Noname manuscript No.
(will be inserted by the editor)

Continuous Piecewise Linear Delta-Approximations for Univariate Functions: Computing Minimal Breakpoint Systems

Steffen Rebennack · **Josef Kallrath**

November 12, 2014

Abstract For univariate functions, we compute optimal breakpoint systems subject to the condition that the piecewise linear approximator, 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 allowing for the computation of an optimal piecewise linear, continuous approximator, under- and overestimator. We develop three non-convex optimization models: two yield the minimal number of breakpoints, and another in which, for a fixed number of breakpoints, the breakpoints are placed such that the maximal deviation is minimized. Alternatively, we use two heuristics which com-

S. Rebennack (✉)

Division of Economics and Business, Colorado School of Mines, Golden, Co, USA

E-mail: srebenna@mines.edu

J. Kallrath

Department of Astronomy, University of Florida, Gainesville, FL, USA

E-mail: kallrath@astro.ufl.edu

pute the breakpoints subsequently, solving small non-convex problems. We present computational results for ten univariate functions. Our approach computes breakpoint systems with up to one order of magnitude less breakpoints compared to an equidistant approach.

Keywords Global optimization · nonlinear programming · mixed-integer nonlinear programming · non-convex optimization

Mathematics Subject Classification (2000) 90C26

1 Introduction

We are interested in computing piecewise linear, continuous functions. These functions should approximate a given non-convex function such that the maximal, absolute deviation from the approximator to the non-convex function does not exceed a pre-defined tolerance δ. We call such a piecewise linear function a δ-approximator. The goal of this paper is to develop algorithms for univariate functions which can compute such δ-approximators using a minimal number of breakpoints.

The δ-approximators are useful to approximate a nonlinear programming problem (NLP) or a mixed-integer nonlinear programming problem (MINLP) by a mixed-integer linear programming problem (MILP). These δ-approximators have to be constructed carefully such that valid bounds on the original (MI)NLP can be recovered from the approximated MILP. Such MILP representations are of particular interest, if the (MI)NLP 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 (MI)NLP 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 this work, are those production planning and distributions problems with additional design aspects [1,2]. (2) Power system optimization problems involving (highly) non-convex constraint systems due to gas or electricity networks [3–5].

The modeling of such piecewise linear functions is closely related to *special ordered sets*. Ref. [6] 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 in Ref. [7], but already used earlier by Beale in [8] to deal with piecewise linear functions. Ref. [9] presents the idea of linear approximations to compute the global minimum of non-convex nonlinear functions using non-negative variables forming an SOS-2 set. The variables contained in an SOS-2 set are subject to the condition that at most two of them can have a non-zero value and the two non-negative variables can only occur for adjacent indices. Beale and Forrest develop efficient branching schemes to exploit this structure. Since 1976, various contributions elaborated on the usage of SOS-2:

– optimizing a discontinuous separable piecewise linear function [10,11],

- constructing a Branch-and-Refine algorithm for mixed-integer non-convex global optimization [12],
- developing a unifying framework and extensions to mixed-integer models for nonseparable piecewise linear optimization problems [13],
- using significantly fewer binary variables growing only logarithmically in the number of breakpoints [14].

All publications above use a *given* set of breakpoints, *i.e.*, the piecewise linear approximators are known.

Given these latest developments in the representation of piecewise linear functions, 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. Also recall that we aim for tight approximators with a guaranteed accuracy by exploiting the placement of breakpoints as a degree of freedom. The framework in [14] profits from tight approximators greatly: For the same number of breakpoints and constraints, we can expect to have (better) bounds on the original (MI)NLP when using tight approximators.

Next, we review two bodies of work, dealing with the computation of such piecewise linear approximators. The first work is by Rosen and Pardalos [15,16]. They proposed piecewise linear interpolators using equidistant breakpoints for concave quadratic minimization problems. 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, [15] is the first work which allows for the computation of breakpoints for a given error tolerance. 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 for a given accuracy.

The second body of work is by Geißler and co-workers [17, 18]. They 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 non-convex 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 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 while we compute continuous ones.

Ensuring that the approximator and the original function do not deviate more than δ from each other, leads to sets of constraints which have to hold over a continuum, resulting in semi-infinite programming (SIP) problems [19–21]. We evaluate this continuum conditions at discrete points, followed by a test involving the computation of a global maximum of the deviation function. If the test fails, we refine the grid [22].

The contributions of this paper are various methods to systematically construct optimal or "good" breakpoint systems, for univariate functions. More specifically:

1. We develop algorithms which compute the *proven* minimal number of breakpoints required to piecewise linearly and continuously approximate, under- or overestimate 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.

The remainder of the paper is organized as follows: We start with the definition of δ-approximators, δ-under- and δ-overestimators in Section 2. We discuss exact models in Section 3 and heuristics in Section 4 to construct such approximators. In Section 5, we present our computational results. Finally, we conclude in Section 6.

This paper is continued by a second paper discussing bivariate functions and transformations of multivariate functions to lower dimensional functions [23].

2 Approximators, Under- and Overestimators

In one dimension, we call a continuous function ℓ over a compact interval $\mathbb{D} \subset \mathbb{R}$ piecewise linear, if there are finitely many intervals partitioning $\mathbb{D}$ (we are particularly interested in partitions whose intervals intersect in at most one point), such that the restriction of ℓ on each interval yields an affine function. We call the two end-points of each interval a *breakpoint*. As such, any function f has at least two breakpoints.

Definition 2.1 (δ-approximator) Let $f : \mathbb{D} \to \mathbb{R}$ be a function on the compact interval $\mathbb{D} \subset \mathbb{R}$ and let scalar $\delta > 0$. A piecewise linear, continuous function $\ell : \mathbb{D} \to \mathbb{R}$ is called a δ-*approximator* for f, iff the following property holds

$$\max_{x \in \mathbb{D}} |\ell(x) - f(x)| \leq \delta. \tag{1}$$

For any continuous function f on the compactum $\mathbb{D}$ and any constant δ, there exists such a δ-approximator function [24]. 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 (1), *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 – it is *NP-hard* to determine a global extremum of an arbitrary, continuous function f [25]. (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*.

Under- and overestimators are defined as follows:

Definition 2.2 (δ-underestimator / δ-overestimator) Let scalar $\delta > 0$. We call function $\ell : \mathbb{D} \to \mathbb{R}$ on the compact interval $\mathbb{D} \subset \mathbb{R}$ a δ-*underestimator* of function $f : \mathbb{D} \to \mathbb{R}$, iff condition (1) is satisfied along with

$$\ell(x) \leq f(x) \quad \forall x \in \mathbb{D}. \tag{2}$$

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

The existence of ε-underestimator / ε-overestimator is also ensured for any continuous function f on the compactum $\mathbb{D}$, by using $\delta = \frac{\varepsilon}{2}$ and shifting the constructed δ-approximator by δ down / up. This procedure sustains the minimality of the number of breakpoints:

Corollary 2.1 *Let $\mathbb{D} \subset \mathbb{R}$ be a compact interval, $\ell : \mathbb{D} \to \mathbb{R}$ be a δ-approximator for $f : \mathbb{D} \to \mathbb{R}$ with a minimal number of breakpoints 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 breakpoints.*

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

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

Definition 2.3 (tightness) A δ-approximator, δ-underestimator or δ-overestimator with B breakpoints for function f is called *tighter* than a ϑ-approximator, ϑ-underestimator or ϑ-overestimator, respectively, with B breakpoints for function f, iff $\delta < \vartheta$. A δ-approximator, δ-underestimator or δ-overestimator with B breakpoints is called *tight* for $f(x)$, iff 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 2.2 *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 breakpoints.*

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*. However, we are also interested in minimizing the area between ℓ and f. Thus, ideally, one should compute

1. the minimum number of breakpoints, B^*, to achieve the δ-approximation, then
2. find a tight ϑ-approximator with B^* breakpoints ($\vartheta \leq \delta$), and then
3. compute a ϑ-approximator with B^* breakpoints 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. The computation of area-minimizing approximators is treated in [26].

Note that all definitions and results in this section naturally extend to n-dimensional functions.

3 Univariate Functions: Exact Approaches

In this section, we discuss the construction of breakpoint systems for one-dimensional functions $f : \mathbb{D} \to \mathbb{R}$ for the compact interval $\mathbb{D} := [X_-, X_+]$.

3.1 Computing an Optimal Set of Breakpoints

We are looking for a piecewise linear, continuous function $\ell : \mathbb{D} \to \mathbb{R}$ that satisfies condition (1), *i.e.*, a δ-approximator for f, which contains the minimal number of 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 3.2.

We allow the linear approximator to deviate $s_b \in [-\delta,+\delta]$ from the function values $f(x_b)$. Once, we have computed x_b and s_b, we can approximate function f by

$$f(x) = \sum_b (f(x_b)+s_b)\,\lambda_b \quad \text{with} \quad x = \sum_b x_b\lambda_b \quad \text{and} \quad \sum_b \lambda_b = 1.$$

For ease of notation, we define

$$\phi(x_b) := f(x_b) + s_b, \quad \forall b \in \mathscr{B}. \tag{3}$$

Now, we are able to construct a piecewise linear, continuous function ℓ (OBSC):

$$z^* = \min \sum_{b\in\mathscr{B}} \chi_b \tag{4}$$

$$\text{s.t.} \quad x_{b-1} \le x_b, \quad \forall b \in \mathscr{B} \tag{5}$$

$$x_b \ge X_- + (X_+ - X_-)\,(1-\chi_b), \quad \forall b \in \mathscr{B} \tag{6}$$

$$x_b - x_{b-1} \ge \frac{1}{M}\chi_b, \quad \forall b \in \mathscr{B} \tag{7}$$

$$x_b - x_{b-1} \le (X_+ - X_-)\,\chi_b, \quad \forall b \in \mathscr{B} \tag{8}$$

$$y_b = x_b - x_{b-1} + (X_+ - X_-)\,(1-\chi_b), \quad \forall b \in \mathscr{B} \tag{9}$$

$$\sum_{b\in\mathscr{B}} \chi^x_{bx} = 1, \quad \forall x \in [X_-,X_+] \tag{10}$$

$$x_{b-1} - (X_+ - X_-)\,(1-\chi^x_{bx}) \le x \le x_b + (X_+ - X_-)\,(1-\chi^x_{bx}),$$

$$\forall b \in \mathscr{B}, \quad \forall x \in [X_-, X_+] \tag{11}$$

$$\ell_b(x) := \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{y_b}(x - x_{b-1}),$$

$$\forall b \in \mathscr{B}, \quad \forall x \in [X_-, X_+] \tag{12}$$

$$\ell(x) := \sum_{b \in \mathscr{B}} \ell_b(x) \chi^x_{bx}, \quad \forall x \in [X_-, X_+] \tag{13}$$

$$|\ell(x) - f(x)| \leq \delta, \quad \forall x \in [X_-, X_+] \tag{14}$$

$$x_b \in [X_-, X_+], \quad s_b \in [-\delta, +\delta], \quad \chi_b \in \{0,1\}, \quad \chi^x_{bx} \in \{0,1\},$$

$$y_b \geq \frac{1}{M}, \quad \forall b \in \mathscr{B}, \quad \forall x \in [X_-, X_+] \tag{15}$$

where we define $x_0 := X_-$ and $\phi(x_b)$ according to (3).

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 (5) sort the breakpoints while (6) connect variables χ_b with the coordinates x_b of the breakpoints. Particularly, if $\chi_b = 0$, inequalities (6) 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_+. Moreover, if OBSC is feasible, then there must exist a breakpoint b such that $x_b = X_+$ with $\chi_b = 1$ and $\chi_{\tilde{b}} = 0$ for all $\tilde{b} > b$ and $\tilde{b} \in \mathscr{B}$, ensured by constraints (6) and (8). Note that the number of breakpoints included in ℓ is thus z^* +1, because the objective (4) does not count $x_0 = X_-$ as breakpoint for ℓ. Variables y_b take value $x_b - x_{b-1}$ if $x_b - x_{b-1} > 0$ and $X_+ - X_-$ otherwise. This is modeled via constraints (7)-(9) 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, modeled via constraints (10)-(11). The definitions (12)-(13) 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 constraints (10) and (14) turn our problem into the class of SIP. As formulation (4)-(15) leads to an Optimal Breakpoint System using a Continuum approach for x, we call it "OBSC." This discussion implies

Corollary 3.1 *If OBSC is feasible, then ℓ is a δ-approximator for f with the minimum number of breakpoints being $z^* + 1$.*

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 3.2 *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.*

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 – a class of optimization problems which are notoriously difficult to solve. To obtain a computationally tractable mathematical program, we discretize the continuum constraints (14) into I finite constraints of the form

$$|\ell(x_i) - f(x_i)| \leq \delta, \quad \forall i \in \mathbb{I} := \{1, \ldots, I\}, \tag{16}$$

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):

$$z^{D*} = \min \sum_{b\in\mathscr{B}} \chi_b \tag{17}$$

$$\text{s.t.} \quad (5)-(9) \tag{18}$$

$$x_{bi} = x_{b-1} + \frac{i}{I+1}(x_b - x_{b-1}), \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{19}$$

$$l_{bi} = \phi(x_{b-1}) + \frac{\phi(x_b)-\phi(x_{b-1})}{y_b}(x_{bi} - x_{b-1}), \quad \forall b \in \mathscr{B}, \forall i \in \mathbb{I} \tag{20}$$

$$|l_{bi} - f(x_{bi})| \le \delta, \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{21}$$

$$x_b \in [X_-, X_+], \quad s_b \in [-\delta, +\delta], \quad \chi_b \in \{0,1\}, \quad y_b \ge \frac{1}{M},$$

$$x_{bi} \in [X_-, X_+], \quad l_{bi} \text{ free}, \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I}, \tag{22}$$

with $x_B = X_+$. Decision variables x_{bi} uniformly discretize the breakpoint interval $[x_{b-1}, x_b]$ into $I+1$ segments, each with length $\frac{1}{I+1}(x_b - x_{b-1})$. This is modeled via (19). Variables l_{bi} evaluate the interpolation of $\phi(x_{b-1})$ and $\phi(x_b)$ at grid point x_{bi} through constraints (20). The maximal absolute deviation of the computed approximator to function $f(x)$ is then bounded by δ at the grid points through constraints (21), replacing constraints (14).

The number of variables and constraints of OBSD depends strongly on the number of breakpoints, B, and the discretization size I. Constraints (20) and (21) make problem OBSD a highly non-convex MINLP. 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 (20). For this function ℓ, one must solve

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

to global optimality. 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

Corollary 3.3 *Let OBSD be feasible for B and I. If ℓ constructed from (20) satisfies (1), then ℓ is a δ-approximator for f with the minimum number of breakpoints being $z^{D*}+1$. If ℓ does not satisfy (1), then $z^{D*}+1$ 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 the *entire* interval $[X_-, X_+]$ into I, a priori given, grid points (OBSI):

$$z^* = \min \sum_{b\in\mathscr{B}} \chi_b \tag{23}$$

$$\text{s.t.} \quad (5)-(9) \tag{24}$$

$$\sum_{b\in\mathscr{B}} \chi_{bi} = 1, \quad \forall i \in \mathbb{I} \tag{25}$$

$$x_{b-1} - (X_+ - X_-)(1-\chi_{bi}) \leq x_i \leq x_b + (X_+ - X_-)(1-\chi_{bi}), \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{26}$$

$$l_{bi} = \phi(x_{b-1}) + \frac{\phi(x_b)-\phi(x_{b-1})}{y_b}(x_i - x_{b-1}), \quad \forall b \in \mathscr{B}, \forall i \in \mathbb{I} \tag{27}$$

$$l_i = \sum_{b\in\mathscr{B}} \ell_{bi}\chi_{bi}, \quad \forall i \in \mathbb{I} \tag{28}$$

$$|l_i - f(x_i)| \leq \delta, \quad \forall i \in \mathbb{I} \tag{29}$$

$$x_b \in [X_-, X_+], \quad \chi_b \in \{0,1\}, \quad \chi_{bi} \in \{0,1\}, \quad y_b \geq \frac{1}{M}, \quad s_b \in [-\delta, +\delta], \quad l_b \text{ free}, \quad l_{bi} \text{ free}, \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{30}$$

where the $x_i = \frac{i}{I}(X_+ - X_-) + X_-$ are input data; $\phi(x_b)$ is obtained by (3) as previously. Binary decision variables χ_{bi} take value 1, if grid point $x_i \in [x_{b-1}, x_b]$ and 0 otherwise.

This is modeled by constraints (25) and (26), replacing (10) and (11). Constraints (27)-(29) model the approximator for the obtained breakpoint system.

Let us compare OBSD with OBSI. For one, OBSD does not require both the $B \cdot I$ binary variables χ_{bi} and constraints (25), (26), (28). Second, additional $B \cdot I$ continuous variables x_{bi} are introduced in the OBSD formulation, requiring constraints (19). Furthermore, constraints (20) involve the additional variables x_{bi} compared to constraints (27). Though binary variables tend to be computationally burdensome, non-convex terms are at least as computationally challenging. Thus, it is not a priori clear which formulation, OBSD or OBSI, is computationally superior.

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

Problems OBSC, OBSD and OBSI are in general too large and difficult to solve. Only for a modest number of breakpoints and not too many discretization points there is a chance to solve these problems to global optimality. Alternatively, we fix the number of breakpoints to $B + 1$ and compute an optimal breakpoint placement which minimized the deviation μ, obtained by the discretized continuum constraint

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

This is then followed by a check whether μ is less than or equal to our δ-tolerance.

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}. By forcing the usage of exactly B breakpoints (note, we do not count

$x_0 = X_-$ as breakpoint in the formulation), we can also eliminate the binary variables χ_b. We obtain the continuous NLP (FBSD)

$$\mu^* = \min \ \mu \tag{31}$$

$$\text{s.t.} \quad (19) - (21) \tag{32}$$

$$x_b - x_{b-1} \geq \frac{1}{M}, \quad \forall b \in \mathscr{B} \tag{33}$$

$$|l_{bi} - f(x_{bi})| \leq \mu, \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{34}$$

$$x_b \in [X_-, X_+], \quad x_{bi} \in [X_-, X_+], \quad l_{bi} \text{ free},$$

$$\mu \geq 0, \quad s_b \in [-\delta, +\delta], \quad \forall b \in \mathscr{B}, \quad \forall i \in \mathbb{I} \tag{35}$$

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 FBSD provides a breakpoint system x_b^*, the shift variables s_b^*, and the minimal value, μ^*. Note that they are functions of B and I, *e.g.*, $\mu^* = \mu^*(B,I)$ 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

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

Let δ^*-approximator be a tight approximator with $B+1$ 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 3.4 *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 3.4 does not provide a practical way of choosing I^*. Furthermore, $\mu^*(\cdot, I)$ is not a monotonic decreasing function in I. 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 3.4, 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.

4 Univariate Functions: Heuristic Approaches

In this section, we present two heuristic methods which respect the δ-tolerance. However, they cannot guarantee the minimality in the number of breakpoints.

4.1 Successively Computing a Good Set of Breakpoints

In Section 3.1, we provided formulations to compute all breakpoints simultaneously by solving one optimization model. Here, we propose a forward scheme moving successively from a given breakpoint, x_{b-1}, to the next breakpoint x_b with (BSB)

$$\zeta^* = \max \quad x_b \tag{36}$$

$$\text{s.t.} \quad \left| \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{x_b - x_{b-1}} (x - x_{b-1}) - f(x) \right| \leq \delta, \ \ \forall x \in [x_{b-1}, x_b] \tag{37}$$

$$x_b \in (x_{b-1}, X_+], \quad s_b \in [-\delta, +\delta]. \tag{38}$$

until the entire interval $[X_-, X_+]$ is covered. When BSB is solved and an optimal x_b^* as well as the shift 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 (37).

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 of 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 $x \in [0,5]$:

$$f(x) := \begin{cases} 1, & \text{if } x \in [0,2) \\ -0.50 + 0.75x, & \text{if } x \in [2,3) \\ 1.75 - \delta(x-3), & \text{if } x \in [3,4) \\ 1.75 - \delta + 2\delta(x-4), & \text{if } x \in [4,5] \end{cases}. \tag{39}$$

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.

4.1.1 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 this section fixes both x_b and the shift variables; they are decision variables in the heuris-

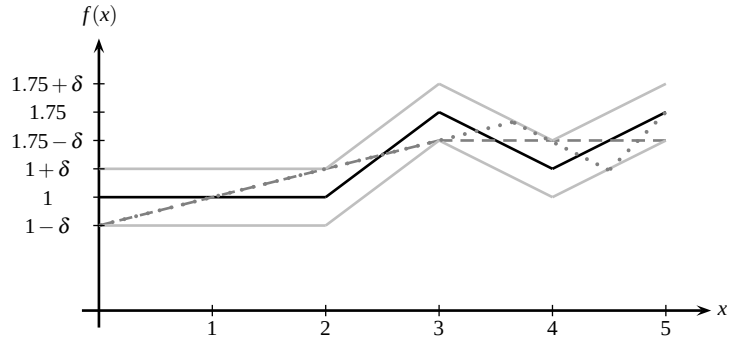


Fig. 1: Maximizing the length of the intervals successively is not optimal, in general

— $f(x)$ — δ-tube around $f(x)$ - - (unique) optimal δ-approximator

$\cdots$ δ-approximator maximizing interval length successively

tic presented in Section 4.1.2. We then need to check whether or not the obtained approximator satisfies $\Delta_b \leq \delta$, by solving

$$\Delta_b := \max_{x \in [x_{b-1}, x_b]} |\ell(x) - f(x)| \tag{40}$$

for interpolator

$$\ell(x) := \phi(x_{b-1}) + \frac{\phi(x_b) - \phi(x_{b-1})}{x_b - x_{b-1}} (x - x_{b-1}) \tag{41}$$

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

$$x_b \leftarrow x_{b-1} + \alpha(x_b - x_{b-1}), \quad 0 < \alpha < 1. \tag{42}$$

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

Algorithm 4.1 α-Forward Heuristic with Backward Iteration

1: // **INPUT:** Function f, scalar $\delta > 0$, parameter $\alpha \in (0,1)$, and shift variable discretization size D

2: // **OUTPUT:** Number of breakpoints, B, breakpoint system x_b and shift variables s_b

3: $x_0 := X_-$, $B := 0$, $b = 1$, and $s_0 := 0$ // Initialize

4: // **Outer loop**

5: **repeat**

6: $x_b := \frac{1}{\alpha} X_+ - \frac{1-\alpha}{\alpha} x_{b-1}$ // x_b equals X_+ after first counter update in line 9

7: // **Inner loop**

8: **repeat**

9: $x_b \leftarrow x_{b-1} + \alpha(x_b - x_{b-1})$ and $d := 0$ // update breakpoint and reset counter

10: **repeat**

11: $d \leftarrow d+1$ and $s_{bd} := \left(\frac{2d}{D+1} - 1\right)\delta$ // assign discretized value for shift variable

12: solve (40) with fixed x_{b-1}, x_b, s_{b-1} and s_{bd} to obtain Δ_b // optimize

13: **until** $\Delta_b \leq \delta$ or $d = D$

14: **until** $\Delta_b \leq \delta$

15: $s_b := s_{bd}$, $b \leftarrow b+1$, $B \leftarrow B+1$ // fix shift variable and update counter

16: **until** $x_b = X_+$

Corollary 4.1 *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, let $\tilde{\ell}(x)$ be a δ-approximator for $f(x)$ on $[x_{b-1}, X_+]$ with fixed shift s_{b-1} (as constructed by the algorithm) and condition $\tilde{\ell}(X_+) = f(X_+)$. Consider the continuous function $\tilde{d}(x) := \left|\tilde{\ell}(x) - f(x)\right|$ in $x \in [x_{b-1}, X_+]$. Let $\tilde{\delta} := \delta - \tilde{d}(x_{b-1})$. Given x_{b-1} and $\tilde{\delta} > 0$, then there exists $\eta > 0$ such that for all $x \in [x_{b-1}, x_{b-1} + \eta)$: $\tilde{d}(x) \in B_{\frac{\tilde{\delta}}{2}}\left(\tilde{d}(x_{b-1})\right)$ (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

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

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 $d(x) := |\ell(x) - f(x)| \leq \delta$ 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 $\min_{s_{bd}}\{|\frac{\tilde{\delta}}{2}|\} = \min_{s_{bd}}\{|\frac{\delta - s_{bd}}{2}|\} = \frac{\delta}{D+1} \geq \min_{s_{bd}}\{|s_{bd}|\}$.

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. □

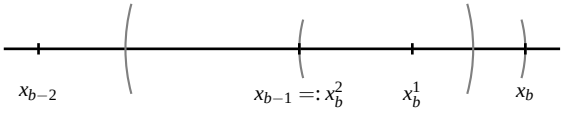


Fig. 2: Cover obtained for outer iteration b of the proof of Corollary 4.1

In order to avoid solving too many global optimization problems (40), we place I grid points, x_{bi}, according to (19) into the interval $[x_{b-1}, x_b]$. For each grid point, we

check whether or not

$$|\ell(x_{bi}) - f(x_{bi})| \leq \delta. \tag{43}$$

Only if condition (43) is satisfied for all grid points, we solve problem (40).

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 trade-off 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 (40).

4.1.2 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 3.2, we treat the continuum inequalities (37) by placing I grid points equidistantly into the interval $[x_{b-1}, x_b]$ according to (19). At these grid

393 points x_{bi}, we require:

$$|\ell(x_{bi}) - f(x_{bi})| \leq \delta. \tag{44}$$

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

396 Maximization of x_b leads to the following NLP

$$\Delta^{\mathrm{I}*} := \max \ \ x_b \tag{45}$$

$$\text{s.t.} \ \ |\ell(x_{bi}) - f(x_{bi})| \leq \delta, \quad \forall i \in \mathbb{I} \tag{46}$$

$$x_{bi} = x_{b-1} + \frac{i}{I+1}\left(x_b - x_{b-1}\right), \quad \forall i \in \mathbb{I} \tag{47}$$

$$x_b \in [x_{b-1}, X_+], \quad x_{bi} \in [x_{b-1}, X_+], \quad s_b \in [-\delta, \delta], \qquad \forall i \in \mathbb{I} \tag{48}$$

397 with the interpolator ℓ derived by (41).

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

$$\Delta^{\mathrm{S}*} := \min \ \ |s_b| \tag{49}$$

$$\text{s.t.} \ \ |\ell(x_{bi}) - f(x_{bi})| \leq \delta, \quad \forall i \in \mathbb{I} \tag{50}$$

$$s_b \in [-\delta, \delta] \tag{51}$$

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

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

$$z^{\max*} := \max_{x \in [x_{b-1}, x_b]} \left|\ell(x) - f(x)\right| \tag{52}$$

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

Algorithm 4.2 Forward Heuristic with Moving Breakpoints

1: // **INPUT:** Function f, scalar $\delta > 0$, initial discretization size $I^{\text{ini}} \in \mathbb{N}$ and parameter $\beta > 1$
2: // **OUTPUT:** Number of breakpoints, B, breakpoint system x_b and shift variable s_b
3: $x_0 := X_-$, $I := I^{\text{ini}}/\beta$, $B := 0$, and $b = 1$ // Initialize
4: // **Outer loop**
5: **repeat**
6: // **Inner loop**
7: **repeat**
8: $I \leftarrow \lceil \beta I \rceil$ // update discretization size
9: solve NLP (45)-(48) to obtain x_b^* // calculate next breakpoint and shift variable
10: solve one-dimensional NLP (49)-(51) to obtain s_b^*
11: solve unconstrained NLP (52) to obtain $z^{\max *}$ // check if obtained ℓ is δ-approximator
12: **until** $z^{\max *} \leq \delta$
13: $x_b := x_b^*$, $s_b := s_b^*$, $b \leftarrow b+1$, $B \leftarrow B+1$ // fix breakpoint, shift variable and update counter
14: **until** $x_b = X_b$

This procedure is summarized in Algorithm 4.2. Similar to the heuristic 4.1, the Algorithm 4.2 always terminates in finitely many steps (given exact arithmetics):

Corollary 4.2 *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.*

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 4.1 is expected to be larger than the one computed by 4.2. Particularly computationally expensive is solving problems (45)-(48) in 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 "backward" 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 backward 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 $\tilde{x}_b^* := X_+ + X_- - x_b^*$.

5 Computational Results

We have implemented the models and algorithms in GAMS (v. 23.6). The global optimization problems are solved using LindoGlobal (v. 23.6.5). 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 (1) and/or (2) is violated by at most 10^{-5}.

For our computational test bed, we consider ten different functions, summarized in Table 1. Figure 3 illustrates the ten functions (black line) together with δ-approxi-

Table 1: One-dimensional functions tested.

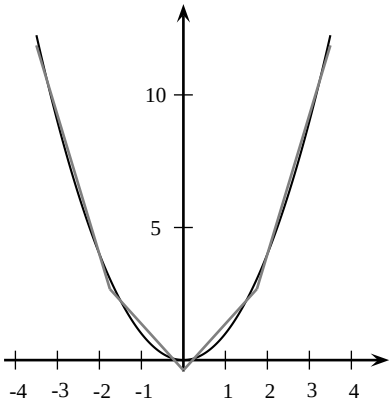
#	$f(x)$	X_-	X_+	Comment
1	x^2	-3.5	3.5	convex function, optimal distribution of breakpoints is uniform; axial symmetric at $x=0$
2	$\ln x$	1	32	concave function
3	$\sin x$	0	2π	point-symmetric at $x=\pi$
4	$\tanh(x)$	-5	5	strictly monotonically increasing; point symmetric at $x=0$
5	$\frac{\sin(x)}{x}$	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 $(-\infty,\infty)$, 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	sum of two Gaussians, with two slightly different maxima (their absolute function value difference is ≈ 0.030)
10	[27]	0	2π	three local minima (the absolute function value difference of the two smallest local minima is ≈ 0.031)

mators, 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 Algorithm 4.1 and Algorithm 4.2, where δ was chosen a priori. One can see, *e.g.*, in Fig. 3(h)-(j),

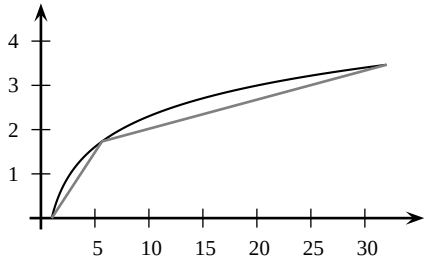
that our models do not compute approximators which are "closest" possible to the original function but which instead stay within a given δ-tube around the function.

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 4.1.1 and 4.1.2, are summarized in Table 2. Both heuristic methods are executed in a forward and backward 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 Algorithm 4.2 is significantly higher than that of Algorithm 4.1, because Algorithm 4.1 requires less NLP solves. Algorithm 4.2 consistently computes the same or fewer number of breakpoints for a given accuracy δ than Algorithm 4.1.A good trade-off between computational time and number of breakpoints computed are parameters $\alpha = 0.985$ and $D = 3$ for Algorithm 4.1 and $I^{\text{ini}} = 10$ and $\beta = 2.5$ for Algorithm 4.2.

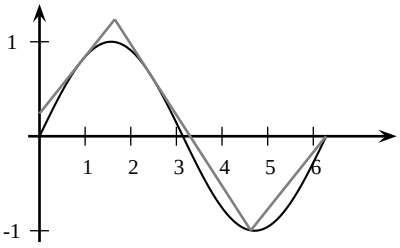
Table 3 summarizes the computational results obtained by FBSD. We use the lowest number of breakpoints calculated by any of the two heuristic methods for a given accuracy δ, *cf.* Table 2, to calculate the tightest possible approximator. We start with a grid size of $I = 1$ and solve FBSD. This yields 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.



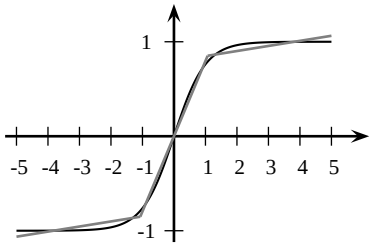
(a) Func 1: FBSD with $B = 5$ yields $\delta^* = 0.383$



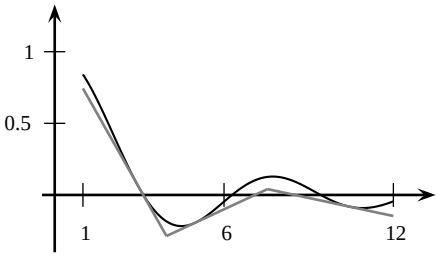
(b) Func. 2: FBSD with $B = 3$ yields $\delta^*_- = 0.361$



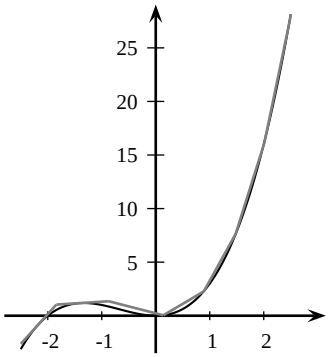
(c) Func. 3: FBSD with $B = 4$ yields $\delta^*_+ = 0.240$



(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. 4.1 w/ $\delta_+ = 0.5$ yields $B = 9$

Fig. 3: Continued.

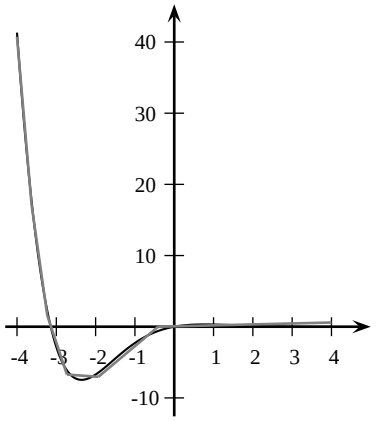
Table 2: Computational results for δ-approximators using heuristics.

#	δ	Algorithm 4.1				Algorithm 4.2			
		Forward		Backward		Forward		Backward	
		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
$\emptyset$	0.100		1.21		0.65		37.82		27.25
$\emptyset$	0.050		1.04		1.02		22.15		30.31
$\emptyset$	0.010		1.95		1.84		22.06		50.58
$\emptyset$	0.005		2.63		2.78		25.63		25.79

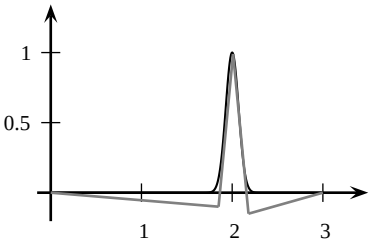
Table 3: Tightness obtained by FBSD for given B.

#	δ_{ini}	B	δ_{LB}	δ_{UB}	Max Grid	sec.
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

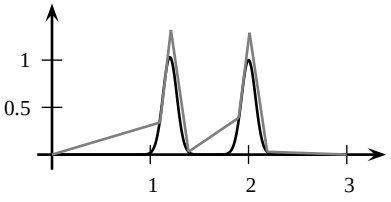
All other instances yield $\delta_{\text{LB}} = 0$ after 10h of CPU time.



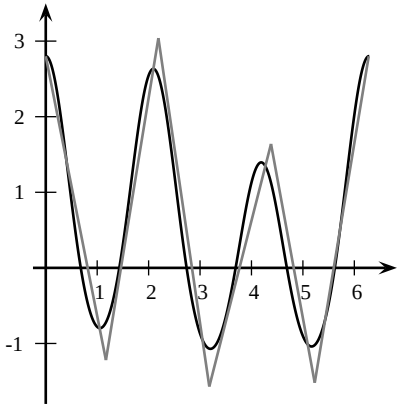
(g) Func. 7: Alg. 4.2 w/ $\delta = 0.6$ yields $B = 7$



(h) Func. 8: Alg. 4.1 w/ $\delta_- = 0.2$ yields $B = 5$



(i) Func. 9: Alg. 4.2 w/ $\delta_+ = 0.3$ yields $B = 8$



(j) Func. 10: Alg. 4.1 w/ $\delta = 0.5$ yields $B = 7$

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

— original function — approximator function

Table 4 summarizes the computational results for the model OBSD. We limit the size of the breakpoint set $\mathscr{B}$ by the lowest number of breakpoints computed in Table 2 for each discretization size δ. The continuum condition is initially discretized into

two points, *i.e.*, $I = 2$. By solving OBSD, we obtain a lower bound B_- on B^*. If B_- equals the initial number of breakpoints or the maximal deviation does not exceed δ (with an accuracy of 0.00125), then the algorithm stops with $B^* = B_-$. Otherwise, the grid size is updated by $I \leftarrow 1.5 \cdot I$ and the process starts over again. One observes in Table 4 that for most of the problems B^* cannot be computed. Furthermore, the required discretization size I is quite large.

OBSI performs much worse compared to OBSD. OBSI is able to obtain the optimal $B^* = 4$ only for function 5 with $\delta = 0.100$. The computational time is approximately 97 seconds, requiring a size of $I = 20$. For most of the other problem instances, not even a feasible point for the original model (using $I = 2 \cdot B$) can be computed within 1800 seconds of CPU time.

Table 5 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 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 when an equidistant distribution of the breakpoints is used together with a function interpolation. Table 6 summarizes the minimum number of equidistant breakpoints needed to ensure a given accuracy δ. We computes these breakpoint systems with the following brute-force algorithm. Starting with two

Table 4: Computational results for model OBSD.

#	δ	B^*	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)

breakpoints, compute the maximal deviation of the approximator to the function $f(x)$. This is accomplished by solving an NLP 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

Table 5: Benchmarks: Minimal number B^* of breakpoints needed for δ-approximators.

#	δ	B^*	B_-	B_+	**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		

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

Table 6: 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

than the reported times in Table 2; 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 6 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

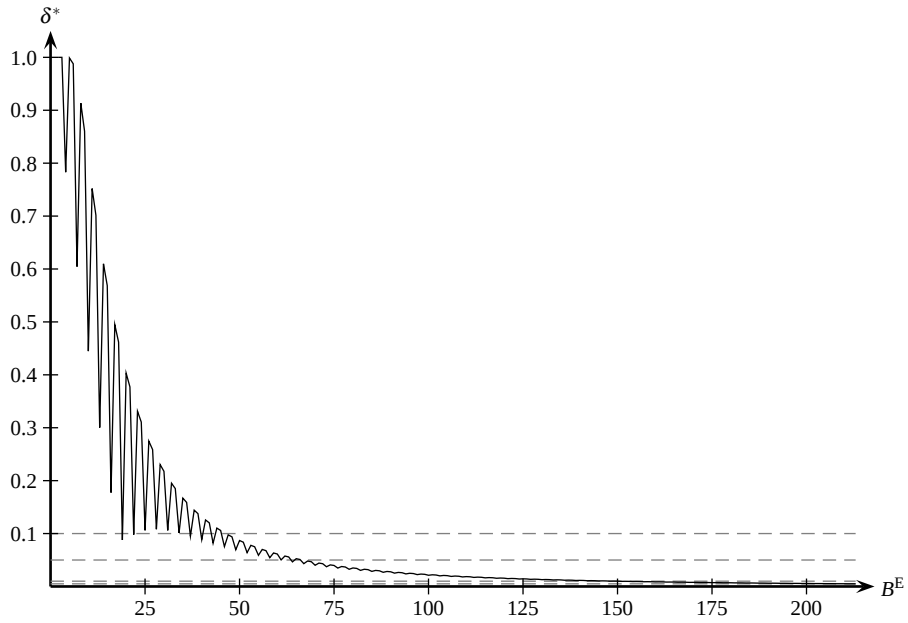


Fig. 4: Maximal deviation δ^* for different number of equidistant breakpoints B^E for function 8.

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.

6 Conclusions

For univariate functions, we have constructed various methods to compute optimal breakpoint systems to be used for piecewise linear approximation, under- and overestimation satisfying a specified accuracy δ. The exact models and heuristic methods require the solution of global optimization problems to ensure the δ-tolerance.

We have introduced the following models and methods:

1) Two MINLP models (OBSD & OBSI) which yield the minimal number and best distribution of breakpoints for a given δ-tolerance,
2) a MINLP model (FBSD) which computes the tightest approximation for a fixed number of breakpoints, and
3) two heuristic methods which compute the breakpoints subsequently by solving MINLPs 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-tolerance, 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.

Future research might develop explicit, piecewise-linear formulations of univariate functions that are only defined at regular or irregular grid 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 grid points, *i.e.*, to replace them by a coarser grid which, relative to the system of given grid 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 (both Colorado School of Mines) for their careful proof-reading.

References

1. Kallrath, J.: Combined strategic and operational planning - an MILP success story in chemical industry. OR Spectrum **24(3)**, 315–341 (2002)
2. Kallrath, J., Maindl, T.I.: Real optimization with SAP-APO. Springer (2006)
3. Zheng, Q.P., Rebennack, S., Iliadis, N.A., Pardalos, P.M.: Optimization models in the natural gas industry. In: S. Rebennack, P.M. Pardalos, M.V. Pereira, N.A. Iliadis (eds.) Handbook of power systems I, chap. 6, pp. 121–148. Springer (2010)
4. Frank, S., Steponavice, I., Rebennack, S.: Optimal power flow: a bibliographic survey I. Energy Systems **3**(3), 221–258 (2012)
5. Frank, S., Steponavice, I., Rebennack, S.: Optimal power flow: a bibliographic survey II. Energy Systems **3**(3), 259–289 (2012)
6. Tomlin, J.A.: Special ordered sets and an application to gas supply operating planning. Mathematical Programming **45**, 69–84 (1988)
7. Beale, E.L.M., Tomlin, J.A.: Special facilities in a general mathematical programming system for nonconvex problem using ordered sets of variables. In: J. Lawrence (ed.) Proceedings of the fifth international conference on operational research 1969, pp. 447–454. Tavistock Publishing, (1970)
8. Beale, E.L.M.: Two transportation oroblems. In: Proceedings of the third international conference on operational research 1963, pp. 780–788. Dunod, Paris and English Universities Press, (1963)
9. Beale, E.M.L., Forrest, J.J.H.: Global optimization using special ordered sets. Mathematical Programming **10**, 52–69 (1976)
10. de Farias Jr., I.R., Johnson, E.L., Nemhauser, G.L.: A generalized assignment problem with special ordered sets: a polyhedral approach. Mathematical Programming Ser. A **89**, 187–203 (2000)
11. de Farias Jr., I.R., Zhao, M., Zhao, H.: A special ordered set approach for optimizing a discontinuous separable piecewise linear function. Operations Research Letters **36**, 234–238 (2008)
12. Leyffer, S., Sartenaer, A., Wanufelle, E.: Branch-and-refine for mixed-integer nonconvex global optimization (2008)
13. Vielma, J.P., Ahmed, S., Nemhauser, G.: Mixed-integer models for nonseparable piecewise-linear optimization: unifying framework and extensions. Operations Research **53**, 303–315 (2009)

14. Vielma, J.P., Nemhauser, G.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Mathematical Programming **128**, 49–72 (2011)
15. Rosen, J.B., Pardalos, P.M.: Global minimization of large-scale constrained concave quadratic problems by separable programming. Mathematical Programming **34**, 163–174 (1986)
16. Pardalos, P.M., Rosen, J.B.: Constrained global optimization: algorithms and applications. Lecture Notes in Computer Science. Springer (1987)
17. Geißler, B., Martin, A., Morsi, A., Schewe, L.: Using piecewise linear functions for solving MINLPs. In: J. Lee, S. Leyffer (eds.) Mixed Integer Nonlinear Programming, The IMA Volumes in Mathematics and its Applications, vol. 154, pp. 287–314. Springer (2012)
18. Geißler, B.: Towards globally optimal solutions for MINLPs by discretization techniques with applications in gas network optimization. Dissertation, Friedrich-Alexander-Universität Erlangen-Nürnberg, (2011)
19. Hettich, R., Kortanek, K.O.: Semi-infinite programming. SIAM Review **35**, 380–429 (1993)
20. Lopez, M., Still, G.: Semi-infinite programming. European Journal of Operational Research **180**, 491–518 (2007)
21. Tsoukalas, A., Rustem, B.: A feasible point adaptation of the blankenship and falk algorithm for semi-infinite programming. Optimization Letters **5**(4), 705–716 (2011).
22. Blankenship, J.W., Falk, J.E.: Infinitely constrained optimization problems. Journal of Optimization Theory and Applications **19**, 268–281 (1976)
23. Rebennack, S., Kallrath, J.: Continuous piecewise linear delta-approximations for MINLP problems. II. Bivariate and multivariate functions. Tech. rep., Colorado School of Mines (2012).
24. Duistermaat, J., Kol, J.: Multidimensional real analysis I: differentiation. Cambridge Studies in Advanced Mathematics (2004)
25. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to global optimization, 2nd edn. Kluwer (2000)
26. Kallrath, J., Rebennack, S.: Computing area-tight piecewise linear overestimators, underestimators and tubes for univariate functions. In: S. Butenko, C. Floudas, T. Rassias (eds.) Optimization in Science and Engineering. Springer (2014)
27. Maranas, C., Floudas, C.A.: Global minimum potential energy conformations of small molecules. Journal of Global Optimization **4**, 135–170 (1994)