(x_1^*) = M$. Let ℓ_+ be an ε-overestimator of $h(x) := -x_1^2 + \frac{\varepsilon}{2}$. Thus, any optimization problem of the form

$$\min \quad My_1 \tag{17}$$
$$\text{s.t.} \quad \ell_+(y_1) \le 0 \tag{18}$$
$$y_1 \in \{0,1\} \quad , \tag{19}$$

may allow $y_1^* = 0$ as an optimal solution of (17)-(19) with objective function value 0. (The interested reader may construct an example over a discrete domain, by replacing the domain by $[0,1]$ and adding the term $Mx_1(1-x_1)$ to the objective, and working with na ε-underestimator for the objective.)

In this example, one could only avoid the phenomenon of arbitrary large differences in the objective function, by using a δ-overestimator of $h(x)$ with $\delta < \frac{\varepsilon}{2}$.

When shrinking the feasible region *(inner approximation)*, one may obtain an upper bound on the objective function of the original problem if the approximated problem is feasible.

Theorem 4 (inner approximation) *Given is problem (1)-(4) with $\mathbb{D}$ being either D_1 or D_2. With functions h_{j-} being ε-underestimators for h_j and ℓ_+ being ε-overestimators for f (h_j and f are defined over D_1), any optimal solution y^* of*

$$\min \quad \ell_+(y) \tag{20}$$
$$\text{s.t.} \quad g(y) = 0 \tag{21}$$
$$h_{j-}(y) \le 0 \quad , \qquad j = 1,\ldots,m_2 \tag{22}$$
$$y \in \mathbb{D} \tag{23}$$

satisfies properties P_1, P_2 and $\ell_-(y^*) \ge f(x^*)$.

Note that P_4 does not hold for the inner approximation (20)-(23).

In the following sections we focus on the construction of tight, minimal piecewise linear approximators, over- and underestimators for single functions rather than a system of functions. These estimators can then be embedded via Theorems 2, 3, 4 as well as Proposition 1 into nonlinear optimization problems of the from (1)-(4).

4 Univariate Functions

In this section we discuss the construction of breakpoint systems for one dimensional functions $f : \mathbb{D} = D_3 \to \mathbb{R}$. For one-dimensional functions, a support area is an interval and we call the two end-points of each interval *breakpoints*. As such, any function f has at least two breakpoints. Thus, minimizing the number of support areas is equivalent to minimizing the number of breakpoints for one-dimensional functions. All the developed methods in the following sections apply as well for each component of the functions g and h defined in the constraint set of problem (1)-(4).

4.1 Computing an Optimal Set of Breakpoints

We are looking for a piecewise linear, continuous function $\ell : \mathbb{D} = D_3 \to \mathbb{R}$ that satisfies condition (5), *i.e.*, a δ-approximator for f. The function $\ell(x)$ depends on the breakpoints $b \in \mathscr{B}$. Let $\mathscr{B} := \{1,\ldots,B\}$ be a sufficiently large, finite set of breakpoints. Later, we explicitly define what "sufficiently large" means in this context (see Corollary 4). Each breakpoint $b \in \mathscr{B}$ occurs at the point $x_b \in (X_-, X_+]$. Given such a function ℓ satisfying condition (5), we are further interested in a minimal set of breakpoints and their locations or arguments x_b.

In the improved SOS-2 based interpolation scheme, we allow the linear approximator to deviate by s_b from the function values $f(x_b)$ at the breakpoint arguments x_b, where $s_b \in [-\delta, +\delta]$ allows us to change the orientation of the piecewise linear segments. Once, we have computed x_b and s_b, the interpolation (5) is replaced by

$$f(x) = \sum_b \left(f(x_b) + s_b \right) \lambda_b \quad ,$$

which again leads to a continuous linear approximator. To keep our formulas simple, we define

$$\phi(x_b) = f(x_b) + s_b \quad , \quad \forall b \in \mathscr{B} \quad . \tag{24}$$

The ideal situation would minimize the number of breakpoints and compute the corresponding optimal distribution of these breakpoints, satisfying condition (5). By using interpolation techniques, one can construct a piecewise linear, continuous function ℓ:

$$\text{OBSC} : z^* = \min \sum_{b \in \mathscr{B}} \chi_b \tag{25}$$

$$\text{s.t.} \quad x_{b-1} \le x_b \quad , \quad \forall b \in \mathscr{B} \tag{26}$$

$$x_b \ge X_- + (X_+ - X_-)(1 - \chi_b) \quad , \quad \forall b \in \mathscr{B} \tag{27}$$

$$x_b - x_{b-1} \ge \frac{1}{M} \chi_b \quad , \quad \forall b \in \mathscr{B} \tag{28}$$

$$x_b - x_{b-1} \le (X_+ - X_-)\, \chi_b \quad , \quad \forall b \in \mathscr{B} \tag{29}$$

$$y_b = x_b - x_{b-1} + 1 - \chi_b \quad , \quad \forall b \in \mathscr{B} \tag{30}$$

$$\chi_{bx}^x \le \chi_b \quad , \quad \forall b \in \mathscr{B}, \quad \forall x \in [X_-, X_+] \tag{31}$$

$$x_{b-1} - X_- (1 - \chi_{bx}^x) \le x \le x_b + X_+ (1 - \chi_{bx}^x) \quad , \quad \forall b \in \mathscr{B}, \quad \forall x \in [X_-, X_+] \tag{32}$$

$$\ell_b(x) := \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{y_b} (x - x_{b-1}) \quad , \quad \forall b \in \mathscr{B}, \quad \forall x \in [X_-, X_+] \tag{33}$$

$$\ell(x) := \sum_{b \in \mathscr{B}} \ell_b(x) \chi_{bx}^x \quad , \quad \forall x \in [X_-, X_+] \tag{34}$$

$$|\ell(x) - f(x)| \le \delta \quad , \quad \forall x \in [X_-, X_+] \tag{35}$$

$$x_b \in [X_-, X_+], \quad s_b \in [-\delta, +\delta], \quad \chi_b \in \{0,1\}, \quad \chi_{bx}^x \in \{0,1\}, \quad y_b \ge \frac{1}{M}, \quad \forall b \in \mathscr{B}, \quad \forall x \in [X_-, X_+] \tag{36}$$

where we define $x_0 := X_-$ and $\phi(x_b)$ is given by (24).

The binary indicator variable χ_b has value 1, if breakpoint $b \in \mathscr{B}$ is included in the linear approximation ℓ and 0 otherwise. Constraints (26) sort the breakpoints while (27) connects variables χ_b with the coordinates x_b of the breakpoints. Particularly, if $\chi_b = 0$, inequalities (27) imply $x_b = X_+$, *i.e.*, all inactive breakpoints are placed on the upper bound, or equivalently, all breakpoints not included in the construction of ℓ are set to X_+. Note that the number of breakpoints included in ℓ is thus z^* +2, because the objective (25) does not count $x_0 = X_-$ and $x_b = X_+$ as breakpoints for ℓ. Variables y_b take value $x_b - x_{b-1}$ if $x_b - x_{b-1} > 0$ and 1 otherwise. This is modeled via constraints (28)-(30) with an appropriate constant M (*e.g.*, $\frac{1}{M}$ equals machine precision).Variable χ^x_{bx} is 1, if $x \in [x_{b-1}, x_b]$ and 0 otherwise. This is modeled via constraints (31)-(32). The definitions (33)-(34) should not be interpreted as constraints but rather as auxiliary definitions to construct the function ℓ as a shifted interpolation of function f. Note that inequality (35) turns our problem into the class of SIP. As formulation (25)-(36) leads to an *O*ptimal *B*reakpoint *S*ystem using a *C*ontinuum approach for x, we call it "OBSC." This discussion implies

Corollary 3 *If OBSC is feasible, then ℓ is a δ-approximator for f with the minimum number of breakpoints being $z^* + 2$.*

Note that any feasible solution to OBSC with B breakpoints can be extended to be valid for OBSC for any $\overline{B} \geq B$, by assigning $\chi_b = 0$, $x_b = X_+$, and $y_b = 1$ for any $\overline{\mathscr{B}} \setminus \mathscr{B}$ and copying the values for other variables from the solution with B breakpoints. This implies that $z^*(B) \geq z^*(\overline{B})$. If OBSC is infeasible for $\overline{B}$, then it is also infeasible for B. Furthermore, if OBSC is feasible for B, then $z^*(B) = z^*(\overline{B})$. Thus, they are either equal, or one is finite and the other is $+\infty$. The existence of a finite choice for B to make OBSC feasible is established in

Corollary 4 *If f is a continuous function over D_3, then there exists a finite B^* such that for all $B \geq B^*$ OBSC is feasible.*

Proof We need to prove that any continuous function f can be constructed via interpolation (as proposed in (33)) with finitely many breakpoints (this proves the existence of a finite B^*). However, this follows by using the arguments of the proof of Theorem 1 and using interpolation to construct the hyperplanes h_k. Once such a finite B^* has been obtained, then for all $B \geq B^*$, OBSC is feasible with the above construction. □

Note that x in OBSC is not a decision variable and can vary in the interval $[X_-, X_+]$. This makes OBSC a semi-infinite MINLP problem, which are notoriously difficult to solve. To obtain a computationally tracktable mathematical program, we discretize the continuum constraints (35) into I finite constraints of the form

$$|\ell(x_i) - f(x_i)| \leq \varepsilon \quad , \quad \forall i \in \mathbb{I} := \{1, \ldots, I\} \quad , \tag{37}$$

for appropriately selected grid points x_i. Applying this approach to *each* of the B breakpoints x_b in formulation OBSC leads to the following Discretized Optimal Breakpoint System (OBSD):

$$\text{OBSD} : z^{D*} = \min \sum_{b \in \mathscr{B}} \chi_b \tag{38}$$

$$
\begin{aligned}
\text{s.t.} \quad & x_{b-1} \leq x_b \quad , \quad \forall b \in \mathscr{B} & (39)\\
& x_b \geq X_- + (X_+ - X_-)(1-\chi_b) \quad , \quad \forall b \in \mathscr{B} & (40)\\
& x_b - x_{b-1} \geq \frac{1}{M}\chi_b \quad , \quad \forall b \in \mathscr{B} & (41)\\
& x_b - x_{b-1} \leq (X_+ - X_-)\,\chi_b \quad , \quad \forall b \in \mathscr{B} & (42)\\
& y_b = x_b - x_{b-1} + 1 - \chi^d \quad , \quad \forall b \in \mathscr{B} & (43)\\
& x_{bi} = x_{b-1} + \frac{i}{I+1}(x_b - x_{b-1}) \quad , \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} & (44)\\
& l_{bi} = \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{y_b}(x_{bi} - x_{b-1}) \quad , & \\
& \qquad \qquad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} & (45)\\
& |l_{bi} - f(x_{bi})| \leq \delta \quad , \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} & (46)\\
& x_b \in [X_-, X_+], \quad s_b \in [-\delta, +\delta], \quad \chi_b \in \{0,1\}, \quad y_b \geq \frac{1}{M}, & \\
& x_{bi} \in [X_-, X_+], \quad l_{bi} \ \text{free}, \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} & (47)
\end{aligned}
$$

with variables x_{bi} are the uniform discretization or grid point within the breakpoint interval $[x_{b-1}, x_b]$, and variables l_{bi} evaluate the interpolation of $\phi(x_{b-1})$ and $\phi(x_b)$ at point x_{bi} with $\phi(x_b)$ being defined by (24).

The size (in terms of number of variables and constraints) of OBSD depends strongly on the number of breakpoints, B, and the discretization size I. Constraints (45) and (46) make problem OBSD a highly non-convex MINLP. Thus, OBSD is potentially a large-scale MINLP which is very hard to solve. However, if X_- and X_+ are relatively close together, then OBSD might be computationally tractable if f is not too "bad."

A piecewise linear, continuous function ℓ can be constructed by using the breakpoints x_b^* obtained from solving OBSD using interpolation as in (45). For this function ℓ, one must solve

$$z_\ell^* = \max_{x \in [X_-, X_+]} |\ell(x) - f(x)|$$

to global optimality to check whether or not its objective function value does exceed δ. If $z_\ell^* \leq \delta$, then ℓ defines a δ-approximator for f. If not, then increasing the interval discretization size I and resolving OBSD might help. However, one may be forced to also increase the number of breakpoints. We summarize this in the following

Corollary 5 *Let OBSD be feasible for B and I. If ℓ constructed from (45) satisfies (5), then ℓ is a δ-approximator for f with the minimum number of breakpoints being $z^{D*} + 2$. If ℓ does not satisfy (5), then $z^{D*} + 2$ defines a lower bound on the minimum number of breakpoints on any δ-approximator for f.*

Alternatively to discretizing each breakpoint interval into I grid points, one can distribute I a priori given grid points within the interval $[X_-, X_+]$:

$$
\begin{aligned}
\text{OBSI} : z^* = \min \quad & \sum_{b \in \mathscr{B}} \chi_b & (48)\\
\text{s.t.} \quad & x_{b-1} \leq x_b \quad , \quad \forall b \in \mathscr{B} & (49)
\end{aligned}
$$

$$x_b \geq X_- + (X_+ - X_-)\,(1 - \chi_b) \quad , \quad \forall b \in \mathscr{B} \tag{50}$$

$$x_b - x_{b-1} \geq \frac{1}{M}\chi_b^d \quad , \quad \forall b \in \mathscr{B} \tag{51}$$

$$x_b - x_{b-1} \leq (X_+ - X_-)\,\chi_b^d \quad , \quad \forall b \in \mathscr{B} \tag{52}$$

$$y_b = x_b - x_{b-1} + 1 - \chi_b^d \quad , \quad \forall b \in \mathscr{B} \tag{53}$$

$$\chi_{bi}^x \leq \chi_b \quad , \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{54}$$

$$x_{b-1} - X_-\,(1 - \chi_{bi}^x) \leq x_i \leq x_b + X_+\,(1 - \chi_{bi}^x) \quad , \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{55}$$

$$l_{bi} = \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{y_b}\,(x_i - x_{b-1}) \quad , \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{56}$$

$$l_i = \sum_{b \in \mathscr{B}} \ell_{bi}\chi_{bi}^x \quad , \quad \forall i \in \mathbb{I} \tag{57}$$

$$|l_i - f(x_i)| \leq \delta \quad , \quad \forall i \in \mathbb{I} \tag{58}$$

$$x_b \in [X_-, X_+], \quad \chi_b \in \{0,1\}, \quad \chi_{bi}^x \in \{0,1\}, \quad y_b \geq \frac{1}{M}, \quad s_b \in [-\delta, +\delta], \quad l_b \text{ free}, \quad l_{bi} \text{ free}, \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{59}$$

where $x_i = \frac{i}{I}\,(X_+ - X_-) + X_-$ are now input data and $\phi(x_b)$ is obtained by (24).

Let us compare OBSD with OBSI. For one, OBSD does not require both the $B \cdot I$ binary variables χ_{bi}^x and constraints (54), (55), (57). Second, additional $B \cdot I$ continuous variables x_{bi} are introduced in the OBSD formulation, requiring constraints (44). Furthermore, constraints (45) involve the additional variables x_{bi} compared to constraints (56). Though binary variables tend to be computationally burdensome, nonconvex terms are at least as computationally challenging. Thus, it is not a priori clear which formulation, OBSD or OBSI, is computationally superior.

Note that an equivalent version to Corollary 5 exists for OBSI.

4.2 Computing a Tight δ-Approximator for a Fixed Number of Breakpoints

Overall, problems OBSC, OBSD and OBSI are in general too large and difficult to solve. Only for modest numbers of breakpoints and not too many discretization points there is a chance to solve these problems to global optimality. Alternatively, we could solve the optimal distribution of a fixed number, B, of breakpoints for the discretized continuum constraint

$$|\ell(x_i) - f(x_i)| \leq \mu \quad , \quad \forall i \in \mathbb{I}$$

and minimize μ followed by a check whether μ is less than or equal to our approximation quality (*e.g.*, δ).

We use the idea of formulation OBSD and discretize each interval (x_{b-1}, x_b) into I equidistant grid points. This puts us into the advantageous situation that we know to which breakpoint interval the variables x_{bi} belong to, *i.e.*, we do not need the binary variables χ_{bi}^x. By forcing the usage of exactly $B-1$ breakpoints (note, we do not count

$x_0 = X_-$ nor $x_B = X_+$ as breakpoints), we can also eliminate the binary variables χ_b. We obtain a continuous NLP:

$$\begin{aligned}
\text{FBSD} : \mu^* = \min \ & \mu && (60)\\
\text{s.t.} \ & x_b - x_{b-1} \geq \frac{1}{M} \quad , \quad \forall b \in \mathscr{B} && (61)\\
& l_{bi} = \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{x_b - x_{b-1}} (x_{bi} - x_{b-1}) \quad , \\
& \qquad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} && (62)\\
& x_{bi} = x_{b-1} + \frac{i}{I+1} (x_b - x_{b-1}) \quad , \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} && (63)\\
& |l_{bi} - f(x_{bi})| \leq \mu \quad , \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} && (64)\\
& x_b \in [X_-, X_+], \quad x_{bi} \in [X_-, X_+], \quad l_{bi} \ \text{free} \quad , \\
& \mu \geq 0, \quad s_b \in [-\delta, +\delta], \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} && (65)
\end{aligned}$$

Note that at the breakpoints the function deviation is bounded by δ. Therefore, we do not need discretization points at the breakpoints. The solution of the FBSD minimization problem provides a breakpoint system x_b, the shift variables s_b, and the minimal value, μ^*. Note that both are functions of B and I, *i.e.*, $\mu^* = \mu^*(B,I)$ and $x_b = x_b(B,I)$.

The obtained breakpoints and shift variables yield a ϑ-approximator for $f(x)$. In order to compute ϑ, we solve the maximization problem

$$\delta_b(B,I) := \max_{x \in [x_{b-1}, x_b]} |\ell(x) - f(x)|$$

for each interval $[x_{b-1}, x_b]$, to yield

$$\vartheta = \delta^*(B,I) := \max_{b \in \mathbb{B}} \delta_b(B,I) \quad .$$

Let δ^*-approximator be a tight approximator with B breakpoints. Then the optimal solution value of FBSD is a lower bound on δ^*, *i.e.*, $\mu^* \leq \delta^*$. Thus, if $\mu^* = \vartheta$, then $\vartheta = \delta^*$ and the computed ϑ-approximator is tight. By choosing the discretization size I appropriately, $\mu^*(B,I)$ and $\delta^*(B,I)$ can get arbitrarily close to each other. In other words, for a fixed number of breakpoints, FBSD can calculate the tightest possible approximator. This is formalized in the next

Corollary 6 *Let f be a continuous function and B be fixed. Then, for each $\eta > 0$, there exists a finite I^*, such that $\mu^*(B,I^*) + \eta \geq \delta^*(B,I^*)$.*

Proof Function $d(x) := |\ell(x) - f(x)|$ is continuous in $[X_-, X_+]$. By definition of a continuous function in $x_0 \in [X_-, X_+]$, we can find for each $\eta > 0$ (this is the same η as in the Corollary) a $\gamma > 0$ such that $d(x) \in B_{\frac{\eta}{2}}(d(x_0))$ for all $x \in B_\gamma(x_0)$. Now, we just need to make sure that each open ball $B_\gamma(x_0)$ contains (at least) one x_{bi} (the shift variables are continuous and thus not of a concern here).

For a given $\eta > 0$, we can find a finite series of γ's such that the corresponding open balls cover $[X_-, X_+]$, because $[X_-, X_+]$ is compact. Let γ^* be the smallest among all γ's and choose $I^* := (X_+ - X_-)\frac{1}{\gamma^*} + 1$. □

The proof of Corollary 6 does not provide a practical way of choosing I^*. Furthermore, $\mu^*(\cdot, I)$ is not a monotonic decreasing function in I (a monotonic decreasing optimal objective function value might help to computationally find the tightest δ-approximator). However, for given I, μ^* provides a lower bound on any approximator quality while δ^* defines an upper bound. Thus, if μ^* and δ^* are close enough to each other (*e.g.*, machine precision), then δ^*-approximator is the tightest possible δ-approximator for f with B breakpoints. This suggests the following algorithm on how to compute a tight δ-approximator: choose $I \in \mathbb{N}$ and solve FBSD; if $\delta^*(B,I) = \mu^*$, then we have found a tight ϑ-approximator, otherwise increase I and start over until $\delta^*(B,I) = \mu^*$. By Corollary 6, this procedure terminates in finitely many steps (at least up to a certain precision when $\delta^*(B,I) \approx \mu^*$).

Observe that $\mu^*(B,\tilde{I})$ is a monotonic non-increasing function in the number of breakpoints B, with $\tilde{I} \geq I^*(B)$. This monotonicity enables us to compute a δ-approximator with the least number of breakpoints as follows: start with an initial number of breakpoints and compute a tight ϑ-approximator via the methods described above; if $\vartheta \leq \delta$, then ϑ-approximator is a δ-approximator with the least number of breakpoints, otherwise, increase the number of breakpoints by one and start over.

Point symmetric functions are somewhat special in the distribution of approximator systems:

Theorem 5 *Let $f : [X_-, X_+] \rightarrow \mathbb{R}$ be a continuous function which is point symmetric at $\frac{X_- + X_+}{2}$. Then, there exists an optimal (least number of breakpoints B^* and tightest among all approximators with B^* breakpoints) δ-approximator which is point symmetric at $\frac{X_- + X_+}{2}$.*

Proof Let $y := \frac{X_- + X_+}{2}$ and x_b^*, $b \in \mathbb{B}$, be a breakpoint system for f with minimal number B^* of breakpoints with a non-symmetric δ-approximator.

If y is one of the breakpoints, then it is easy to see that B^* is odd and that there has to be an optimal, point symmetric δ-approximator.

If B^* is odd, then one can construct an optimal, symmetric breakpoint system as follows: add a breakpoint at y and axially mirror the breakpoint system with fewer breakpoints in one of the intervals. The constructed approximator has the same tightness as the original approximator.

Otherwise, if B^* is even, then there has to be the same number of breakpoints in each of the two intervals (reasoning follows below). In this case, one mirror the breakpoint systems for the intervals $[X_-, y)$ and $(y, X_+]$ where the corresponding breakpoint is closes to y, in order to obtain a symmetrically distributed breakpoint system for f which is as tight as possible. In case that the number of breakpoints is different for each of the two intervals, one can mirror the breakpoint system with fewer breakpoints and add a breakpoint at y, leading to an approximator with the same (or better) tightness using fewer breakpoints. This is a contradiction. □

4.3 Successively Computing a Good Set of Breakpoints

In Section 4.1, we provided formulations to compute all breakpoints simultaneously by solving one optimization model. As we learn in Section 4.5, solving these mathematical programming problems is, computationally, very expensive, which is not surprising because we show in Section 3 that the computation of optimal breakpoint systems is *NP-hard*. Thus, we propose a forward scheme moving successively from $x_0 = X_-$ to some breakpoint $x_b \leq X_+$ covering the whole interval.

For a given breakpoint, x_{b-1}, we can compute the next breakpoint, x_b, with the following SIP problem:

$$\text{BSB}: \zeta^* = \max \; x_b \tag{66}$$

$$\text{s.t.} \quad \left| \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{x_b - x_{b-1}} (x - x_{b-1}) - f(x) \right| \leq \delta \quad , \quad \forall x \in [x_{b-1}, x_b] \tag{67}$$

$$x_b \in (x_{b-1}, X_+], \quad s_b \in [-\delta, +\delta] \quad . \tag{68}$$

When BSB is solved and an optimal x_b^* as well as the shifting variable s_b^* is obtained, then both x_b^* and s_b^* are fixed for the problem $b+1$ (if $x_b < X_+$). Thus, BSB contains only two decision variables for $b > 1$. However, for $b = 1$, we use the convention that $x_0 := X_-$ and that $s_0 \in [-\delta, +\delta]$ is an additional decision variable for BSB. Though BSB only has two or three decision variables, it is difficult to solve because of the continuous constraints (67).

The appearance of constraints (67) is worrisome as the decision variable x_b appears in the index set of the infinitely many constraint (67). However, we find this presentation of problem BSB beneficial when discussing the presented heuristic methods below.

Note that successively computing breakpoints by maximizing the length of the intervals does not necessarily lead to an optimal breakpoint system, *i.e.*, a δ-approximator with the least number of breakpoints. It might be beneficial, in certain cases, to consider intervals between two breakpoints which are not maximal length; particularly as maximizing the interval length may lead to a large shift variable which might decrease the length of the proceeding intervals. Therefore, consider the following continuous function $f(x)$ for fixed $\delta = 0.25$ and $x \in [0, 5]$:

$$f(x) := \begin{cases} 1, & \text{if } x \in [0, 2) \\ -0.50 + 0.75x, & \text{if } x \in [2, 3) \\ 1.75 - \delta(x-3), & \text{if } x \in [3, 4) \\ 1.75 - \delta + 2\delta(x-4), & \text{if } x \in [4, 5] \end{cases} \quad . \tag{69}$$

Figure 1 shows $f(x)$ together with a (unique) optimal δ-approximator using three breakpoints and a δ-approximator using four breakpoints obtained by a method maximizing the interval length successively from X_- to X_+.

We present two heuristic methods to compute a breakpoint system iteratively, based on two different approaches on how to tackle problem BSB.

4.3.1 α-Forward Heuristic with Backward Iterations

Similar to the setup in the previous section, we assume that a breakpoint x_{b-1} is already given and that we want to find the next one, x_b. The heuristic presented in

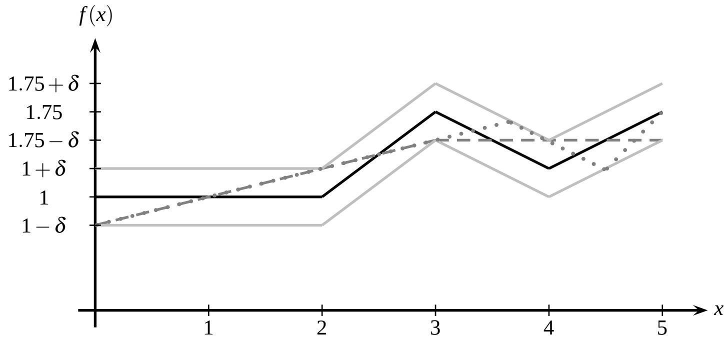


Fig. 1: Maximizing the length of the intervals successively is not optimal, in general
— $f(x)$
— δ-tube around $f(x)$
- - (unique) optimal δ-approximator (3 breakpoints)
··· δ-approximator maximizing interval length successively (4 breakpoints)

this section fixes both x_b and the shift variables; they are decision variables in the heuristic presented in Section 4.3.2. We then need to check whether the obtained approximator satisfies $\Delta_b \leq \delta$, by solving

$$\Delta_b := \max_{x \in [x_{b-1}, x_b]} |\ell(x) - f(x)| \tag{70}$$

for interpolator

$$\ell(x) := \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{x_{b-1} - x_b} (x - x_{b-1}) \tag{71}$$

to global optimality. If $\Delta_b \leq \delta$, then we accept x_b as the new breakpoint together with the shift variables. Otherwise, we try a different value for the shift variables or shrink the interval and replace the current value of x_b by

$$x_b \leftarrow x_{b-1} + \alpha(x_b - x_{b-1}) \quad , \quad 0 < \alpha < 1 \quad . \tag{72}$$

This idea is summarized in pseudo-code format in Algorithm 4.1. This heuristic method never gets “stuck:”

Corollary 7 *Algorithm 4.1 terminates after a finite number of iterations for any continuous function f, any $\delta > 0$, any $\alpha \in (0,1)$ and any $D \in \mathbb{N}$. The calculated breakpoints with the shift variables yield a δ-approximator for f.*

Proof We need to show that both the inner and the outer loop are finite.

For the inner loop, consider the continuous function $\tilde{d}(x) := \left|\tilde{\ell}(x) - f(x)\right|$ in $x \in [x_{b-1}, X_+]$ with fixed shift variable s_{b-1} and $\tilde{d}(X_+) = 0$. Let $\tilde{\delta} := \delta - \tilde{d}(x_{b-1})$. Given x_{b-1} and $\tilde{\delta} > 0$, then there exists an $\eta > 0$ such that for all $x \in [x_{b-1}, x_{b-1} + \eta)$: $\tilde{d}(x) \in$

Algorithm 4.1 α-Forward Heuristic with Backward Iteration Computing a δ-Approximator

// **INPUT:** Continuous function f, scalar $\delta > 0$, parameter $\alpha \in (0,1)$, and shift variable discretization size D
// **OUTPUT:** Number of breakpoints, B, breakpoint system x_b and shift variables s_b
// Initialize
$x_0 := X_-$, $B := 0$, $b = 1$, and $s_0 := 0$
// **Outer loop**
repeat
    // x_b will be assigned X_+ after first counter update
    $x_b := \frac{1}{\alpha} X_+ - \frac{1-\alpha}{\alpha} x_{b-1}$
    // **Inner loop**
    repeat
        // update breakpoint and reset counter
        $x_b \leftarrow x_{b-1} + \alpha(x_b - x_{b-1})$ and $d := 0$
        repeat
            // increment counter and assign discretized value for shift variable
            $d \leftarrow d + 1$ and $s_{bd} := \left(\frac{2d}{D+1} - 1\right)\delta$
            // optimize
            solve (70) with fixed x_{b-1}, x_b, s_{b-1} and s_{bd} to obtain Δ_b
        until $\Delta_b \le \delta$ or $d = D$
    until $\Delta_b \le \delta$
    // fix shifting variable and update counter
    $s_b := s_{bd}$, $b \leftarrow b + 1$, $B \leftarrow B + 1$
until $x_b = X_b$

$B_{\frac{\tilde{\delta}}{2}}\left(\tilde{d}(x_{b-1})\right)$ (because $\tilde{d}$ is continuous in x_{b-1}). Thus, choose any $x_b \in \left(x_{b-1}, x_{b-1} + \frac{\eta}{2}\right]$ which can be obtained, for instance, by looping

$$n \ge \left\lceil \frac{\log\left(\frac{\eta}{2\left(X_+ - x_{b-1}\right)}\right)}{\log(\alpha)} \right\rceil$$

and $n \in \mathbb{N}$ times. Note that the function $\tilde{\ell}(x)$ is *not* necessarily an approximator we can construct in the algorithm because $\tilde{d}(x_b)$ might not be equal to one of the discretized shift variables. However, for the corresponding function $\ell(x)$ on $[x_{b-1}, x_b]$ with any shift variable $s_b \in [-\frac{\tilde{\delta}}{2}, \frac{\tilde{\delta}}{2}]$, we have that $d(x) := |\ell(x) - f(x)| \le \delta$ for all $x \in [x_{b-1}, x_b]$ because $d(x) \in B_{\frac{\tilde{\delta}}{2}}\left(\tilde{d}(x)\right)$ for all $x \in [x_{b-1}, x_b]$. Such an s_b exists for $D \in \mathbb{N}$ because $\min_{s_{bd}}\{|\frac{\tilde{\delta}}{2}|\} = \min_{s_{bd}}\{|\frac{\delta - s_{bd}}{2}|\} = \frac{\delta}{D+1} \ge \min_{s_{bd}}\{|s_{bd}|\}$.

The outer loop is finite through the compactness of interval $[X_-, X_+]$: Construct an open cover of $[X_-, X_+]$ as follows. For each outer iteration b, choose $x_b^1 := x_{b-1} + \frac{1}{2}(x_b - x_{b-1})$ and $\xi_b^1 = \frac{1}{2}(x_b - x_{b-1})$ as well as $x_b^2 := x_{b-1}$ and $\xi_b^2 \in (x_{b-1} - x_{b-2}, x_b - x_{b-1})$ with $x_{-1} := X_- - \tau$ and appropriate $\tau > 0$ (*e.g.*, $\tau = x_1 - x_0$), as shown in Figure 2. Then, $\bigcup_b \left(B_{\xi_b^1}(x_b^1) \cup B_{\xi_b^2}(x_b^2)\right)$ is an open cover of $[X_-, X_+]$. Removing any of the open balls $B_{\xi_b^1}(x_b^1)$ or $B_{\xi_b^2}(x_b^2)$ from the cover destroys the cover. Thus, by compactness of $[X_-, X_+]$, the number of open balls has to be finite. $\square$

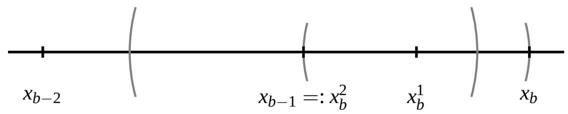


Fig. 2: Cover obtained for outer iteration b of the proof of Corollary 8

In order to avoid solving too many global optimization problems (70), we place I grid points, x_{bi}, according to (44) into the interval $[x_{b-1}, x_b]$. For each grid point, we check whether or not

$$|\ell(x_{bi}) - f(x_{bi})| \leq \delta \quad , \quad \forall b \in \mathscr{B} \quad , \quad \forall i \in \mathbb{I} \quad . \tag{73}$$

Only if condition (73) is satisfied for all grid points, we solve the global optimization problem (70). Note that condition (73) is satisfied for both breakpoints x_{b-1} and x_b by construction (as long as the absolute value of the shift variables do not exceed δ).

Further, it is not necessary to fix the shift variable for the first breakpoint X_- at value 0. This value can be discretized in the same way as all other shift variables, however, this made it easier to present the algorithm.

This discretization of $[x_{b-1}, x_b]$, together with the global optimality check, as well as the discretization of the shift variables, s_0, does not alter the correctness and finiteness of Algorithm 4.1.

Note the tradeoff of choosing α close to 0 (many subproblems to solve and many breakpoints) and close to 1 (smaller number of breakpoints but possibly many subproblems which fail the test "$\Delta_b \leq \delta$?"). However, when using the discretization of $[x_{b-1}, x_b]$, the computational burden for increasing α values is rather small as the bottleneck of Algorithm 4.1 is the solution of the global optimization problem (70).

4.3.2 Forward Heuristic with Moving Breakpoints

We again employ a marching procedure to cover the interval $[X_-, X_+]$. Similar to Heuristic 4.1, we are providing a heuristic to solve problem BSB. However, in this section, for a given breakpoint x_{b-1} and shift variable s_{b-1}, we maximize the interval length by treating x_b and the shift variable s_b as decision variables. To decrease the notational burden, we assume $s_0 \equiv 0$ and we discuss the generalization later.

Using the idea of Section 4.2, we treat the continuum inequalities (67) by placing I grid points equidistantly into the interval $[x_{b-1}, x_b]$ according to (44). At these grid points x_{bi}, we require:

$$|\ell(x_{bi}) - f(x_{bi})| \leq \delta \quad , \quad \forall i \in \mathbb{I} \quad . \tag{74}$$

Note that we do not need grid points at the breakpoints x_{b-1} and x_{b-1} because per defintionem we know that the maximal deviation is s_{b-1} and s_b, which in turn is bounded by δ.

Maximization of x_b leads to the following NLP

$$\Delta^{\mathrm{I}*} := \max \;\; x_b \tag{75}$$

$$\text{s.t.} \;\; |\ell(x_{bi}) - f(x_{bi})| \leq \delta \quad , \quad \forall i \in \mathbb{I} \tag{76}$$

$$x_{bi} = x_{b-1} + \frac{i}{I+1}(x_b - x_{b-1}) \quad , \quad \forall i \in \mathbb{I} \tag{77}$$

$$x_b \in [x_{b-1}, X_+], \quad x_{bi} \in [x_{b-1}, X_+], \quad s_b \in [-\delta, \delta], \qquad \forall i \in \mathbb{I} \tag{78}$$

with the interpolator ℓ derived by (71).

For given breakpoint x_b^*, we minimize the absolute value of s_b. That way, we get the tightest approximator for the given interval $[x_b, x_{b-1}]$, by solving the one-dimensional optimization problem

$$\Delta^{\mathrm{S}*} := \min \ |s_b| \tag{79}$$

$$\text{s.t.} \ |\ell(x_{bi}) - f(x_{bi})| \le \delta \quad , \quad \forall i \in \mathbb{I} \tag{80}$$

$$s_b \in [-\delta, \delta] \tag{81}$$

where the discrete grid points x_{bi} are now fixed together with x_b.

Due to the discretization of the continuum $[x_{b-1}, x_b]$, we need to check whether for the given value of x_{b-1}, x_b, s_{b-1}, and s_b inequalities (5) are fulfilled for $\mathbb{D} = [x_{b-1}, x_b]$. We do this by solving the unconstrained problem

$$z^{\max *} := \max_{x \in [x_{b-1}, x_b]} \left| \ell(x) - f(x) \right| \tag{82}$$

to global optimality. If $z^{\max *} \le \delta$, then we accept x_b and s_b. Otherwise, we increase I by a factor of $\beta > 1$. This algorithm stops when the whole interval $[X_-, X_+]$ is covered.

Algorithm 4.2 Forward Heuristic with Moving Breakpoints Computing a δ-Approximator

```
// INPUT: Continuous function f, scalar δ > 0, initial discretization size I^ini ∈ ℕ and parameter β > 1
// OUTPUT: Number of breakpoints, B, breakpoint system x_b and shift variable s_b
// Initialize
x_0 := X_−, I := I^ini/β, B := 0, and b = 1
// Outer loop
repeat
    // Inner loop
    repeat
        // update discretization size
        I ← ⌈βI⌉
        // calculate next breakpoint and shift variable
        solve NLP (75)-(78) to obtain x_b^*
        solve one-dimensional NLP (79)-(81) to obtain s_b^*
        // check if obtained ℓ is δ-approximator on [x_{b−1}, x_b]
        solve unconstrained NLP (82) to obtain z^{max*}
    until z^{max*} ≤ δ
    // fix breakpoint, shifting variable and update counter
    x_b := x_b^*, s_b := s_b^*, b ← b+1, B ← B+1
until x_b = X_b
```

This procedure is summarized in Algorithm 4.2. An efficient implementation of Algorithm 4.2 first checks whether or not the obtained $\Delta^{\mathrm{I}*} \le \delta$ before solving (79)-(81) and (82). Similar to the heuristic 4.1, the Algorithm 4.2 always terminates in finitely many steps (given exact arithmetics):

Corollary 8 *Algorithm 4.2 terminates after a finite number of iterations for any continuous function f, any $\delta > 0$, any initial discretization size $I^{\text{ini}} \in \mathbb{N}$ and parameter $\beta > 1$. The calculated breakpoints with the shift variables yield a δ-approximator for f.*

Proof We need to show that both the inner and the outer loop are finite.

The inner loop is finite due to Theorem 2 and Corollary 6: We know that there exists an interval $[x_{b-1}, x_b]$ for fixed x_{b-1} and fixed s_{b-1} (this follows from the proof of Theorem 2) with a linear δ-approximator function ℓ (because a δ-approximator exists for finitely many B). Furthermore, $z^{\max *}$ and Δ^{S*} can be arbitrarily close in a finite number of iterations.

The finiteness of the outer loop follows from the compactness of interval $[X_-, X_+]$ with the same argument as given in the proof of Corollary 8. □

Allowing the shift variable s_0 to vary between $-\delta$ and δ does not change any principles of Algorithm 4.2. However, the NLP (79)-(81) is then two-dimensional for the first breakpoint.

There are several advantages and disadvantages of both heuristic methods 4.1 and 4.2. While 4.1 needs to solve a much smaller number of optimization problems to global optimality than 4.2, the number of breakpoints of the δ-approximator computed by Algorithm 4.1 is expected to be larger than the one computed by Algorithm 4.2. Particularly computationally expensive is solving problems (75)-(78) in Algorithm 4.2.

Both Algorithms 4.1 and 4.2 are of a "forward" nature, *i.e.*, the interval $[X_-, X_+]$ is successively covered by intervals of breakpoints "moving" from X_- to X_+. Dependent on the shape of the function f and given that both methods are heuristics, it might be beneficial to run the algorithm in a "backwards" manner, *e.g.*, the obtained δ-approximator might have less breakpoints. To run both a forward and a backward algorithm might be particularly promising for functions which are highly asymmetric around $\frac{X_- + X_+}{2}$. Such a backward algorithm can be achieved by substituting $f(x)$ by $\tilde{f}(x) := f(X_+ + X_- - x)$ and running the forward Algorithm 4.1 for $\tilde{f}$ and $x \in [X_-, X_+]$. The breakpoint system for the backwards algorithm is then obtained as follows: Let x_b^* be the breakpoints obtained by the forward algorithm for $\tilde{f}(x)$. The new breakpoints are given by $\tilde{x}_b^* := X_+ + X_- - x_b^*$.

4.4 Deriving δ-Underestimators and δ-Overestimators

We have seen, in Section 3, that under- and overestimator play a crucial role when establishing safe bounds on unconstrained optimization problems as well as optimization problems with nonlinear, nonconvex functions in the constraint sets. For one-dimensional functions, the concept of under- and overestimators leads to piecewise linear δ-tubes around the function f.

Corollary 1 established that ε-underestimators for a continuous function f with minimal number of breakpoints can be calculated by shifting an $\frac{\varepsilon}{2}$-approximator for f with minimal number of breakpoints up by value $\frac{\varepsilon}{2}$. However, the optimality of the

Table 1: Changes to Algorithm 4.1 to compute a Δ-underestimator.

	δ-Approximator	**Δ-Underestimator**
Shift variables (line 15)	$s_b \in [-\delta, +\delta]$ $s_{bd} := \left(\frac{2d}{D+1} - 1\right)\delta$	$s_b \in [-\delta, 0]$ $s_{bd} := \left(\frac{d}{D} - 1\right)\delta$
Optimization problem (line 17)	(70)	$z_-^{\max*} := \max_{x\in[x_{b-1},x_b]} \left(f(x) - \ell(x)\right)$ and $z_-^{\min*} := \min_{x\in[x_{b-1},x_b]} \left(f(x) - \ell(x)\right)$
Stopping criteria (line 18)	$\Delta_b \leq \delta$ or $d = D$	$\left(z_-^{\max*} \leq \delta \text{ and } z_-^{\min*} \geq 0\right)$ or $d = D$
Stopping criteria (line 19)	$\Delta_b \leq \delta$	$z_-^{\max*} \leq \delta$ and $z_-^{\min*} \geq 0$

Table 2: Changes to Algorithm 4.2 to compute a Δ-underestimator.

	δ-Approximator	**Δ-Underestimator**
Optimization problem (line 12)	(75)-(78)	$\Delta_-^{I*} := \max x_b$ s.t. $f(x_{bi}) - \ell(x_{bi}) \leq \delta \quad , \quad \forall i \in \mathbb{I}$ $\ell(x_{bi}) \leq f(x_{bi}) \quad , \quad \forall i \in \mathbb{I}$ $(76), (77), (78)$
Optimization problem (line 13)	(79)-(81)	$\Delta_-^{S*} := \max s_b$ s.t. $f(x_{bi}) - \ell(x_{bi}) \leq \delta \quad , \quad \forall i \in \mathbb{I}$ $\ell(x_{bi}) \leq f(x_{bi}) \quad , \quad \forall i \in \mathbb{I}$ $s_b \in [-\delta, 0]$
Optimization problem (line 15)	(82)	$z_-^{\max*} := \max_{x\in[x_{b-1},x_b]} \left(f(x) - \ell(x)\right)$ and $z_-^{\min*} := \min_{x\in[x_{b-1},x_b]} \left(f(x) - \ell(x)\right)$
Stopping criteria (line 16)	$z^{\max*} \leq \delta$	$z_-^{\max*} \leq \delta$ and $z_-^{\min*} \geq 0$

breakpoint system with respect to the number of breakpoints is lost in general, if the used $\frac{\varepsilon}{2}$-approximator for f does not have the minimal number of breakpoints.

Exact methods to compute optimal (in the sense of minimality of breakpoints or tightness) δ-approximators are very difficult to solve, thus heuristic methods may be the only choice to obtain a good set of breakpoints. Furthermore, computational performance of the models OBSC, OBSD and OBSI is crucially influenced by good upper bounds on the number of breakpoints; such upper bounds can be provided by heuristic methods. Thus, we present a tailored algorithm to construct under- and overestimators for one-dimensional functions.

The following discussion is focused on δ-underestimators. A δ-overestimator ℓ_+ for function f over D_3 can be obtained by constructing a δ-underestimator ℓ_- for function $\tilde{f}(x) := -f(x)$ and $x \in D_3$. Then, $\ell_+ := -\ell_-$ is a δ-overestimator for function f over D_3. Alternatively, the adjustments for δ-overestimators are straightforward.

The changes needed for Algorithms of Sections 4.3.1 and 4.3.2 to compute δ-underestimators are summarized in Tables 1 and 2.

4.5 Computational Results

We have implemented the presented models and algorithms using the modeling language GAMS version 23.6 (cf. Brooke et al. (1992, [7] or Bussieck and Meeraus (2004, [8]). The global optimization problems are solved using LindoGlobal version 23.6.5 ([33]). The computations are preformed by an Intel(R) i7 using a single core with 2.93 GHz and 12.0 GB RAM on a 64-bit Windows 7 operating system. We allow a maximal deviation from the δ-tube by at most 10^{-5}; *i.e.*, equation (5) is validated by at most 10^{-5}. The same is true for under- and overestimators; *i.e.*, equation (7) is validated by at most 10^{-5}.

As our computational test bed, we consider ten different functions, summarized in Table 3. The columns $\mathbf{X}_{-}$ and $\mathbf{X}_{+}$ define the lower and upper bound, respectively, of the compact interval D_1. We summarize relevant characteristics for our purposes for each function in the last column of the table.

Table 3: One-dimensional functions tested.

#	$\mathbf{f}(\mathbf{x})$	$\mathbf{X}_{-}$	$\mathbf{X}_{+}$	Comment
1	x^2	-3.5	3.5	convex function, optimal distribution of breakpoints is uniform; axial symmetric at $x=0$
2	$\ln x$	1	32	concave function
3	$\sin x$	0	2π	point symmetric at $x=\pi$
4	$\tanh(x)$	-5	5	strictly monotonically increasing; point symmetric at $x=0$
5	$\frac{\sin(x)}{x}$	1	12	for numerical stability reason we avoid the removable singularity and the oscillation at 0, the two local minima have an absolute function value difference of ≈ 0.126
6	$2x^2+x^3$	-2.5	2.5	in (∞,∞), there is one local minimum at $x=0$ and one local maximum at $x=\frac{4}{3}$
7	$e^{-x}\sin(x)$	-4	4	one global minimum ($x_m \approx -2.356$ and $f(x_m) \approx -7.460$)
8	$e^{-100(x-2)^2}$	0	3	a normal distribution with a sharp peak at x=2
9	$1.03e^{-100(x-1.2)^2}+e^{-100(x-2)^2}$	0	3	the sum of two Gaussians, with two slightly different maxima (their absolute function value difference is ≈ 0.030)
10	Maranas & Floudas (1994)	0	2π	three local minima (the absolute function value difference of the two smallest local minima is ≈ 0.031)

Figure 3 illustrates the ten functions (black line) together with δ-approximators, δ-underestimators or δ-overestimators (gray line), obtained from different methods. Method FBSD is used to compute approximators for the first five functions. The number of breakpoints, B, is chosen a priori. FBSD is then used to compute the optimal δ^*, δ^*_- or δ^*_+ (with a precision of < 0.001) together with an estimator. Estimators for functions six to ten are computed with the heuristic methods Alg. 1 and Alg. 2, where

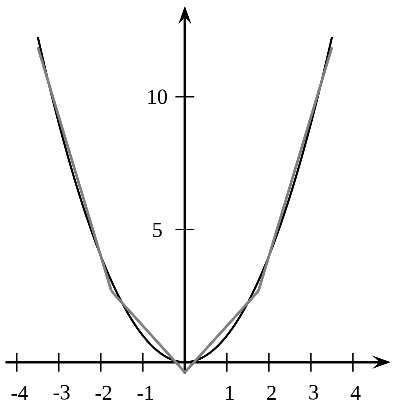
δ was chosen a priori. One can clearly see (*e.g.*, in Fig. 3(h)-(j)) that our models do not compute approximators which are "closest" possible to the original function but which instead stay within a given δ-tube around the function.

For each function and four different values of $\delta \in \{0.100, 0.050, 0.010, 0.005\}$, the number of breakpoints and the computational times for the two heuristic methods, presented in Sections 4.3.1 and 4.3.2, are summarized in Table 4. Both heuristic methods are executed in a forward and backwards fashion. One observes that the number of breakpoints and the computational times are similar for both the forward and the backward iterations. However, the running time of Alg. 2 is significantly higher than the running time of Alg. 1. This comes as no surprise because Alg. 1 requires less NLP solves compared to Alg. 2. Alg. 2 consistently computes the same or fewer number of breakpoints for a given accuracy δ than Alg. 1.

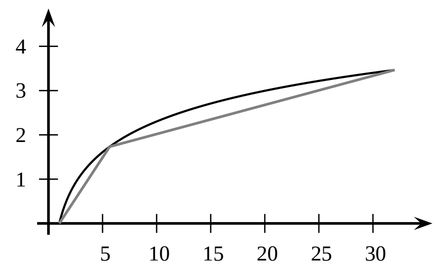
The following parameters are a good trade-off between computational time and number of breakpoints computed:

Alg. 1: $\alpha = 0.985$ and $D = 3$
Alg. 2: $I^{\text{ini}} = 10$ and $\beta = 2.5$

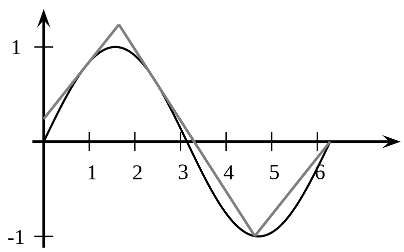
Computational results for δ-under- and δ-overestimators for both heuristic methods are presented in Table 5. One observes that the number of breakpoints calculated is consistent with the ones obtained by the $\frac{\delta}{2}$-approximators, *cf.* to Table 4. However, the computational times increase significantly compared to the computations of the $\frac{\delta}{2}$-approximators. This is explained by the additional NLP to be solved to check condition (7).



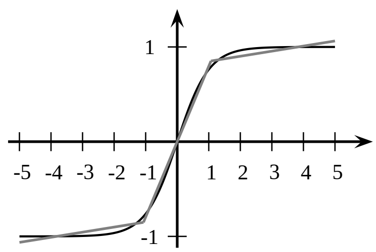
(a) Func 1: FBSD with $B = 5$ yields $\delta^* = 0.383$



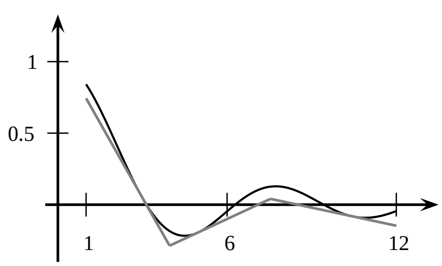
(b) Func. 2: FBSD with $B = 3$ yields $\delta^*_- = 0.361$



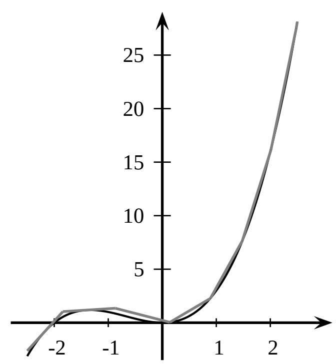
(c) Func. 3: FBSD with $B = 4$ yields $\delta^*_+ = 0.240$



(d) Func. 4: FBSD with $B = 4$ yields $\delta^* = 0.063$



(e) Func. 5: FBSD with $B = 4$ yields $\delta^*_- = 0.103$

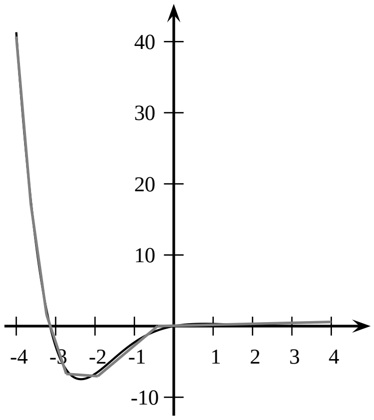


(f) Func. 6: Alg. 1 with $\delta_+ = 0.5$ yields $B = 9$

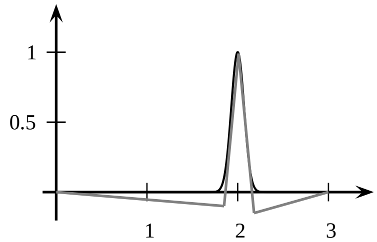
Fig. 3: Continued.

Table 4: Computational results for δ-approximators using heuristics.

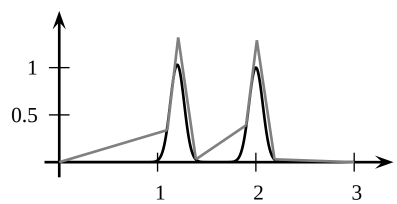
		Algorithm 4.1				**Algorithm 4.2**			
		Forward		**Backward**		**Forward**		**Backward**	
#	δ	**B**	**sec.**	**B**	**sec.**	**B**	**sec.**	**B**	**sec.**
1	0.100	9	0.41	9	0.41	9	2.69	9	2.64
	0.050	13	0.58	13	0.57	13	3.98	13	4.06
	0.010	26	1.18	26	1.23	26	8.85	26	9.10
	0.005	37	1.71	37	1.70	**36**	10.46	**36**	10.99
2	0.100	4	0.21	4	0.16	4	1.65	4	1.73
	0.050	5	0.35	5	0.21	5	1.20	5	1.26
	0.010	10	0.68	10	0.45	10	3.31	10	3.01
	0.005	14	0.69	14	0.66	14	5.90	14	4.96
3	0.100	6	0.27	6	0.28	6	31.29	6	35.30
	0.050	6	0.26	6	0.27	6	4.35	6	4.89
	0.010	14	0.70	14	0.69	14	5.47	14	5.77
	0.005	18	0.84	18	0.85	18	7.43	18	7.68
4	0.100	4	0.17	4	0.16	4	20.61	4	0.61
	0.050	6	0.26	6	0.29	6	1.67	6	1.83
	0.010	10	0.49	10	0.45	10	3.71	10	3.84
	0.005	14	0.72	14	0.73	14	5.40	14	5.64
5	0.100	5	5.60	**4**	0.21	5	34.10	**4**	63.94
	0.050	6	1.04	6	0.44	6	46.20	6	93.47
	0.010	11	1.43	10	0.61	10	11.31	10	272.19
	0.005	13	0.82	13	2.01	13	12.08	13	12.38
6	0.100	12	0.77	12	0.64	12	23.09	12	17.74
	0.050	16	1.00	16	0.86	16	17.66	16	20.40
	0.010	35	2.16	35	2.26	35	22.48	35	41.87
	0.005	49	3.10	49	3.19	**48**	28.34	**48**	30.38
7	0.100	15	0.97	15	0.93	15	40.57	15	31.62
	0.050	21	1.48	21	2.36	21	28.73	**20**	51.40
	0.010	45	2.88	**44**	2.95	45	53.22	**44**	51.54
	0.005	62	4.11	62	4.37	62	72.87	62	62.29
8	0.100	5	0.30	5	0.26	5	8.43	5	6.09
	0.050	7	0.50	7	0.39	7	10.52	7	7.56
	0.010	12	0.74	12	0.73	12	6.90	12	6.50
	0.005	16	0.97	16	1.00	**15**	7.77	16	10.43
9	0.100	8	0.47	8	0.44	8	11.67	8	14.93
	0.050	13	0.85	12	0.74	12	18.70	12	19.66
	0.010	22	1.41	22	1.39	22	13.66	22	15.98
	0.005	30	2.15	**29**	2.07	**29**	17.94	**29**	16.92
10	0.100	17	2.90	17	3.03	17	204.11	17	97.87
	0.050	23	4.04	23	4.11	23	88.50	23	98.57
	0.010	**46**	7.82	47	7.61	**46**	91.71	47	95.95
	0.005	68	11.17	68	11.17	68	88.13	**67**	96.22
∅	0.100		1.21		0.65		37.82		27.25
∅	0.050		1.04		1.02		22.15		30.31
∅	0.010		1.95		1.84		22.06		50.58
∅	0.005		2.63		2.78		25.63		25.79



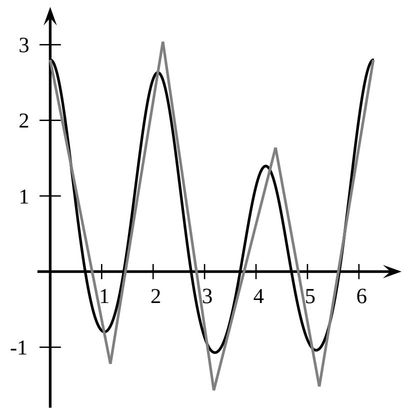
(g) Func. 7: Alg. 2 with $\delta = 0.6$ yields $B = 7$



(h) Func. 8: Alg. 1 with $\delta_- = 0.2$ yields $B = 5$



(i) Func. 9: Alg. 2 with $\delta_+ = 0.3$ yields $B = 8$



(j) Func. 10: Alg. 1 with $\delta = 0.5$ yields $B = 7$

Fig. 3: The ten univariate functions tested together with some approximator functions.
— original function
— approximator function

Table 5: Computational results for δ-under- and δ-overestimators. Optimal δ-under- and δ-overestimators can be obtained by the solutions as shown in Table 3 with double precision, *i.e.*, half the δ values.

#	δ	**Underestimator**								**Overestimator**							
		Algorithm 4.1				**Algorithm 4.2**				**Algorithm 4.1**				**Algorithm 4.2**			
		Forward		**Backward.**		**Forward**		**Backward**		**Forward**		**Backward.**		**Forward**		**Backward**	
		B	**sec.**	**B**	**sec.**	**B**	**sec.**	**B**	**sec.**	**B**	**sec.**	**B**	**sec.**	**B**	**sec.**	**B**	**sec.**
1	0.10	13	3.50	13	4.25	13	9.70	13	11.08	13	1.55	13	1.68	13	4.94	13	5.16
	0.02	26	9.99	26	9.91	26	23.20	26	22.98	26	2.17	26	2.20	26	10.25	26	10.79
	0.01	37	14.46	37	14.81	36	37.15	36	36.23	37	3.85	37	3.97	36	14.80	36	15.99
2	0.10	5	0.39	5	0.35	5	1.25	5	1.39	5	1.60	5	1.19	5	7.46	5	11.66
	0.02	10	0.81	10	1.22	10	1.90	10	2.42	10	18.17	10	10.54	10	19.71	10	23.40
	0.01	14	1.20	14	3.63	14	2.85	14	6.18	14	31.68	14	22.59	14	30.72	14	34.81
3	0.10	6	0.51	6	0.45	6	5.42	6	5.49	6	0.44	6	0.57	6	5.47	6	5.82
	0.02	14	1.81	14	2.03	14	8.63	14	7.61	14	2.14	14	2.00	14	8.14	14	8.06
	0.01	18	2.47	18	3.13	18	10.08	18	9.53	18	3.03	18	2.74	18	10.23	18	10.04
4	0.10	6	0.51	6	0.76	6	3.14	6	2.91	6	0.74	6	0.65	6	3.43	6	2.27
	0.02	10	2.93	10	2.11	10	7.10	10	8.22	10	2.12	10	2.44	10	8.65	10	8.26
	0.01	14	5.51	14	4.44	14	9.63	14	9.20	14	4.96	14	4.90	14	10.38	14	8.30
5	0.10	6	0.79	6	1.08	6	53.29	6	8.36	6	0.78	6	1.19	6	10.46	6	5.22
	0.02	11	3.08	10	3.19	10	9.23	10	196.31	11	2.96	10	1.41	10	7.48	10	9.94
	0.01	13	3.43	13	3.49	13	10.15	13	8.15	13	2.57	13	2.41	13	35.21	13	9.89
6	0.10	16	1.83	16	1.91	16	18.31	16	21.21	16	1.56	16	1.55	16	16.03	16	19.13
	0.02	35	4.84	35	3.97	35	25.89	35	30.81	35	3.74	35	3.95	35	22.11	35	27.95
	0.01	49	7.30	48	5.57	48	36.74	48	37.06	49	5.40	48	5.07	48	31.14	48	35.62
7	0.10	21	5.34	21	6.22	21	26.20	20	45.78	21	5.07	21	5.16	21	26.97	20	68.21
	0.02	45	11.31	44	11.18	45	49.46	44	43.75	45	11.14	44	10.66	45	47.54	44	42.02
	0.01	62	14.98	62	15.60	62	64.63	62	69.09	62	14.96	62	15.89	62	64.83	62	55.52
8	0.10	7	0.71	7	0.64	7	15.14	7	5.10	7	0.83	7	1.66	7	14.81	7	5.45
	0.02	12	1.21	12	1.12	12	6.68	12	5.59	12	1.24	12	1.21	12	6.19	12	10.50
	0.01	16	1.65	16	1.59	15	6.65	16	6.61	16	1.66	16	1.78	15	9.15	16	8.96
9	0.10	13	1.53	12	1.29	12	19.23	12	14.48	13	1.54	12	1.27	12	19.15	12	24.95
	0.02	22	2.44	22	2.67	22	11.35	22	13.25	22	2.69	22	2.73	22	11.20	22	17.49
	0.01	30	3.76	29	3.32	29	15.69	29	15.08	30	4.10	29	3.67	29	15.92	29	17.32
10	0.10	23	4.69	23	4.59	23	85.18	23	91.54	23	4.72	23	4.48	23	86.81	23	85.62
	0.02	46	9.45	47	9.74	46	102.37	47	98.85	46	9.11	47	9.54	46	93.00	47	95.07
	0.01	68	14.61	68	14.97	68	111.49	67	115.59	68	13.84	68	14.40	68	96.84	67	106.67

Table 6: Tightness obtained by FBSD for given B.

#	δ_{ini}	B	δ_{LB}	δ_{UB}	**Max Grid**	**Time**
1	0.100	9	0.095703	0.095703	1	16.4
	0.050	13	0.042535	0.042535	1	115.4
2	0.100	4	0.081899	0.081922	4	12.1
	0.050	5	0.046281	0.046595	4	25.8
	0.010	10	0.009211	0.009287	1	3.9
	0.005	14	0.004429	0.004446	1	68.7
3	0.100	6	0.048109	0.048250	19	411.0
	0.050	6	0.048109	0.048250	19	190.8
	0.010	14	0.009696	0.010275	9	5659.3
	0.005	18	0.004637	0.004829	6	11079.1
4	0.100	4	0.062853	0.063728	3	2.9
	0.050	6	0.024160	0.024541	3	20.2
	0.010	10	0.007855	0.008148	3	39.4
	0.005	14	0.003578	0.004409	2	27.8
5	0.100	4	0.051237	0.051847	13	182.4
	0.050	6	0.018513	0.022101	6	36162.8
6	0.100	4	0.085288	0.095080	9	108806.1
8	0.100	5	0.053910	0.054603	13	283.6
	0.050	7	0.009178	0.990842	13	36416.9
	0.010	12	0.009158	0.990842	9	42195.4
9	0.100	8	0.085773	0.941691	9	38712.4
	0.050	12	0.000087	1.029913	4	36324.0

All other instances yield a lower bound for δ^* of 0 after 10 hours of CPU time.

Table 6 summarizes the computational results obtained by FBSD (60)-(65). We use the lowest number of breakpoints calculated by any of the two heuristic methods for a given accuracy δ, *cf.* Table 4, to calculate the tightest possible approximator with the help of FBSD. We start with a grid size of $I = 1$ and solve FBSD. This yields a lower bound δ_{LB} on δ^* (for the fixed number of breakpoints). For the computed approximator, we evaluate the maximal deviation to the function $f(x)$. This yields an upper bound δ_{UB} on δ^*. If the upper bound and the lower bound are within 0.001, then we stop the algorithm. Otherwise, we increase I to $I \leftarrow \max\{1.5 \cdot I, I+1\}$ and re-iterate. δ_{ini} is used as a (tight) initial bound on the shift variables and the maximal deviation.

Table 7 summarizes the computational results for the model OBSD (38)-(47). We limit the size of the breakpoint set $\mathscr{B}$ by the lowest number of breakpoints computed in Table 4 for each discretization size δ. The continuum condition is initially discretize into two points, *i.e.*, $I = 2$. By solving OBSD, we obtain a lower bound $\mathbf{B}_-$ on $\mathbf{B}^*$. If $\mathbf{B}_-$ equals the initial number of breakpoints or the maximal deviation of the obtained approximator system does not exceed δ (with an accuracy of 0.00125), then the algorithm stops with $\mathbf{B}^* = \mathbf{B}_-$. Otherwise, the grid size is updated by $I \leftarrow 1.5 \cdot I$ and the process starts over again. One observes in Table 7 that for most of the problems B^* cannot be computed. Furthermore, the required discretization size I is quite large.

As expected, model OBSI (48)-(59) performs much worse compared to model OBSD. Only for function 5 with $\delta = 0.100$, OBSI is able to obtain the optimal $B^* = 4$.

Table 7: Computational results for model OBSD.

#	δ	$\mathbf{B}^*$	$\mathbf{B}_-$	# iter.	I	sec.
1	0.100	–	5	9	42	1965.25†
	0.050	–	5	8	28	1967.30†
	0.005	–	5	5	9	1997.34†
2	0.100	4	–	9	42	24.16
	0.050	5	–	10	63	550.41
	0.010	–	5	9	42	2236.04†
	0.005	–	5	8	28	2128.16†
3	0.100	6	–	11	94	195.33
	0.050	6	–	11	94	212.71
4	0.100	4	–	10	63	29.95
	0.050	–	5	11	94	2815.14†
	0.010	–	5	9	42	2427.14†
	0.005	–	5	8	28	2157.83†
5	0.100	4	–	9	42	150.40
	0.050	–	4	9	42	1877.28†
	0.010	–	5	8	28	2918.74†
	0.005	–	5	7	19	2832.80†
6	0.100	–	4	7	19	1871.30†
	0.050	–	4	7	19	1990.46†
	0.010	–	0	1	2	1801.24†
	0.005	–	0	1	2	1801.62†
7	0.100	–	5	8	28	2833.29†
	0.050	–	0	1	2	1800.99†
	0.010	–	0	1	2	1801.50†
	0.005	–	0	1	2	1802.60†
8	0.100	–	4	11	94	2224.76†
	0.050	–	4	10	63	2077.81†
	0.010	–	4	8	28	1836.07†
	0.005	–	5	8	28	2758.10†
9	0.100	–	4	9	42	2250.79†
	0.050	–	4	8	28	2082.88†
	0.010	–	4	6	13	1863.82†
	0.005	–	4	5	9	1828.28†
10	0.100	–	4	6	13	3617.24†
	0.050	–	4	5	9	3032.66†
	0.010	–	0	1	2	1804.50†
	0.005	–	0	1	2	1802.37†

†: time limit reached (1800 sec. per iteration)

The computational time is approximately 97 seconds, requiring a size of $I = 20$. For most of the other problem instances, not even a feasible solution for the original model (using $I = 2 \cdot B$) could be computed within 1800 seconds of CPU time.

Table 8 summarizes the optimal number of breakpoints required for the various functions and approximation accuracies along with the methods computed (again, we have a numerical accuracy of 10^{-5}). For 25 out of 40 instances, an optimal B^* can be computed, while for 15 instances, B^* is unknown. We do not report exact computational times in seconds, as different solver versions, different parameter settings and initial values on B are used for each of the computations. To prove optimality of B

Table 8: Benchmarks: Minimal number B^* of breakpoints needed for δ-approximators.

#	δ	B^*	**LB**	**UB**	**Algorithm**	**Time**
1	0.100	9			FBSD	few sec.
	0.050	13			FBSD	few sec.
	0.010	26			FBSD	hours
	0.005	36			FBSD	hours
2	0.100	4			FBSD	frac. sec.
	0.050	5			FBSD	few sec.
	0.010	10			FBSD	sec.
	0.005	14			FBSD	sec.
3	0.100	6			FBSD	few sec.
	0.050	6			FBSD	few sec.
	0.010	14			FBSD	sec.
	0.005	18			FBSD	few min.
4	0.100	4			FBSD	frac. sec.
	0.050	6			FBSD	few sec.
	0.010	10			FBSD	few sec.
	0.005	14			FBSD	few min.
5	0.100	4			FBSD	frac. sec.
	0.050	6			FBSD	sec.
	0.010	10			FBSD	sec.
	0.005	13			FBSD	few min.
6	0.100	12			FBSD	min.
	0.050	16			FBSD	few days
	0.010		16	35		
	0.005		16	48		
7	0.100		5	15	OBSD	
	0.050		5	20		
	0.010		5	44		
	0.005		5	62		
8	0.100	5			FBSD	sec.
	0.050		5	7		
	0.010		5	12		
	0.005		5	15		
9	0.100	8			FBSD	few days
	0.050		8	12		
	0.010		8	22		
	0.005		8	29		
10	0.100		4	17	OBSD	
	0.050		4	23		
	0.010		4	46		
	0.005		4	67		

LB: best known lower bound on B^*, only if B^* is unknown
UB: best known upper bound on B^*, only if B^* is unknown
frac.: $\geq \frac{1}{10}$ and < 1
few: ≥ 1 and ≤ 10

with the help of FBSD, one computes the optimal δ^* for $B-1$. If a lower bound on δ^* is greater than δ, then the optimal number of breakpoints has to be $\geq B$.

Let us compare our results with breakpoint systems obtained in a straight forward manner: use a equidistant distribution of the breakpoints in the interval D_3 together with a function interpolation. Table 9 summarizes the minimum number

Table 9: Minimal number B^{E} of equidistant breakpoints needed for interpolator with δ accuracy.

#	$\delta = 0.100$			$\delta = 0.050$			$\delta = 0.010$			$\delta = 0.005$		
	B^{E}	B	δ^*	B^{E}	B	δ^*	B^{E}	B	δ^*	B^{E}	B	δ^*
1	13	9	0.0851	17	13	0.0479	36	26	0.0100	51	36	0.0049
2	23	4	0.0956	37	5	0.0480	96	10	0.0100	141	14	0.0050
3	8	6	0.0966	11	6	0.0489	24	14	0.0093	33	18	0.0048
4	6	4	0.0923	15	6	0.0378	32	10	0.0099	45	14	0.0049
5	7	4	0.0989	10	6	0.0450	21	10	0.0093	29	13	0.0048
6	25	12	0.0997	36	16	0.0474	78	35	0.0099	110	48	0.0050
7	77	15	0.0993	109	20	0.0492	241	44	0.0100	340	62	0.0050
8	19	5	0.0879	64	7	0.0465	151	12	0.0097	213	15	0.0050
9	46	8	0.0777	68	12	0.0481	151	22	0.0099	216	29	0.0049
10	33	17	0.0973	46	23	0.0495	103	46	0.0100	146	67	0.0050

of equidistant breakpoints needed to ensure a given accuracy δ. We computes these breakpoint systems with the following brute-force algorithm. Starting with two breakpoints, compute the maximal deviation of the approximator to the function $f(x)$. This is accomplished by solving an DNLP to global optimality. If the maximal deviation is less than or equal to δ (with a tolerance of 10^{-5}), then we have found the minimum number of breakpoints. Otherwise, increment the number of breakpoints and start over. This leads to several order of magnitudes higher computational times than the reported times in Table 4; however, we decided not to report computation times because there might be more efficient algorithms and implementations to obtain the minimum number of equidistant breakpoints. Table 9 reports on the minimum number of equidistant breakpoints, B^{E}, and the actual maximal deviation, δ^*, of the interpolation function to $f(x)$. B^{E} is contrasted with the minimum number of breakpoints, B, computed with our methods. For a given δ, observe that the required number of equidistant breakpoints is between 1.3 and 14.2 times the actual number of breakpoints needed.

Fig. 4 plots the maximum deviation of the interpolation function for different number of equidistant breakpoints. The function is not monotonic decreasing but the tendency is clearly visible. The curve seems to follows an reciprocal logarithmic curve. Thus, the number of equidistant breakpoints grows exponentially in the reciprocal of δ.

5 Discussion

For univariate nonlinear functions we have constructed various methods to compute optimal breakpoint systems to be used for piecewise linear approximations as well as piecewise linear over- and underestimators satisfying a specified accuracy δ. The exact models and heuristic methods we have developed involve solving nonlinear problems to global optimality for proving that δ-accuracy is achieved for continuum intervals.

The breakpoints are useful to replace nonlinear terms in large MINLP problems which are mostly dominated by mixed-integer linear terms. This allows us to ap-

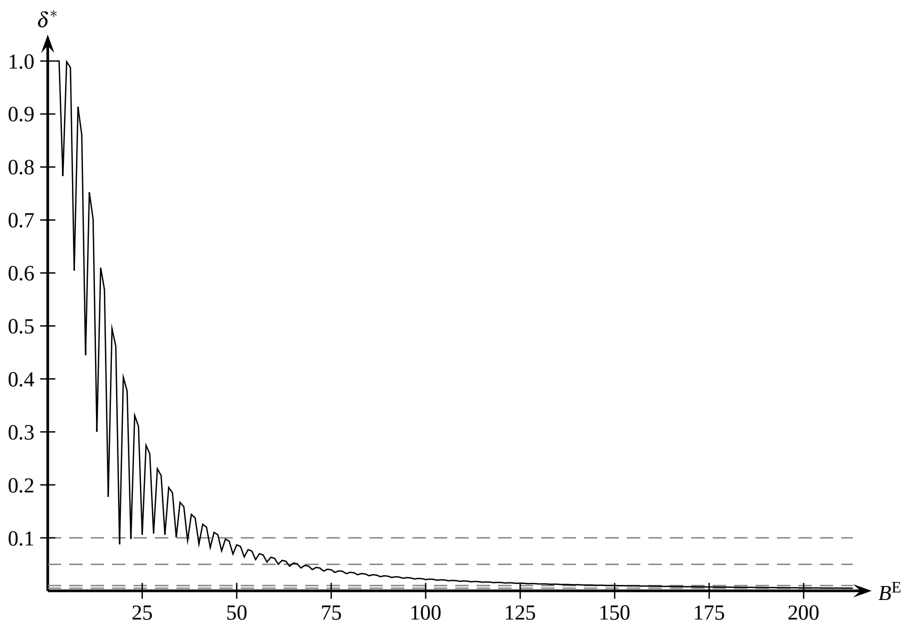


Fig. 4: Maximal deviation δ^* for different number of equidistant breakpoints B^{E} for function 8.

proximate a significant subset of NLP or MINLP problems just by MILP solvers. Approximator systems for NLP problems to be embedded in a large MILP problem framework should posses the following two desirable properties:

I solution with a given or known accuracy δ, and
II the resulting MILP problem has a relatively small number of breakpoints.

Property II forbids the usage of equidistant distribution of the breakpoints in the support interval, even when combining it with a global solver to ensure property I, *cf.* Table 9.

Note that the computation of the breakpoint systems is not restricted by available CPU time because they are computed only once a priori for later usage in large scale MILP problems which are very well restricted in the available CPU time.

Although CPU time is not critical, the problem size or properties of the functions to be approximated can lead to the situation that we cannot solve the breakpoint problem to global optimality. Therefore, we have developed several exact MINLP models and heuristic methods to obtain the optimal or at least very good breakpoint systems:

1) A MINLP model (OBSD) which yields the minimal number and best distribution of breakpoints for a given δ,
2) a MINLP model (FBSD) which gives the best approximation for a fixed number of breakpoints, and
3) two heuristic methods which compute the breakpoints, subsequently by solving MINLP problems with a small number of variables.

The heuristics always work, *i.e.*, even for complicated functions requiring large numbers of breakpoints we are able to obtain a breakpoint system satisfying the required δ, and more so, an upper bound on the minimal number of breakpoints. This upper bound can be used to solve 1) or 2) with a significant smaller number of variables. If 1) gives the proven minimal number of breakpoints, 2) can be used to compute the tightest δ-approximation.

Note that the measure to quantify the approximation quality is the maximal deviation between the approximator and function. In a forthcoming paper we define the best approximator as the one which satisfy the δ-criterion and minimizes the area between the approximator and the function.

Another research activity is the development of explicit, piecewise-linear formulations of functions $f(x):\mathbb{R}^n \rightarrow \mathbb{R}, n=1$, that are only defined at regular or irregular break points, but are not available in a closed algebraic form. This is an interesting problem relevant to various situations and industries. Such situations occur if the functions are evaluated by complex black box models involving, for instance, differential equations, or if the functions have been established only by experiments or observations. An important subtask is also to reduce the number of break points, *i.e.*, to replace them by a coarser grid which, relative to the system of given break points, preserves δ-accuracy.

Acknowledgements We thank Jan Jagla (GAMS GmbH, Cologne) for discussions on bi-level programs and Dr. Alexander Mitsos (MIT, Boston) for his favorable comments related to the SIP nature of our problem, Timo Lohmann and Greg Steeger for their careful proof-reading, and GAMS GmbH (Cologne) for providing an evaluation license of the global solver Lindoglobal.

References

1. E. L. M. Beale. Two Transportation Problems. In *Proceedings of the Third International Conference on Operational Research 1963*, pages 780–788, London, 1963. Dunod, Paris and English Universities Press.
2. E. L. M. Beale and J. A. Tomlin. Special Facilities in a General Mathematical Programming System for Nonconvex Problem Using Ordered Sets of Variables. In J. Lawrence, editor, *Proceedings of the Fifth International Conference on Operational Research 1969*, pages 447–454, London, 1970. Tavistock Publishing.
3. E. M. L. Beale and J. J. H. Forrest. Global Optimization Using Special Ordered Sets. *Mathematical Programming*, 10:52–69, 1976.
4. M. L. Bergamini, I. Grossmann, N. Scenna, and P. Aguirre. An improved piecewise outer-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms. *Computers and Chemical Engineering*, 32:477–493, 2008.
5. J. W. Blankenship and J. E. Falk. Infinitely Constrained Optimization Problems. *Journal of Optimization Theory and Applications*, 19:268–281, 1976.
6. B. Borchers and J. E. Mitchell. A Computational Comparison of Branch and Bound and Outer Approximation Algorithms for 0-1 Mixed Integer Nonlinear Programs. *Computers and Operations Research*, 5:186–204, 1997.
7. A. Brooke, D. Kendrick, and A. Meeraus. *GAMS – A User's Guide (Release 2.25)*. Boyd & Fraser Publishing Company, Danvers, Massachusetts, 1992.
8. M. R. Bussieck and A. Meeraus. General Algebraic Modeling System (GAMS). In J. Kallrath, editor, *Modeling Languages in Mathematical Optimization*, pages 137–157. Kluwer Academic Publishers, Norwell, MA, 2004.
9. I. R. de Farias Jr., E. L. Johnson, and G. L. Nemhauser. A Generalized Assignment Problem with Special Ordered Sets: a Polyhedral Approach. *Mathematical Programming Ser. A*, 89:187–203, 2000.

10. I. R. de Farias Jr., M. Zhao, and H. Zhao. A Special Ordered Set Approach for Optimizing a Discontinuous Separable Piecewise Linear Function. *Operations Research Letters*, 36:234–238, 2008.
11. M. A. Duran and I. E. Grossmann. An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programms. *Mathematical Programming*, 36:307–339, 1986.
12. R. Fletcher and S. Leyffer. Solving Mixed Integer Nonlinear Programs by Outer Approximation. *Mathematical Programming*, 66:327–349, 1994.
13. C. A. Floudas. *Deterministic Global Optimization: Theory, Algorithms and Applications*, volume 37 of *Nonconvex Optimization and Its Applications*. Kluwer Academic Publishers, 2000. 309–554.
14. C. A. Floudas, I. G. Akrotirianakis, S. Caratzoulas, C. A. Meyer, and J. Kallrath. Global Optimization in the 21st Century: Advances and Challenges for Problems with Nonlinear Dynamics. *Computers and Chemical Engineering*, 29:1185–1202, 2005.
15. C. A. Floudas and P. M. Pardalos, editors. *Frontiers in Global Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2004.
16. B. Geißler. *Towards Globally Optimal Solutions for MINLPs by Discretization Techniques with Applications in Gas Network Optimization*. Dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen-Nürnberg, Germany, 2011.
17. B. Geißler, A. Martin, A. Morsi, and L. Schewe. Using Piecewise Linear Functions for Solving MINLPs. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 287–314. Springer, 2012.
18. I. Grossmann. Review of Nonlinear andMixed Integer and Disjunctive Programming Techniques. *Optimization and Engineering*, 3:227–252, 1993.
19. R. Hettich and K. O. Kortanek. Semi-infinite Programming. *SIAM Review*, 35:380–429, 1993.
20. R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, 2nd edition, 2000.
21. J. Kallrath. Combined Strategic and Operational Planning - An MILP Success Story in Chemical Industry. *OR Spectrum*, 24(3):315–341, 2002.
22. J. Kallrath and T. I. Maindl. *Real Optimization with SAP-APO*. Springer, Heidelberg, Germany, 2006.
23. J. Kallrath and J. M. Wilson. *Business Optimisation Using Mathematical Programming*. Macmillan, Houndmills, Basingstoke, UK, 1997.
24. P. Kesavan, R. J. Allgor, E. P. Gatzke, and P. I. Barton. Outer Approximation Algorithms for Separable Nonconvex Mixed-integer Nonlinear Programs. *Mathematical Programming*, 100:517–535, 2004.
25. S. Leyffer, A. Sartenaer, and E. Wanufelle. Branch-and-refine for mixed-integer nonconvex global optimization, 2008.
26. L. Liberti and N. Maculan, editors. *Global Optimization: From Theory to Implementation*, volume 84 of *Nonconvex Optimization and Its Applications*. Springer, 2006. 223–232.
27. M. Lopez and G. Still. Semi-infinite Programming. *European Journal of Operational Research*, 180:491–518, 2007.
28. G. P. McCormick. Computatbility of Global Solutions to Factorable Nonconvex Programs: Part I - Convex Underestimators Problems. *Mathematical Programming*, 10:147–175, 1976.
29. R. Misener and C. A. Floudas. Piecewise-Linear Approximations of Multidimensional Functions. *Journal of Optimization Theory and Applications*, 145:120–147, 2010.
30. P. M. Pardalos and J. B. Rosen. *Constrained Global Optimization: Algorithms and Applications*. Lecture Notes in Computer Science. Springer, 1987.
31. S. Rebennack and J. Kallrath. Continuous Piecewise Linear δ-Approximations for MINLP Problems. II. Bivariate and Multivariate Functions. submitted, 2012.
32. J. B. Rosen and P. M. Pardalos. Global minimization of large-scale constrained concave quadratic problems by separable programming. *Mathematical Programming*, 34:163–174, 1986.
33. L. Schrage. LindoSystems: LindoAPI, 2004.
34. M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*, volume 65 of *Nonconvex Optimization And Its Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.
35. M. Tawarmalani and N. V. Sahinidis. Global Optimization of Mixed Integer Nonlinear Programs: A Theoretical and Computational Study improve MIP Solutions. *Mathematical Programming*, 99:563–591, 2004.
36. J. A. Tomlin. Special Ordered Sets and an Application to Gas Supply Operating Planning. *Mathematical Programming*, 45:69–84, 1988.

37. J. P. Vielma, S. Ahmed, and G. Nemhauser. Mixed-Integer Models for Nonseparable Piecewise-Linear Optimization: Unifying Framework and Extensions. *Operations Research*, 53:303–315, 2009.
38. J. P. Vielma and G. Nemhauser. Modeling Disjunctive Constraints with a Logarithmic Number of Binary Variables and Constraints. *Mathematical Programming*, 128:49–72, 2011.