

LinA: A faster approach to piecewise linear approximations using corridors and its application to mixed-integer optimization

Julien Codsi, Sandra Ulrich Ngueveu, Bernard Gendron

▶ **To cite this version:**

Julien Codsi, Sandra Ulrich Ngueveu, Bernard Gendron. LinA: A faster approach to piecewise linear approximations using corridors and its application to mixed-integer optimization. 2021. hal-03336003

HAL Id: hal-03336003
https://hal.science/hal-03336003

Preprint submitted on 6 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

LinA: A faster approach to piecewise linear approximations using corridors and its application to mixed-integer optimization

Julien Codsi[1] Sandra Ulrich Ngueveu[2] Bernard Gendron[1]

[1] CIRRELT and Département d'informatique et de recherche opérationnelle, Université de Montréal, Canada
[2]Université de Toulouse, CNRS, INP, LAAS, Toulouse, France

Abstract

In this paper, we address the problem of approximating and over/under-estimating univariate functions with piecewise linear (PWL) functions with the minimum number of linear segments given a bound on the pointwise approximation error allowed. Through a new geometric approach and building on the work of Ngueveu [Ngu19], we develop new algorithms that can solve the problem in quasi-logarithmic time on a very broad class of error types. Such algorithms find many applications, mostly related to solving certain classes of (mixed-integer) nonlinear and nonconvex programming (MINLP) problems by mixed-integer linear programming (MILP) techniques. An efficient implementation of our algorithms is available as a Julia package. Benchmarks are also provided to showcase how our method outperforms the state-of-the-art for this problem. Finally, we show how our algorithms can be used to efficiently solve certain classes of MINLP problems by a case study on multicommodity network design problems with congestion.

Keywords: piecewise linear functions, approximation, overestimation, underestimation, guaranteed tolerance, piecewise linear regression with bounded error, MINLP, MILP, Julia package

1 Introduction

The simplicity and ease of use of linear functions make them very attractive to many researchers and practitioners. However, when dealing with non linear behaviors, it is often preferable to use models that better encompass the reality. As a trade-off between simplicity and precision, piecewise linear functions (PWL) are used in a variety of fields such as computer graphics, data science and optimization. Throughout these fields, different criterion are used to judge the quality of a PWL. Although our results are very general, this paper is motivated by the applications of PWL in Mixed integer nonlinear programming (MINLP) which guided the exact choices for the specifications of the problem tackled.

MINLP models are generally hard to solve. In addition to the presence of integer variables, one often has to handle nonlinear functions that are not necessarily convex. A widely used approach is to approximate the nonlinear functions with piecewise linear ones in order to derive mixed integer linear programming (MILP) models, thus benefiting from the large scale, continuous and sustained effort of the optimization community on MILP for the past thirty years and beyond. [GMMS12] were among the first to show that, in certain cases, MINLP models can be solved by applying purely techniques from MILP after approximating nonlinearities by piecewise linear functions. However, they did not focus on minimizing the number of linear segments. Piecewise linear approximation for solving MINLP is often performed in a preprocessing step using ad hoc methods. In general, the approximation error

is not known a priori and the number of linear pieces (translating into binary variables in the MILP model) is not minimized. In this paper, we address these issues and propose a geometric approach to compute a piecewise linear (PWL) function that minimizes the number of linear segments needed to approximate, over-estimate or under-estimate a nonlinear continuous univariate function with a bounded pointwise approximation error.

The specificities of our problem, which separate it from the large majority of studies in the field of univariate piecewise linear approximation, are the following:

- Most contributions on piecewise linear approximations of nonlinear univariate functions focus on the minimization of an approximation error given a predefined number of linear segments (see for example [AMM13], [CN15]). In contrast, in this paper, we are interested in minimizing the number of linear segments given a bounded approximation error.

- Because the ultimate goal is to solve MINLP problems using techniques from MILP, we are interested in pointwise errors, i.e errors that can be expressed as a function of the maximal difference between the nonlinear function and its approximation. This excludes most of the metrics classically used for piecewise linear regression in data mining or statistics, such as the sum of square deviations or the total sum of absolute deviations ([EF76], [Wat98], [YLTP16]).

- Most piecewise linear regression studies, also known as segmented linear regression or curve fitting or piecewise linear function fitting, consider as an input a discrete set of points, instead of the continuous function we consider. Even in cases where the continuous function was known, the function was sampled and the approximation was performed on the set of sample points. The algorithms proposed in the regression field did not ensure the respect of the predefined approximation error on the entire continuous domain and are therefore not directly applicable to our problem, ([DT73], [LABOO01], [TV12], [CN15]).

- Finally, we are interested in exact methods providing optimal solutions or, at least, guarantees on the quality of the solutions produced, which excludes most heuristics from the literature.

To the best of our knowledge, only five papers address the specific problem we are interested in, computing piecewise linear functions which minimize the number of linear segments, given a bounded approximation error expressed in function of the maximal difference with the nonlinear univariate function.

[RP86] first proposed to build *continuous* PWL interpolators that verify a specified "tolerance", but only for concave quadratic programs and using equidistant breakpoints.

[RK15] introduced two nonconvex optimization models and two heuristics for the computation of *continuous* PWL approximations that minimize the number of linear segments. The authors distribute breakpoints freely and allow shifts from the function at breakpoints, leading to up to an order of magnitude less breakpoints compared to the classical equidistant interpolation approach.

[RK19] proposed the first convex model for computing a *continuous* PWL approximation of a finite set of points that minimizes the number of pieces. The model proposed is then used to develop an exact algorithm for piecewise linear approximation of continuous univariate functions. The convex formulation is based on the idea that computing the exact locations of the breakpoints is not needed during function fitting. Rather, it is sufficient to ensure that adjacent linear segments of the constructed function intersect within a certain range. The exact algorithm consists in solving a series of finite point fitting problems via the proposed convex models. By evaluating the computed *continuous* PWL function (also known in the literature as linear or first-order splines) the authors can identify points where the maximum difference with the original function is larger than desired. These points are added to the discretization and new finite point fitting problems are solved. Computational results show that the resulting algorithm outperforms the ones from [RK15].

[KM20] also proposed mixed-integer programs for computing a *continuous* PWL approximation of a finite set of points, that is then used to develop an exact algorithm for piecewise linear approximation

of continuous univariate functions. However, the reported computational results show that this is more efficient than approaches that utilize nonlinear constraints, but the proposed method may not be competitive in comparison to [RK15] and [RK19].

[Ngu19] propose approximating a general univariate continuous function with a *non necessarily continuous* PWL function, adding an additional degree of freedom to obtain a breakpoint system of equal or less linear segments. The author presents models and algorithms to compute the PWL approximator/over-estimator/under-estimator with predefined absolute or relative error tolerance and with an additive worst case guarantee on the number of linear segments needed. Evaluations on the instances from [RK15] and [RK19] show a drastic reduction of the computing times in comparison to the state-of-the-art, and sometimes a reduction of the number of linear segments. For solving MINLPs with an objective-function that is separable in a sum of positive univariate nonlinear terms, [Ngu19] proposes a method based on upper and lower bounding the nonlinear terms using *non necessarily continuous* PWL functions with a predefined relative tolerance and the solution of a pair of mixed integer linear programs. Such an approach yields a performance guarantee when the nonlinearity is restricted to the objective function. To illustrate the efficiency of the method in comparison to state-of-the-art methods and general-purpose MINLP solvers, computational evaluation was performed on an energy optimization problem for hybrid electric vehicles.

The main contributions of this paper are the following: (I) the notion of a "corridor" that generalizes the classes of errors considered in piecewise linear approximation, (II) through a greedy algorithm, a reduction of the original problem to the maximal linear piece problem, (III) an algorithm to solve the problem in the general case, (IV) a logarithmic time algorithm for the special case of convex corridors through a strong characterisation of the optimal solution, (V) clever speed-ups for a large class of corridors, (VI) an efficient and flexible Julia package, (VII) benchmarks on nonlinear functions from the literature that illustrate the major performance speed-ups of our method, and (VIII) benchmark on the resolution of MINLP through our linearization + MILP solver approach

Most notably, the practical performance is drastically improved thanks to a new characterization of optimal solutions with intersections and tangents that are easy to compute. The piecewise linear functions computing times reported in [Ngu19] varied from dozen to hundreds seconds. Although it was proven sufficient to outperform general purpose MINLP solvers on problems containing a single nonlinear function to approximate, it was not fast enough to tackle cases where there might be multiple nonlinear functions to approximate in a single instance. For example, this is the case of the network design problem with congestion studied in this paper, where there is a different nonlinear congestion function for each node of a graph, and graphs can contain up to a hundred nodes.

2 Definitions

Here we define terms that will be used throughout this paper.

Definition 1 (PWL function)**.** *A function $g : \mathbb{D} = [x_-, x_+] \to \mathbb{R}$ is a piecewise linear (PWL) function with n linear segments if it can be defined by equations (1) where $x_1 = x_-$ and $x_{n+1} = x_+$. The* domain length *of g is equal to $x_+ - x_-$.*

$$g(x) = \begin{cases} g_i(x) = a_i x + b_i, & \forall i \in \{1..n-1\}, \forall x \in [x_i, x_{i+1}[\\ g_n(x) = a_n x + b_n & \text{if } x \in [x, x_{n+1}] \end{cases} \tag{1}$$

Remark. *It is to important to note that the continuity property $g_{i-1}(x_i) = g_i(x_i)\ \forall\ i \in \{2, \ldots, n\}$ is neither imposed nor forbidden, which implies that a PWL is non necessarily continuous.*

Definition 2 (corridor)**.** *Let $h, l : \mathbb{D} = [x_-, x_+] \to \mathbb{R}$ be two continuous functions verifying $h(x) > l(x), \forall x \in \mathbb{D}$. The surface area $\mathcal{C} \subset \mathbb{R}^2$ is called the* corridor *between h and l iff $\mathcal{C} = \{(x, y) | x \in \mathbb{D}, l(x) \leqslant y \leqslant h(x)\}$.*

The *domain length* of $\mathcal{C}$ is equal to $x_+ - x_-$.

Definition 3 (sub-corridor)**.** *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two corridors defined by functions $h_1, l_1 : \mathbb{D}_1 \to \mathbb{R}$ and $h_2, l_2 : \mathbb{D}_2 \to \mathbb{R}$, respectively. We call $\mathcal{C}_2$ a sub-corridor of $\mathcal{C}_1$ iff $\mathbb{D}_2 \subseteq \mathbb{D}_1$, $h_1(x) = h_2(x)$, and $l_1(x) = l_2(x), \forall x \in \mathbb{D}_2$.*

Definition 4 (truncated-corridor)**.** *Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two corridors defined by some function l and h respectively on the interval $[a, b]$ and $[c, d]$. We call $\mathcal{C}_2$ a truncated-corridor of $\mathcal{C}_1$ iff $\mathcal{C}_2$ is a sub-corridor of $\mathcal{C}_1$ and $a = c$.*

Definition 5 (function within a corridor)**.** *A function $g : \mathbb{D}_g \to \mathbb{R}$ is* within *a corridor $\mathcal{C}$ iff $(x, g(x)) \in \mathcal{C}\ \forall x \in \mathbb{D}_g$.*

Definition 6 (induced sub-corridor)**.** *Let $g : \mathbb{D}_g \to \mathbb{R}$ be a function within a corridor $\mathcal{C}_1$. A corridor $\mathcal{C}_2$ is called an induced sub-corridor iff $\mathcal{C}_2 = C_1 \cap (\mathbb{D}_g \times \mathbb{R})$, i.e C_2 is the same as $\mathcal{C}_1$ but limited to the region where g is defined. We denote $\mathcal{C}_2 = \mathcal{C}_1(g)$*

Definition 7 (fitting)**.** *A function g* fits *a corridor $\mathcal{C}$ iff g is within $\mathcal{C}$ and $\mathcal{C}(g) = \mathcal{C}$, i.e the projected length of g is the same as the length of C.*

Definition 8. *(maximal linear segment) A maximal linear segment in a corridor $\mathcal{C}$ is a linear segment within $\mathcal{C}$ that induces a truncated-corridor of maximal domain length.*

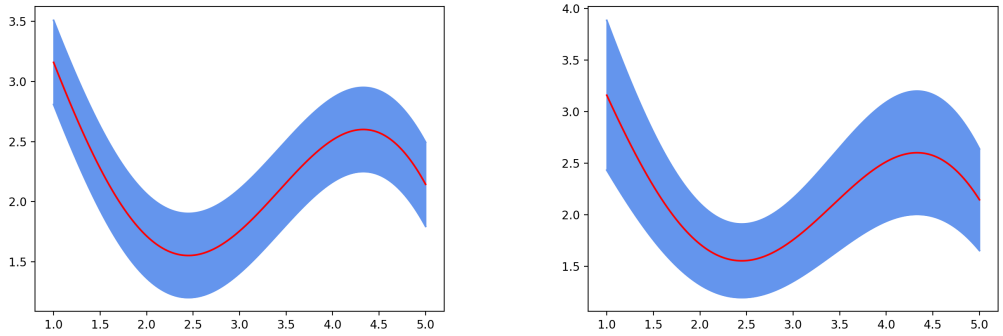
3 Fitting a piecewise linear function through a corridor

We define the notion of corridor and use this notion to present a new geometric approach for the problem. A corridor is defined as the region between two functions that do not intersect on a compact interval. We are interested in finding a PWL function g that fits a given corridor $\mathcal{C}$ with a minimal number of linear segments. This problem is named the *corridor fitting problem*. To compute an approximation, an overestimation or an underestimation of a function with a predefined pointwise error tolerance, we define a corridor surrounding the function so that the problem is equivalent to finding a piecewise linear function fitting the corridor. The corridor fitting problem therefore generalizes the problem of finding a piecewise linear function that overestimates, underestimates or approximates an univariate function, such that the number of linear segments is minimized given a bound on a pointwise error metric. As shown in table 1, this includes corridors induced by absolute and relative pointwise errors, thus generalizing the work of [Ngu19].

	absolute tolerance δ	relative tolerance ε
approximation	$h(x) = f(x) + \delta$	$h(x) = f(x) + \varepsilon\lvert f(x)\rvert$
	$l(x) = f(x) - \delta$	$l(x) = f(x) - \varepsilon\lvert f(x)\rvert$
overestimator	$h(x) = f(x) + \delta$	$h(x) = f(x) + \varepsilon\lvert f(x)\rvert$
	$l(x) = f(x)$	$l(x) = f(x)$
underestimator	$h(x) = f(x)$	$h(x) = f(x)$
	$l(x) = f(x) - \delta$	$l(x) = f(x) - \varepsilon\lvert f(x)\rvert$

Table 1: Equivalence of the piecewise linear approximation, overestimation and underestimation of a function f with the corridor fitting problem for a corridor $\mathcal{C}$ defined by h and l

To visualise this, figure 1 shows the corridors associated to the function $f(x) = \sqrt{x}\sin(-x) - x + 5$ defined on interval $[1, 5]$ for an absolute error of 0.35 and for a relative error of 23%.



(a) absolute error tolerance b) relative error tolerance

Figure 1: Corridors induced by an absolute and a relative error

Such error classes finds many applications, mostly related to solving (mixed-integer) nonlinear and nonconvex programming problems by (mixed-integer) linear programming techniques.

Remark. *The absolute error case can be interpreted as a bound on the Chebyshev norm* $\|\cdot\|_\infty$ *between the PWL and the original function.*

3.1 Overview of the algorithm

Our solution is based on a greedy algorithm that creates the PWL function in an iterative manner by choosing the linear segment of maximal possible domain length at every step. This reduce the corridor fitting problem to a simpler problem, namely the *maximal linear piece problem*. This procedure is detailed in algorithm 1.

Algorithm 1 Computation of an optimal PWL function fitting a corridor $\mathcal{C}$

Input: Corridor $\mathcal{C}$
Output: PWL function g defined by the set of linear segments $\mathcal{P}$

```
while C ≠ ∅ do
    p* ← Compute maximum linear piece of C
    P ← P ∪ p*
    C ← C\C(p*)
end while
return P
```

The correctness of the algorithm is proved in subsection 3.2. We postponed the resolution of the maximal linear piece problem to the section 4. The section 5 mostly tackles clever ways to accelerate the algorithm 1 for corridors with slightly more structure or when an almost-optimal approximation of the corridor fitting problem is sufficient. Finally, section 6 is mainly constituted of computational results to show how our approach outperforms the state of the art.

3.2 Optimality of a piecewise linearization with maximal linear segments

Proposition 1 and Corollary 1 extend the Theorem 3.3 from [Ngu19] to the concept of corridors.

Proposition 1. *Given a corridor* $\mathcal{C}$*, there exists an optimal solution of the corridor fitting problem on* $\mathcal{C}$ *where the first segment is a maximal linear segment in* $\mathcal{C}$*.*

Proof. Let $\mathcal{C}$ be a corridor. Let g be any optimal PWL function that fits $\mathcal{C} = [x_-, x_+]$ and let $\{x_1, x_2, \dots x_n\}$ be the break points of the linear segment of g. Note that $x_1 = x_-$ and $x_n = x_+$. Let $p : [x_-, x^*] \to \mathbb{R}$ be a maximal linear piece in $\mathcal{C}$. Let g^* be the PWL function defined by 2

$$g^*(x) = \begin{cases} p(x) & \forall x \in [x_-, x^*[\\ g(x) & \forall x \in [x^*, x_+] \end{cases} \tag{2}$$

Note that, by definition, $x^* \geqslant x_2$. Therefore g^* has at most as much linear pieces as g which concludes the proof.

□

Corollary 1. *Given a corridor $\mathcal{C}$, there exists an optimal solution of the corridor fitting problem such that each linear segment is a maximal linear segment.*

Proof. Proposition 1 can be applied iteratively on an optimal PWL function to ensure that each linear segment induces a truncated-corridor of maximal domain length. This works because this problem has optimal substructure. In other words, if the optimal function is splitted at the end of a segment, it will induce optimal solutions for the two sub-corridors. □

Corollary 1 proves that the corridor fitting problem can be solved with a greedy algorithm that computes a succession of maximal linear segment problems. This is a sharp contrast with what can be done with continous PWL as in this case, the greedy algorithm is not guarantied to give an optimal solution as shown in figure 2. Here, the blue curve is an optimal continuous PWL with 2 segments and the red curve is the continous PWL with 3 segments obtained with a greedy algorithm.

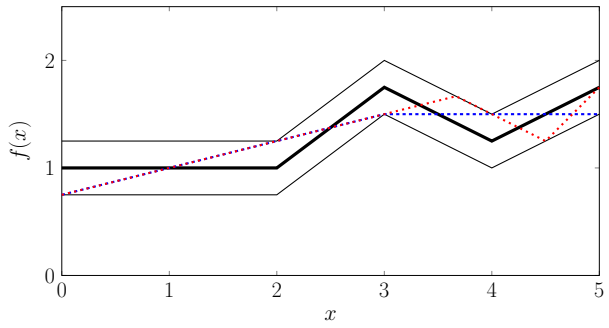


Figure 2: Example of the non optimality of the greedy algorithm in the continuous case

4 Maximal linear segment problem

In this section, we first tackle the maximal linear segment problem for the case of convex corridor. Later, we tackle the general case. Even though the former is a special case of the later, a dedicated algorithm was conceived for convex corridors because a strong characterization of the solution was found which results in a major speedup over the general case.

4.1 Case of a convex or concave corridor

We consider a convex corridor $\breve{\mathcal{C}}$ defined by functions $h, l : \mathbb{D} = [x_-, x_+] \to \mathbb{R}$ both continuously differentiable , i.e. a corridor for which the two bounding functions h, l are C^1 and convex. Note that all differentiable convex functions are C^1, so the condition on the continuity of the derivative was only added for clarity. Notice that solving the maximal linear segment problem on a concave corridor $\hat{\mathcal{C}}$ defined by functions $\hat{l}, \hat{h}$ reduces to the convex case as described above. Indeed, the corridor defined

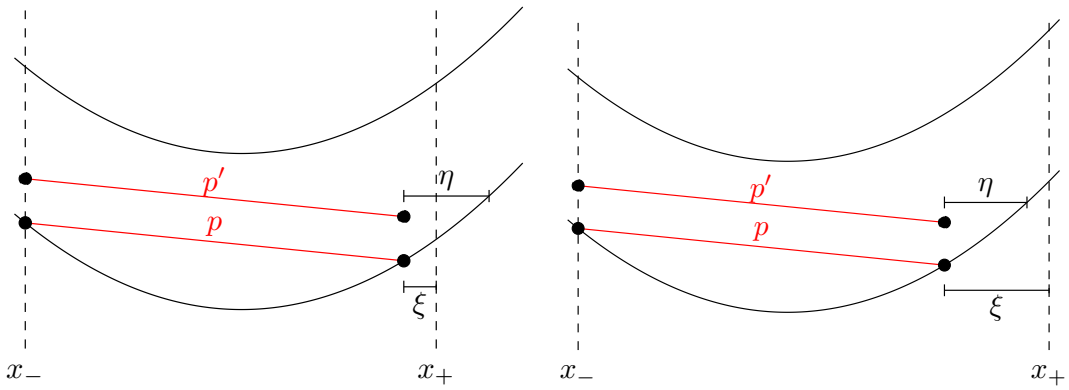


Figure 3: Illustration of Lemma 2: case where $\min\{\xi, \eta\} = \xi$ and where $\min\{\xi, \eta\} = \eta$

by the convex functions $-\hat{l}$ and $-\hat{h}$ is only a sign off and the solution to the original problem can easily be retrieved from a solution of this new instance by simply again multiplying by -1. Lemmas 1 and 2 characterize a maximal linear segment in $\breve{\mathcal{C}}$.

Lemma 1. *For any linear segment within $\breve{\mathcal{C}}$ there exists a linear segment of equal domain length within $\breve{\mathcal{C}}$ that intersects the curve $\mathcal{L} = \{(x, l(x))\}$ at its two endpoints.*

Proof. Given any linear segment p within corridor $\breve{\mathcal{C}}$ defined by endpoints (x_1, y_1) and (x_2, y_2), we can define a linear piece of equal projected length p^* with its endpoints $(x_1, l(x_1))$ and $(x_2, l(x_2))$ on the curve $\mathcal{L}$. Because h, l are convex, this linear piece is within $\breve{\mathcal{C}}$. Precisely, p^* must lies above $\mathcal{L}$ as it is a line segment between two points on the graph of a convex function. Moreover, p^* must lies under $\mathcal{H} = \{(x, h(x))\}$ as $p^*(x) \leqslant p(x) \leqslant h(x)$ for every x. Therefore, p^* is within $\breve{\mathcal{C}}$. □

Lemma 2. *A maximal linear segment in $\breve{\mathcal{C}}$ either fits $\breve{\mathcal{C}}$ or is tangent to the curve $\mathcal{H} = \{(x, h(x))\}$.*

Proof. This assertion is proven by contradiction as follows. Let us assume that a segment p, defined by the points (x_1, y_1) and (x_2, y_2), is a maximal linear segment in $\breve{\mathcal{C}}$ that does not fit $\breve{\mathcal{C}}$ and is not tangent to $\mathcal{H}$. Since p is not tangent to $\mathcal{H}$ and clearly doesn't cross $\mathcal{H}$ there is no x such that $p(x) = h(x)$. Therefore, there exists $c \in \mathbb{R}$ such that $p' = p + c$ doesn't intersect $\mathcal{H}$ i.e it's possible to do a vertical translation of p by c while still being within the corridor. For example, c could be taken as half the minimal vertical distance between p and $\mathcal{H}$. For the sake of simplicity, let us denote $y_1 + c$ by y_1', $y_2 + c$ by y_2' and the slope of p' by Δ. By construction, $(x_2, y_2') \notin \mathcal{L}$. Also, since p and therefore p' does not fit $\breve{\mathcal{C}}$, there exists $\xi > 0$ such that $x_2 + \xi \leqslant x_+$. Finally, since none of the endpoints of p' lie on $\mathcal{L}$ or $\mathcal{H}$, there exist $\eta > 0$ such that $\forall z \leqslant \eta, \quad (x_2 + z, y_2' + \Delta z) \notin \mathcal{L} \cup \mathcal{H}$ (i.e it is possible to extend p' without intersecting $\mathcal{L}$ or $\mathcal{H}$). We can therefore construct the linear segment within $\breve{\mathcal{C}}$ defined by (x_1, y_1') and $(x_2 + \min\{\xi, \eta\}, y_2' + \Delta \min\{\xi, \eta\})$, which has a domain length of $x_2 + \min\{\xi, \eta\} - x_1 > x_2 - x_1$, contradicting the optimality of p. □

Figure 3 illustrates the process in the proof of lemma 2.

Lemma 1 implies we can always choose $(x_-, l(x_-))$ as the starting endpoint of a maximal linear segment. Furthermore, Lemma 2 implies that an optimal segment will always be tangent to $\mathcal{H}$ if it is not possible to fit the corridor with a single linear segment. By combining these two lemmas, we obtain a unique (in the non-degenerate case) characterization of the optimal linear segment (as illustrated in figure 4) which gives rise to an efficient algorithm. To do so, let us define $t(q, x)$ as the equation of the tangent to $\mathcal{H}$ at q evaluated at point x. Formally:

$$t(q, x) = h'(q)(x - q) + h(q)$$

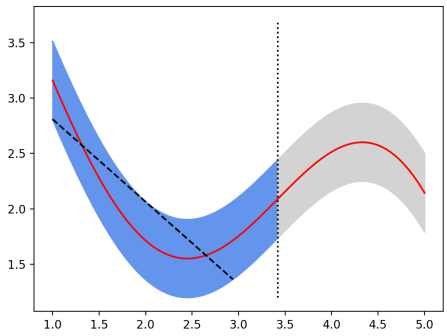


Figure 4: Maximal linear piece on the convex sub-corridor derived from Figure 1(a)

We are looking for a point q^* such that $t(q^*, x_-) = l(x_-)$ or equivalently, $t(q^*, x_-) - l(x_-) = 0$. Let

$$\Gamma(q) = t(q, x_-) - l(x_-)$$

The geometric interpretation of $\Gamma(q)$ is illustrated in Figure 5. Our interest in $\Gamma(q)$ is motivated by the fact that solving the maximal linear piece problem boils down to finding a zero of the decreasing function $\Gamma(q)$ which can be solved by dichotomy. We will prove this assertion in the following lemma.

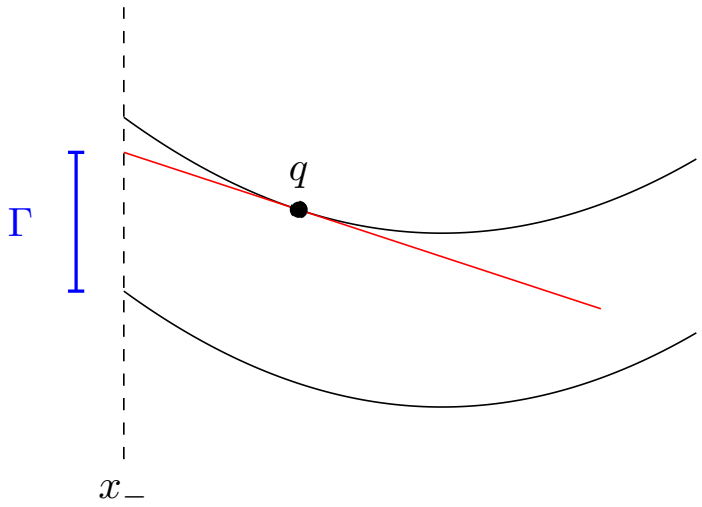


Figure 5: Function $\Gamma(q)$

Lemma 3. Γ *is a decreasing function (strictly decreasing if* h *is strictly convex)*

Proof. Without loss of generality, we'll assume that $x_- = 0$. Since $\Gamma(q)$ is only a constant off $t(q, 0)$, it is sufficient to prove that $t(q, 0)$ is decreasing. Let $a, b \in \mathbb{R}$ such that $0 > a > b$. We will show that

$t(a,0) \geqslant t(b,0)$.

$$\begin{aligned} & t(a,0) = h'(a)(-a) + h(a) \\ \implies\ & t(a,0) + h'(a)b = h'(a)(b-a) + h(a) \\ & = t(a,b) \end{aligned}$$

Since the tangent of a convex function is always below the curve, we have that $t(a,b) \leqslant h(b)$

$$\begin{aligned} & \leqslant h(b) \\ \implies\ & t(a,0) \leqslant h'(a)(-b) + h(b) \\ \xRightarrow{Concavity}\ & \leqslant h'(b)(-b) + h(b) \\ & = t(b,0) \end{aligned}$$

□

Once the optimal tangent point is found by finding the zero of Γ, we only need to find the second endpoint. This is done by computing where the line supporting the linear segment crosses $\mathcal{L}$ again using dichotomy since there will be only one other crosspoint and since the difference between a linear function and a convex function is still convex.

The full procedure is summarized in Algorithm 2. As the only costly step of this algorithm is the dichotomic search, this give rise to asymptotic logarithmic run time. The logarithmic time complexity of this approach is to be stressed as it is the main reason behind the efficiency of the implementation.

Algorithm 2 Solve the maximal linear piece problem on a convex corridor

Input: Convex corridor $\mathcal{C}_v$ defined by convex functions $h, l : \mathbb{D} = [x_-, x_+] \to \mathbb{R}$
Output: Optimal linear piece defined by breakpoints (x_1, y_1) and (x_2, y_2)

$(x_1, y_1) \leftarrow (x_-, l(x_-))$
if $\Gamma(x_+) \geqslant 0$ **then**
  $(x_2, y_2) \leftarrow (x_+, h(x_+))$
else
  $q^* \leftarrow$ Solve $\Gamma(q) = 0$ using dichotomy
  $p \leftarrow$ linear segment defined by (x_1, y_1) and q^*
  if $p(x_+) > l(x_+)$ **then**
    $(x_2, y_2) \leftarrow (x_+, h(x_+))$
  else
    $x^* \leftarrow$ Solve $p(x) - l(x) = 0$ for $x > x_-$ using dichotomy
    $(x_2, y_2) \leftarrow (x^*, l(x^*))$
  end if
end if
return $(x_1, y_1), (x_2, y_2)$

As previously mentioned, Algorithm 2 applies to both the convex and the concave cases if we pre-process and post-process concave instances. Adding these processing steps gives us Algorithm 2'.

4.2 General case based on converting corridors into data ranges

An iterative procedure solves the maximal linear segment problem in the general case. The procedure builds a discretized variant of the problem, solves it using a state-of-the-art algorithm on the discretized version of the problem and then retrieves a solution candidate for our continuous problem. Finally, it updates the discretization and repeats the last two steps until the solution obtained is feasible and optimal for the initial problem.

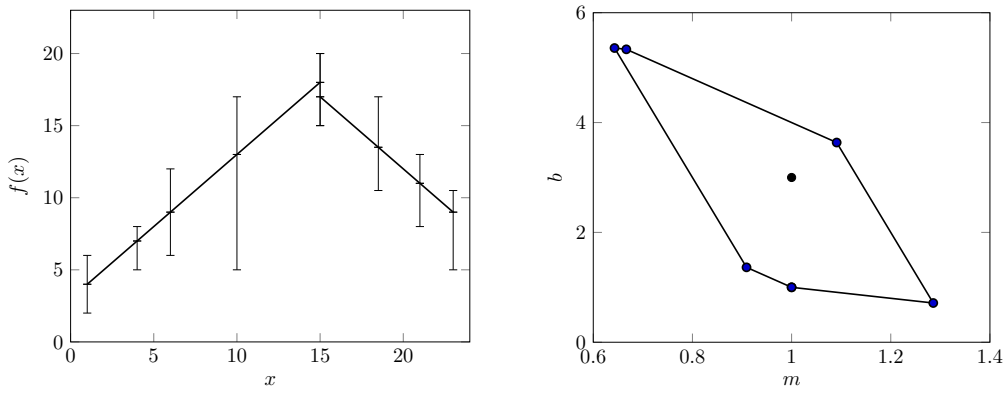


Figure 6: Example from [O'R81] with $n = 8$ data ranges and the polyhedron in the m-b space corresponding to the first five data ranges

To obtain the discrete variant of the problem the interval is discretized into a finite set of points $\mathcal{X}$. Any valid linear segment within a corridor $\mathcal{C}$ has to at least verify equations (3) on a consecutive subset of $\mathcal{X}$. The discrete variant of the maximal linear segment problem consists in finding a linear segment that verifies equation (3) for a maximal number of consecutive point $x_i \in \mathcal{X}$ including the first one x_1.

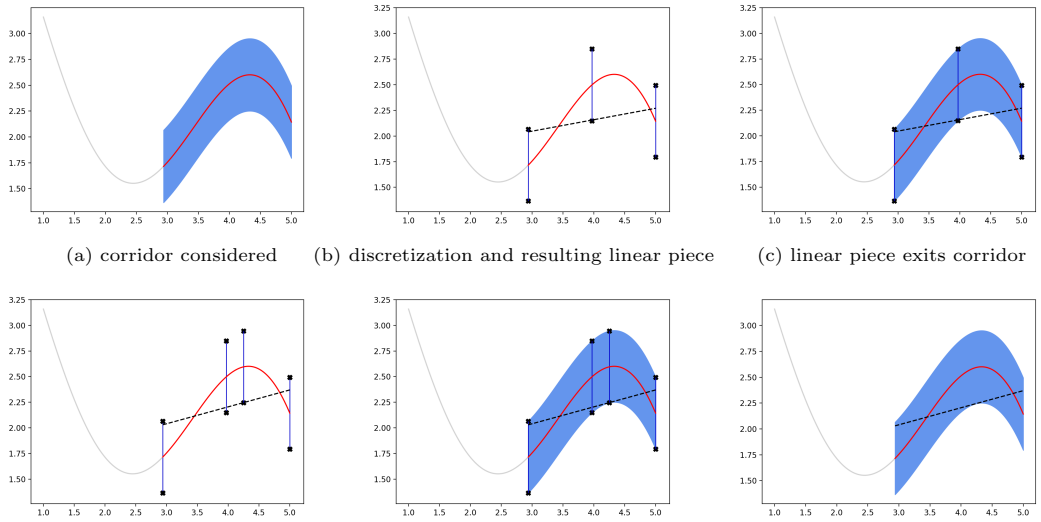
$$l(x_i) \leqslant mx_i + b \leqslant h(x_i) \tag{3}$$

This problem is known in the literature as data fitting or fitting a straight line between data ranges. For such discrete inputs, there exists publications on piecewise linear approximation with a minimum number of segments given a predefined bound on the absolute error in the fields of data reduction, pattern recognition or classification, and ECG waveform preprocessing ([Tom74b], [Tom74a], [GP83]).

A state-of-the-art algorithm for this problem is the one of O'Rourke [O'R81]. It works by considering each equation (3) as a pair of constraints (4.1), (4.2) in the m-b parameter space as illustrated in Figure 6. These constraints define a polyhedron denoted P_k for the k first consecutive points of $\mathcal{X}$. If the resulting polyhedron P_k is empty, then no single line can verify equation (3) for the k first points of $\mathcal{X}$. Otherwise, any point inside or on the boundary of P_k represents a valid line. The resulting algorithm computes P_n with an $O(n)$ complexity (for n input points). The algorithm stops if either $k = n$ if all points have been covered or $P_{k+1} = \varnothing$ if k is the maximal number of consecutive points of $\mathcal{X}$ that can be covered and therefore $(m, b) \in P_{k-1}$.

$$\begin{cases} b \leqslant (-x_i)m + h(x_i) & (4.1) \\ b \geqslant (-x_i)m + l(x_i) & (4.2) \end{cases} \tag{4}$$

Let (m, b) and k be the straight line parameters and the number of intersected data ranges for the optimal solution of the discretized maximal linear segment problem. Verifying if the solution is feasible for the original non-discretized maximal linear segment problem consists in verifying if the linear segment f defined by $(x_1, mx_1 + b)$ and $(x_k, mx_k + b)$ fits within the truncated-corridor $\mathcal{C}_k$ with domain $\mathbb{D}_k = [x_-, x_k]$. If the solution is not feasible, then at least one of the points which is on the linear segment but out of the truncated-corridor should be added to $\mathcal{X}$ before the discretized problem is solved again. In our implementation, we compute all intersections between $\mathcal{L}$ and f, i.e we solve $l(x) - mx - b = 0$ with $x_1 \leqslant x \leqslant x_k$. Then for any two consecutive intersection points $(x_A, l(x_A))$ and $(x_B, l(x_B))$, we check if $](x_A, l(x_A)), (x_B, l(x_B))[$ is out of the corridor. If it is the case,the middlepoint of the interval $\frac{x_A + x_B}{2}$ is added to $\mathcal{X}$ before the discretized problem can be solved again. The same procedure is repeated for possible intersections with $\mathcal{H}$. If no point was added during this process,



(a) corridor considered (b) discretization and resulting linear piece (c) linear piece exits corridor

(d) new discretization and resulting linear piece (e) linear piece stays within corridor (f) final solution

Figure 7: Maximal linear piece on a corridor neither convex nor concave

then the solution of the discretized problem is feasible for the non-discretized problem. With such an implementation, no optimization solver is needed and a drastic reduction of the computing times can be achieved in comparison to the state-of-the-art.

When the solution of the discretized problem is feasible for the non-discretized problem, it is optimal with a precision level of $x_{k+1} - x_k$. Let ϵ be a target precision level. If $x_{k+1} - x_k > \epsilon$, then the point $\frac{x_k + x_{k+1}}{2}$ is added to $\mathcal{X}$, before re-solving the discretized problem. Note that in the later case, as a speed-up, O'Rourke's algorithm for solving the discretized problem at step $k+1$ can be warmstarted with the polyhedron P_k avoiding the unnecessary overhead of recomputing P_k.

The resulting iterative procedure solving the maximal linear segment problem in the general case is described in Algorithm 3 and illustrated on Figure 7. In an attempt to not obfuscate Algorithm 3 the warmstart discussed above is not included but is straightforward to implement.

The initial discretization of the corridor has non negligible impact in the run time of the algorithm. We could therefore try to estimate where the function varies the most to help the algorithm converge faster. However, we are not assuming differentiability in this section but rather looking for a general solution, therefore we simply use equidistant points to initialize our algorithm.

Algorithm 3 Solve the general maximal linear segment problem

Input: Corridor $\mathcal{C}$ defined by functions $h, l : \mathbb{D} = [x_-, x_+] \to \mathbb{R}$
Output: Optimal linear segment defined by endpoints (x_S, y_S) and (x_T, y_T)

$\mathcal{X} \leftarrow$ discretize corridor $\mathcal{C}$
FEASIBLE $\leftarrow$ true
$\{m, b, k\} \leftarrow$ O'Rourke's algorithm applied on $\mathcal{X}$
$\mathcal{I} \leftarrow$ Solve $l(x) - mx - b = 0$
$\mathcal{I} \leftarrow \mathcal{I} \cup$ Solve $h(x) - mx - b = 0$
for all (x_A, x_B) a set of consecutive values in $\mathcal{I}$ **do**
  if $(\frac{x_A+x_B}{2}, m\frac{x_A+x_B}{2} + b) \notin \mathcal{C}$ **then**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{\frac{x_A+x_B}{2}\}$
    FEASIBLE $\leftarrow$ false
  end if
end for
if FEASIBLE **then**
  if $x_{k+1} - x_k > \epsilon$ **then**
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{\frac{x_{k+1}+x_k}{2}\}$
    go to step 2
  else
    return Segment induced by x_-, x_k, m, b
  end if
end if

5 Speed-ups for uniform corridors

In this section, we look at how the previous algorithms for the maximal linear segment problem can be combined to obtain faster convergence on the corridor fitting problem in the special case of *uniform corridor*. A corridor is said to be uniform if it is defined by C^1 functions having the same concavity on $\mathbb{D}$, i.e at every $x \in \mathbb{D}$ the functions h and l are either both convex or both concave.

Remark. *In the case of $\mathcal{C}^2$ functions, the concavity condition can simply be stated as $l''(x)h''(x) \geqslant 0 \, \forall x \in \mathbb{D}$*

As this class of corridors includes approximations from relative and absolute pointwise errors, it covers most of the instances seen in practice.

5.1 A faster exact method

The corridor fitting problem can be solved on a uniform corridor using Algorithm 1 with the algorithm 3 as a subroutine, but a significant speed-up can be obtained by making use of the specific structure of uniform corridors. To do so, Algorithm 2 can be used on the convex and concave sub-corridors of $\mathcal{C}$, while Algorithm 3 is only used on sub-corridors that contain a change of concavity. The idea is to apply Algorithm 2 on $\mathcal{C}$ until a change of concavity is reached. Then remove the last segment computed and use Algorithm 3 to compute the maximal linear piece that crosses the change of concavity. Algorithm 2 is then called again, until the next change of concavity. This is summarized in Algorithm 4. The main advantage of this approach is that it utilizes the logarithmic time complexity of algorithm 2 for most of the domain while only using the slower but more general algorithm 3 when it's really needed.

In our implementation, the positions of the changes of concavity can be either provided by the user at the same time as the corridor, or be left to be computed by a dedicated algorithm. We have implemented a dedicated algorithm that requires the bounding functions of the corridor to both be C^2

Algorithm 4 Faster exact computation of an optimal PWL function fitting corridor $\mathcal{C}$

Input: Uniform corridor $\mathcal{C}$
Output: PWL function g defined by the set of linear segments P

while $\mathcal{C} \neq \varnothing$ **do**
  $\{\mathcal{C}_1, ..., \mathcal{C}_k\} \leftarrow$ Partition into convex or concave subcorridors of $\mathcal{C}$
  while $\mathcal{C}^c \neq \varnothing$ **do**
    $p^* \leftarrow$ Algorithm $2'(\mathcal{C}^c)$
    if $\mathcal{C}^c(p^*) \neq \mathcal{C}^c$ OR $\mathcal{C}^c(p^*) = \mathcal{C}$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup p^*$
      $\mathcal{C} \leftarrow \mathcal{C} \backslash \mathcal{C}^c(p^*)$
    end if
    $\mathcal{C}^c \leftarrow \mathcal{C}^c \backslash \mathcal{C}^c(p^*)$
  end while
  if $\mathcal{C} \neq \varnothing$ **then**
    $p^* \leftarrow$ Algorithm $3(\mathcal{C})$
    $\mathcal{P} \leftarrow \mathcal{P} \cup p^*$
    $\mathcal{C} \leftarrow \mathcal{C} \backslash \mathcal{C}(p^*)$
  end if
end while

instead of C^1. It finds the positions of changes of concavity by computing where the second derivative of a bounding function varnishes.

5.2 A heuristic introducing a very limited number of additional segments

In practice, it's often the case that a *good* solution to the corridor fitting problem is sufficient for our needs. Here we describe an algorithm that add at most as much additional segments as the number of change of concavities in the corridor but, as a trade-off, converges faster.

Let $\mathcal{C}$ be a uniform corridor we are looking to fit. $\mathcal{C}$ can be partitioned into k sub-corridors $\mathcal{C}_1, ..., \mathcal{C}_k$ such that each sub-corridor $\mathcal{C}_i, \forall i \in \{1...k\}$ is either convex or concave. A feasible solution of the corridor fitting problem on $\mathcal{C}$ is obtained by concatenating optimal solutions of the corridor fitting problem of all the sub-corridors. On each sub-corridor, we can use algorithm 2 to compute the maximal linear segments. This is summarized in Algorithm 5.

Algorithm 5 Heuristic computation of a PWL function fitting a uniform corridor $\mathcal{C}$

Input: Uniform corridor $\mathcal{C}$
Output: PWL function g defined by the set of linear pieces $\mathcal{P}$

$\{\mathcal{C}_1, ..., \mathcal{C}_k\} \leftarrow$ Partition into convex or concave subcorridors of$(\mathcal{C})$
for $i \in \{1...k\}$ **do**
  while $\mathcal{C}_v \neq \varnothing$ **do**
    $p^* \leftarrow$ Algorithm $2'$ $(\mathcal{C}_v)$
    $\mathcal{P} \leftarrow \mathcal{P} \cup p^*$
    $\mathcal{C}_v \leftarrow \mathcal{C}_v \backslash \mathcal{C}(p^*)$
  end while
end for

The advantage of this approach is that it is possible to compute the linearizarion in each sub-corridor in parallel. Also, the implementation is simpler and quicker as it only relies on Algorithm 2 which has a logarithmic time complexity for every segment. The drawback is that the number

of segments might not be optimal as there are segments that are not maximal at the junction of sub-corridors. However, the number of additional segment is tightly bounded as stated in Lemma 4.

Lemma 4. *Let $\overline{n}$ be the number of linear segments from Algorithm 5 and let n^* be the optimal number of linear segments, then $n^* \leqslant \overline{n} \leqslant n^* + k - 1$.*

Proof. Let $\mathcal{C}$ be a uniform corridor over $\mathbb{D} = [x_-, x_+]$. Let $B_1, B_2, \dots B_{k-1}$ represent the changes of concavity of $\mathcal{C}$, $B_0 = x_-$ and $B_k = x_+$. $\mathbb{D}$ can be partitioned as $\bigcup_{i=0}^{k-1}[B_{i-1}, B_i[\bigcup [B_{k-1}, B_k]$ such that the corridor is either convex or concave on each interval. Let g^* be the optimal PWL function fitting corridor $\mathcal{C}$ and having as breakpoints $x_0, \dots, x_{n*}$ and f be the PWL function obtained by Algorithm 5. Let S_i be the minimal x_j such that $x_j \geqslant B_i$. Finally, let n be the counting function for the number of segments in a piecewise linear function (so that we have $\nu(g^*) = n^*$). To make the notation less cumbersome, $\nu(g^*)$ will not count a segment made of a single point so that we can work with closed intervals. Since $[B_{i-1}, B_i]$ is either convex or concave, $\nu(f\big|_{[B_{i-1},B_i]})$ is optimal over this restricted domain and we therefore have

$$\nu(f\big|_{[B_{i-1},B_i]}) \leqslant \nu(g^*\big|_{[B_{i-1},B_i]}), \qquad \forall i \in \{1, ..., k\}$$

Since f and g^* share the same number of segments before the first change in concavity, we have equality for $i = 1$. Let us analyze the case where $i \neq 1$. The minimality of S_{i-1} implies that the interval $[B_{i-1}, S_{i-1}]$ is covered by a single segment in g^*. Therefore,

$$\nu(f\big|_{[B_{i-1},B_i]}) \leqslant \nu(g^*\big|_{[B_{i-1},B_i]}) \leqslant \nu(g^*\big|_{[S_{i-1},B_i]}) + 1, \qquad \forall i \in \{1, \dots, k\}$$

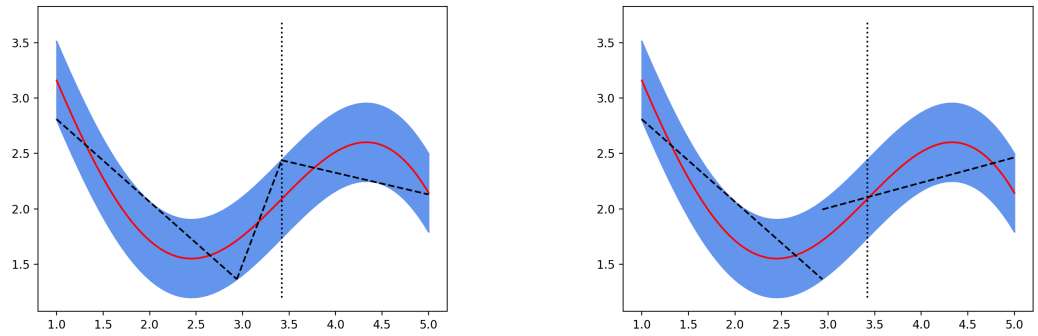
Noting that $\nu(f\big|_{[B_{i-1},B_i]})+\nu(f\big|_{[B_i,B_{i+1}]}) = \nu(f\big|_{[B_{i-1},B_{i+1}]})$ as there is no segment overlapping between two sub-corridors by the definition of the algorithm, we have that

$$\begin{aligned}
\nu(f) &= \sum_{i=1}^{k} \nu(f\big|_{[B_{i-1},B_i]}) \\
&= \nu(f\big|_{[B_0,B_1]}) + \sum_{i=2}^{k} \nu(f\big|_{[B_{i-1},B_i]}) \\
&= \nu(f\big|_{[x_-,B_1]}) + \sum_{i=2}^{k} \nu(f\big|_{[B_{i-1},B_i]}) \\
&\leqslant \nu(f\big|_{[x_-,B_1]}) + \sum_{i=2}^{k} \nu(g^*\big|_{[S_{i-1},B_i]}) + 1
\end{aligned}$$

Since the segment off g^* fitting $[B_i, S_{i+1}]$ is also contributing to $g^*\big|_{[S_i,B_i]}$

$$\begin{aligned}
&= \nu(f\big|_{[x_-,S_1]}) + \sum_{i=2}^{k} \nu(g^*\big|_{[S_i,S_{i+1}]}) + 1 \\
&= k - 1 + \nu(f\big|_{[x_-,S_1]}) + \sum_{i=2}^{k} \nu(g^*\big|_{[S_i,S_{i+1}]}) \\
&= k - 1 + \sum_{i=1}^{k} \nu(g^*\big|_{[S_i,S_{i+1}]}) \\
&= k - 1 + n^*
\end{aligned}$$

□



Heuristic Exact method

Figure 8: Comparison between both approaches

In addition of providing a guarantee on the quality of PWL obtained, Lemma 4 gives an easy way to compute lower bounds for the number of linear segments needed to solves the corridor fitting problem optimally.

To illustrate the differences between both algorithms presented in this section, Figure 8 compares the solution obtained by the algorithm 5 and 4 on the corridor derived from an absolute tolerance of 0.3 of $\sqrt{x}\sin x + x$ on interval $[1, 5]$. Here, a change in concavity happens at $x \approx 3.42$ as marked by the vertical doted line.

Remark. *As a consequence of Lemma 1, the PWL computed on uniform corridors happens to be continuous on each convex or concave sub-corridor and the only possible discontinuities occur in the junctions of sub-corridors. It also happens to be a convex (resp. concave) function on these convex (resp. concave) sub-corridors.*

6 Computational evaluation

6.1 Implementation

The package proposed is named LinA and is compatible with Julia 1.0.1 and onward. It is available at the following address: http://homepages.laas.fr/sungueve/LinA.html. It accepts as input the expression of a function, the type of error requested (relative or absolute) and optionally the type of over-/under-/approximation expected (by default approximation) and the algorithm desired (by default it uses the heuristic). It is also possible to provide directly the two functions defining the corridor to fit or even to define custom pointwise error types.

One of the key feature of the implementation is the use of symbolic differentiation which proved to be faster than other methods in our benchmarks for large instances. This is explained by the fact that the tangent functions are called so often that the higher initial cost of symbolic differentiation is overshadowed by the lower cost of each evaluation done in the different dichotomy searches. In some cases, where symbolic expressions are hard to obtain, the user can replace the expression by a native Julia function. In this case, the package will seamlessly use numerical differentiation through the package FowardDiff.jl [RLP16]. This is achieved through the extensive use of multiple dispatching which gives a very modular code and, in addition, allows to add custom error types easily (for the exact procedure, see the documentation). Another perk is the use of functors[1] to make the syntax as

[1] see https://docs.julialang.org/en/v1/manual/methods/

natural as possible as shown in Figure 9.

```
julia> using LinA

julia> f = Linearize(:(x^2.0+1),-3.0,3.0,Absolute(0.1))
7-element Array{LinA.LinearPiece,1}:
 -5.105572809000085 x -5.416718427000255 from -3.0 to -2.1055728090000847
 -3.316718427000254 x -1.6501552810007603 from -2.1055728090000847 to -1.2111456180001692
 -1.5278640450004222 x + 0.5164078649987369 from -1.2111456180001692 to -0.31671842700025316
 0.26099033699940966 x + 1.0829710109982338 from -0.31671842700025316 to 0.577708763999663
 2.049844718999242 x + 0.04953415699772967 from 0.577708763999663 to 1.4721359549995794
 3.838699100999076 x -2.5839026970027774 from 1.4721359549995794 to 2.366563145999497
 5.627553482998908 x -6.817339551003286 from 2.366563145999497 to 3.0

julia> f[1]
-5.105572809000085 x -5.416718427000255 from -3.0 to -2.1055728090000847

julia> f[1](-2.5)
7.347213595499959

julia> f(2)
5.093495504995374

julia> f[1].xMax
-2.1055728090000847
```

Figure 9: Simple example of the LinA syntax

The current implementation of LinA relies on the root finding functions available in the package Roots.jl (https://github.com/JuliaMath/Roots.jl). In particular, it uses the dichotomy search-based function. The function find_zeros is used to identify the location of changes of concavity for C^2 uniform corridor by computing the points where the second derivative of the input function vanishes but also in variety of contexts such as finding the intersection between a line and the corridor in Algorithm 3 (used in the exact method). Also, the Polyhedra.jl package [LDE+19] is used to compute the intersections of polyhedrons for Algorithm 3.

6.2 Benchmarks in approximation of nonlinear continuous functions

To assess the computational efficiency of LinA we perform a computational study on twelve nonlinear continuous functions from the literature and provided in Table 3, then compare the results obtained by LinA with the ones reported by [RK15], [RK19], [Ngu19] and [KM20]. Table 3 shows the characteristics of the machines used by the different authors. Table 4 and Table 5 report the number of linear pieces and the computing times for the piecewise linear approximation with four different absolute tolerance values, as in the literature : $\delta = 0.1$, 0.05, 0.01 and 0.005. Table 6 reports the computing times for the piecewise linear overestimation or underestimation of the three nonlinear functions that represent energy conversion functions in [Ngu19], with three different relative tolerance values : $\varepsilon =$ 0.01, 0.001, and 0.0001.

ref	function	$\mathbb{D}$	error	tol.*	[RK15]	[RK19]	[KM20]	[Ngu19]	LINA
(A-I)	x^2	$[-3.5; 3.5]$	absolute	a	✓		✓	✓	✓
(A-II)	$\ln(x)$	$[1; 32]$	absolute	a	✓	✓	✓	✓	✓
(A-III)	$\sin(x)$	$[0; 2\pi]$	absolute	a	✓			✓	✓
(A-IV)	$\tanh(x)$	$[-5; 5]$	absolute	a	✓			✓	✓
(A-V)	$\frac{\sin(x)}{x}$	$[1; 12]$	absolute	a	✓	✓		✓	✓
(A-VI)	$2x^2 + x^3$	$[-2.5; 2.5]$	absolute	a	✓	✓		✓	✓
(A-VII)	$e^{-x}\sin(x)$	$[-4; 4]$	absolute	a	✓	✓		✓	✓
(A-VIII)	$e^{-100(x-2)^2}$	$[0; 3]$	absolute	a	✓	✓		✓	✓
(A-IX)	$1.03e^{-100(x-1.2)^2} + e^{-100(x-2)^2}$	$[0; 3]$	absolute	a	✓	✓		✓	✓
(R-I)	$0.001x^3 - 0.024x^2 + 1.92x + 5.91$	$[1; 60]$	relative	o, u				✓	✓
(R-II)	$-0.005x^3 + 0.5x^2 - 0.8x + 10.0$	$[1; 60]$	relative	o, u				✓	✓
(R-III)	$0.0000002x^5 - 0.0000274x^4 + 0.00151450x^3 - 0.02453270x^2 + 1.92434870x + 5.90568630$	$[1; 60]$	relative	o, u				✓	✓

*tolerance: (a)pproximation, (o)vertimation, (u)nderestimation

Table 2: Benchmark : univariate nonlinear functions and tolerance types used

[RK15]	Computer	Intel(R) i7 single core 2.93 GHz, 12.0 GB RAM 64-bit Windows 7
	single node ?	yes
	Passmark cpu score	5366
	Software	GAMS 23.6, LindoGlobal 23.6.5
	Accuracy	10^{-5}
	Time limit	1800s per iteration
[RK19]	Computer	Intel 3.5 GHz, 32 GB of RAM
	single node ?	not specified
	Passmark cpu score	between 8687 and 28594
	Software	GAMS 24.8, LindoGlobal (for GO), CPLEX (for MILP)
	Accuracy	10^{-3}
	Time limit	24 hours per instance
[Ngu19]	Computer	Intel(R) Xeon(R) single core CPU E3- 1271 v3, 32 GB RAM (heuristic) Neos server (exact)
	single node ?	yes
	Passmark cpu score	10086 (heuristic); 6878, 7641, 21149 (exact)
	Software	GAMS 23.6, GAMS 24.9.2 r64480 single core, LindoGlobal (for GO), CPLEX 12.6 (for MILP)
	Accuracy	10^{-5}
	Time limit	1800s per iteration
[KM20]	Computer	not specified
	single node ?	not specified
	Passmark cpu score	not specified
	Software	GAMS (25.0.1), CPLEX (12.8.0.0), ANTIGONE v1.1, BARON 17.10.16, SCIP 4.0
	Accuracy	not specified
	Time limit	1200 s
This paper	Computer	Intel(R) Core(TM) i7-7700 CPU 3.60GHz, 16 GB of RAM
	single node ?	yes
	Passmark cpu score	8625
	Software	LINA
	Accuracy	10^{-5}
	Time limit	1800s per iteration

Table 3: Parameter settings per publication

Checkmarks ✓(resp. ✓) in Table 3 mean that the corresponding numerical results for all (resp. some) values of δ were reported by the authors. The first five columns of the table show: the reference by which the function will be referred to in the subsequent tables of results, the analytic expression of the function, the domain ($\mathbb{D}$) of the function, if the error considered is absolute or relative, and finally, if an approximation, overestimation or underestimation was applied. In Tables 4-6 the first column gives the reference of the continuous input function considered. In these tables, p is the number of convex or concave subintervals of $\mathbb{D}$, δ is the absolute tolerance value used, ϵ is the relative tolerance value used, n_* is the optimal number of linear pieces, n_- (resp. n_+) is the lower (resp. upper) bound returned when n_* could not be found. Note that [RK15] reported only the order of magnitude of the computing time for instances that could be solved to optimality. In their scale *frac* means "$\leqslant 1$", *few* means "$\geqslant 1$ and $\leqslant 10$". The instances with t.o.$^{1800s/it}$ (resp. t.o^{1200s}) are the ones for which the 1800 (resp. 1200) seconds time limit was reached during one of the iterations (resp. in total). All papers did not report results on all instances. The missing cases are denoted "-".

Results in Table 4 suggest that [KM20] may not be competitive with the other solution methods. [RK19] improved the results from [RK15], but there still remain instances for which the optimal PWL approximation could not be found, contrary to [Ngu19] and LinA. We can also observe that our heuristic provides lower bounds that are higher than the lower bounds from [RK19] for the cases that they could not solve to optimality.

Results in Table 5 show that a major improvement in the computing times is achieved: the computation of approximations that required hours of calculations in [RK15], [RK19] or hundred of seconds in [Ngu19] is done in one-tenth of a second with LinA. Results in Table 6 show that a drastic reduction of the computational times is also achieved when relative tolerance is used. It is to be noted that for running times so small, the operating system task scheduling has a non-negligible impact and therefore adds some randomness to the results which is why some high precision instances required less time than the lower precision ones in some cases. In any case, such major performance speed-up in the computation of the piecewise linear functions opens up new possibilities, such as the application of LinA to help solving with minimal efforts some instances of linearly constrained MINLPs with a nonlinear but separable objective-function. This is the case of the congested multicommodity network design problem used as an illustrative case study in the next subsection.

6.3 Application on linearly constrained MINLPs: the case of the multicommodity network design problem with congestion

To illustrate how LinA can be used as an effective pre-processing to derive high quality bounds with minimal effort for linearly constrained MINLPs, we consider the case of the multicommodity network design problem with congestion where there is a different nonlinear congestion function for each node of a graph. The problem was introduced by Paraskevopoulos *et al* [PGB16] to explicitly take into account the congestion occurring at nodes of networks used to represent a wide range of planning and operation management problems in transportation, telecommunications, logistics and production. The resulting MINLP is linearly constrained and its nonlinear objective-function is separable in a sum of positive univariate nonlinear terms. The state-of-the-art solution method for the problem is a mixed-integer second-order cone reformulation solved with a dedicated solver.

The LinA+MILP method consists in replacing each nonlinear term of the MINLP with its PWL underestimation computed with LinA, then solving the resulting MILP with a black box MILP solver. Details on the resulting MILP reformulation are provided in appendix. The computational evaluation of the method was done on the 258 instances from the cMCNDlib benchmark generated by [PGB16], partitioned into 43 groups of 6 instances each. These instances include either 20, 25, 30 or 100 nodes; a number of commodities ranging from 10 to 400 and a number of arcs from 100 to 700. All experimentations of [PGB16] were performed by using CPLEX 12.1 running on single thread, on a computing cluster with 2.4 GHz and 64 GB RAM, while our experiment was performed with CPLEX 12.8 using the computer described in Table 3 with an 1% relative underestimation tolerance in the

input function		δ	continuous approximation								nnc approximation			
			[RK15]			[RK19]			[KM20]		Exact[1]	Heuristic[1]		
ref	p		n_*	n_-	n_+	n_*	n_-	n_+	n_*	n_+	n_*	n_*	n_-	n_+
(A-I)	1	0.1	**8**			-			**8**		**8**	**8**		
		0.05	**12**			-				15	**12**	**12**		
		0.01	**25**			-					**25**	**25**		
		0.005	**35**			-			-		**35**	**35**		
(A-II)	1	0.1	**3**			**3**			**3**		**3**	**3**		
		0.05	**4**			**4**				4	**4**	**4**		
		0.01	**9**			**9**			-		**9**	**9**		
		0.005	**13**			**13**			-		**13**	**13**		
(A-III)	2	0.1	**5**			-			-		**5**		5	6
		0.05	**5**			-			-		**5**		5	6
		0.01	**13**			-			-		**13**		13	14
		0.005	**17**			-			-		**17**		17	18
(A-IV)	2	0.1	**3**			-			-		**3**		3	4
		0.05	**5**			-			-		**5**		5	6
		0.01	**9**			-			-		**9**		9	10
		0.005	**13**			-			-		**13**		13	14
(A-V)	4	0.1	**3**			**3**			-		**3**		2	5
		0.05	5			5			-		**4**		3	6
		0.01	9			9			-		**8**		7	10
		0.005	**12**			**12**			-		**12**		12	15
(A-VI)	2	0.1	**11**			**11**			-		**11**		11	12
		0.05	**15**			**15**			-		**15**		15	16
		0.01		15	**34**		31	**34**	-		**34**		34	35
		0.005		15	**47**		40	**47**	-		**47**		47	48
(A-VII)	3	0.1		4	**14**	**14**			-		**14**		14	16
		0.05		4	**19**	**19**			-		**19**		19	21
		0.01		4	**43**		34	**43**	-		**43**		43	45
		0.005		4	**61**		35	**61**	-		**61**		61	63
(A-VIII)	3	0.1	**4**			**4**			-		**4**		4	6
		0.05		4	6	**5**			-		**5**		4	6
		0.01		4	**11**	**11**			-		**11**		10	12
		0.005		4	**14**	**14**			-		**14**		14	16
(A-IX)	5	0.1	**7**			**7**			-		**7**		7	11
		0.05		7	11	**9**			-		**9**		7	11
		0.01		7	**21**	**21**			-		**21**		19	23
		0.005		7	28	**27**			-		**27**		27	31

[1] same results obtained in [Ngu19] and with LinA

Table 4: Number of linear segments obtained by [RK15], [RK19], [KM20] [Ngu19] and LinA for the piecewise linear approximation of nonlinear functions with absolute tolerance δ

input function		δ	continuous approximation			nnc approximation			
			[RK15]	[RK19]	[KM20]	[Ngu19]		LinA	
ref	p					Exact	Heuristic	Exact	Heuristic
(A-I)	1	0.1	Few sec.	-	350 s	-	-	0.003 s	1.5 ms
		0.05	Few sec.	-	t.o.^{1200s}	-	-	0.003 s	1.7 ms
		0.01	Hours	-	-	19 s	30 s	0.054 s	4.3 ms
		0.005	Hours	-	-	4 s	43 s	0.59 s	4.2 ms
(A-II)	1	0.1	Frac. sec.	0.1 s	26 s	-	-	0.006 s	1.3 ms
		0.05	Few sec.	0.1 s	t.o.^{1200s}	-	-	0.003 s	1.6 ms
		0.01	Sec	9.8 s	-	221 s	11 s	0.002 s	1.9 ms
		0.005	Sec	128.2 s	-	172 s	16 s	0.002 s	1.6 ms
(A-III)	2	0.1	Few sec.	-	-	-	-	0.12 s	2.0 ms
		0.05	Few sec.	-	-	-	-	0.056 s	2.1 ms
		0.01	Sec	-	-	57 s	12 s	0.09 s	2.8 ms
		0.005	Few min	-	-	68 s	17 s	0.058 s	2.4 ms
(A-IV)	2	0.1	Frac sec.	-	-	-	-	0.10 s	2.6 ms
		0.05	Few sec	-	-	-	-	0.039 s	2.6 ms
		0.01	Few sec	-	-	161 s	10 s	0.085 s	2.9 ms
		0.005	Few min	-	-	128 s	15 s	0.040 s	2.8 ms
(A-V)	4	0.1	Frac sec.	0.7 s	-	-	-	0.40 s	6.6 ms
		0.05	Sec.	6.5 s	-	-	-	0.37 s	2.6 ms
		0.01	Sec	49.7 s	-	143 s	11 s	0.38 s	5.7 ms
		0.005	Few min	35.8 s	-	181 s	15 s	0.34 s	5.7 ms
(A-VI)	2	0.1	Min	24.4 s	-	115 s	11 s	0.087 s	4.0 ms
		0.05	Few days	107.7 s	-	88 s	17 s	0.10 s	4.2 ms
		0.01	t.o.$^{1800s/it}$	6819.1 s	-	164 s	36 s	0.047 s	4.9 ms
		0.005	t.o.$^{1800s/it}$	35787.4 s	-	195 s	59 s	0.052 s	6.5 ms
(A-VII)	3	0.1	t.o.$^{1800s/it}$	311.5 s	-	226 s	17 s	0.10 s	4.1 ms
		0.05	t.o.$^{1800s/it}$	14514.6 s	-	287 s	28 s	0.15 s	4.4 ms
		0.01	t.o.$^{1800s/it}$	35411 s	-	268 s	52 s	0.17 s	10 ms
		0.005	t.o.$^{1800s/it}$	70313.1 s	-	869 s	72 s	0.13 s	5.7 ms
(A-VIII)	3	0.1	Sec	1.7 s	-	74 s	4 s	0.16 s	4.0 ms
		0.05	t.o.$^{1800s/it}$	5.3 s	-	83 s	6 s	0.096 s	4.3 ms
		0.01	t.o.$^{1800s/it}$	59.6 s	-	138 s	11 s	0.066 s	4.4 ms
		0.005	t.o.$^{1800s/it}$	247.2 s	-	1466 s	16 s	0.087 s	6.4 ms
(A-IX)	5	0.1	Few days	1.9 s	-	77 s	8 s	0.23 s	13 ms
		0.05	t.o.$^{1800s/it}$	12 s	-	64 s	12 s	0.18 s	12 ms
		0.01	t.o.$^{1800s/it}$	13873.4 s	-	114 s	23 s	0.20 s	11 ms
		0.005	t.o.$^{1800s/it}$	42068.4 s	-	784 s	27 s	0.19 s	13 ms

Table 5: Cpu times from [RK15], [RK19], [Ngu19] and LinA for the piecewise linear approximation of nonlinear functions with absolute tolerance δ

function ref	ϵ	[Ngu19]		LinA	
		Overestimation	Underestimation	Overestimation	Underestimation
(R-I)	0.01	8 s	8 s	0.17 μs	0.14 μs
	0.001	17 s	19 s	0.45 μs	0.14 μs
	0.0001	52 s	50 s	0.21 μs	0.16 μs
(R-II)	0.01	10 s	11 s	7.1 μs	7.5 μs
	0.001	25 s	27 s	8.2 μs	0.12 μs
	0.0001	73 s	74 s	0.13 μs	0.1 μs
(R-III)	0.01	14 s	16 s	7.4 μs	9.7 μs
	0.001	41 s	42 s	8.6 μs	9.6 μs
	0.0001	109 s	114 s	0.13 μs	0.13 μs

Table 6: Cpu times from [Ngu19] and LinA for under/overestimation with relative toleranceϵ

piecewise linearization phase by LinA, i.e $\epsilon = 0.01$.

The computational results displayed in Table 7 show that with a minimal of implementation efforts and without trying to take advantage of the structure of the problem, the straightforward LinA+MILP approach is competitive with the state-of-the-art method for solving the multicommodity network design problem with congestion.

7 Conclusion

In conclusion, the proposed algorithm solves the corridor fitting problem, outperforms what already exists to compute PWL approximations with predefined pointwise maximal error and does not require any optimization solver in contrast to the recent contributions of [RK15], [RK19] and [Ngu19]. Instances from the literature are solved in mere milli-seconds. We also demonstrated how an approach based on the linearization of MINLPs using this algorithm can be used to obtain state of the art results in a simple and almost *black box* manner. Further work could be done on this approach by exploring other types of errors when approximating a MINLP with a MILP. Another interesting perspective is to generalize these approaches to higher dimensions by approximating functions with hyper-planes.

For the Julia package, a further improvement would be to utilize the specific structure of the polyhedrons in the coefficient space while performing Algorithm 3 to obtain a sensible speedup while calculating the intersection as proposed in [O'R81].

8 Acknowledgements

This research benefited from the support of the FMJH Program PGMO, from the support of EDF-Thales-Orange. This research also benefited from the support of the program "Soutien à la Mobilité Internationale (SMI) de Toulouse INP" and from the Undergraduate Student Research Awards from the NSERC. We would also like to thank Jean Laprés-Chartrand for his fruitful discussions and his help during the implementation of the algorithms.

References

[AMM13] Hamed Ahmadi, Jos R. Mart, and Ali Moshref. Piecewise linear approximation of generators cost functions using max-affine functions. In *2013 IEEE Power Energy Society General Meeting*, pages 1–5, 2013.

Group of instances	[PGB16]		LinA+MILP	
	Average Time (sec)	Average Gap	Average Time (sec)	Average Gap
c100_400_10_F_L_10	3490.1	1.45%	1713.1	1.45%
c100_400_10_F_T_10	3602.1	2.81%	2196.0	2.46%
c100_400_10_V_L_10	111.3	0.01%	11.5	0.01%
c100_400_30_F_L_10	3615.9	1.08%	1640.4	1.04%
c100_400_30_F_T_10	3612.1	2.98%	3600.0	2.78%
c100_400_30_V_T_10	3600.9	0.14%	242.3	0.18%
c25_100_10_F_L_5	1.5	0.00%	5.9	0.00%
c25_100_10_F_T_5	3.0	0.00%	5.2	0.00%
c25_100_10_V_L_5	0.5	0.00%	0.4	0.01%
c25_100_30_F_L_5	21.5	0.01%	22.0	0.03%
c25_100_30_F_T_5	127.1	0.01%	21.6	0.02%
c25_100_30_V_T_5	4.7	0.01%	4.1	0.05%
c33	2.8	0.00%	3.4	0.01%
c35	6.5	0.01%	7.0	0.04%
c36	13.3	0.01%	11.3	0.03%
c37	3600.6	0.24%	3600.0	0.29%
c38	3600.8	1.69%	3600.0	1.70%
c39	1935.8	0.03%	2480.3	0.08%
c40	3601.0	1.24%	3600.1	1.28%
c41	2.4	0.01%	3.6	0.01%
c42	4.5	0.01%	12.0	0.01%
c43	5.2	0.01%	7.1	0.02%
c44	5.4	0.01%	10.8	0.02%
c45	1538.5	0.05%	1990.4	0.15%
c46	3600.9	1.74%	3600.0	1.79%
c47	961.6	0.01%	955.9	0.07%
c48	3600.8	0.86%	3600.0	0.91%
c49	73.4	0.01%	98.9	0.01%
c50	884.6	0.01%	1901.7	0.02%
c51	266.9	0.01%	125.2	0.03%
c52	3600.9	0.33%	3600.0	0.38%
c53	3467.1	0.15%	3304.9	0.20%
c54	3603.8	0.74%	3600.1	0.74%
c55	3126.5	0.20%	3098.7	0.25%
c56	3602.8	0.84%	3600.1	0.85%
c57	41.6	0.01%	130.5	0.02%
c58	26.5	0.00%	62.2	0.00%
c59	219.3	0.01%	424.2	0.01%
c60	101.1	0.01%	157.0	0.02%
c61	3617.4	0.36%	3600.1	0.45%
c62	3603.3	2.49%	3600.1	2.67%
c63	3603.5	0.30%	3600.2	0.35%
c64	3612.8	1.61%	3600.1	1.10%

Table 7: Results on instances from the congested multicommodity network design problem

[CN15] Eduardo Camponogara and Luiz Fernando Nazari. Models and algorithms for optimal piecewise-linear function approximation. *Mathematical Problems in Engineering*, 2015:9 pages, 2015.

[DT73] D.H. Douglas and Peucker T.K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Classics in Cartography: Reflections on Influential Articles from Cartographica*, 10:112–122, 1973.

[EF76] James E. Ertel and Edward B. Fowlkes. Some algorithms for linear spline and piecewise multiple linear regression. *Journal of the American Statistical Association*, 71(355):640–648, 1976.

[GMMS12] Björn Geißler, Alexander Martin, Antonio Morsi, and Lars Schewe. Using piecewise linear functions for solving minlps. In Jon Lee and Sven Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 287–314, New York, NY, 2012. Springer New York.

[GP83] F. Gritzali and G. Papakonstantinou. A fast piecewise linear approximation algorithm. *Signal Processing*, 5(3):221 – 227, 1983.

[HH21] F.J. Hwang and YH. Huang. An effective logarithmic formulation for piecewise linearization requiring no inequality constraint. *Computational Optimization and Applications*, 79:601 – 631, 2021.

[HV19] Joey Huchette and Juan Pablo Vielma. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools, 2019.

[KM20] L. Kong and C.T. Maravelias. On the derivation of continuous piecewise linear approximating functions. *INFORMS Journal On Computing*, 32(3):1–16, 2020.

[LABOO01] Koen Luwel, Leo A. Beem, Patrick Onghena, and Lieven Onghena. Susing segmented linear regression models with unknown change points to analyze strategy shifts in cognitive tasks. *Behavior Research Methods, Instruments, & Computers*, 33:470–478, 2001.

[LDE+19] Benot Legat, Robin Deits, Oliver Evans, Gustavo Goretkin, Twan Koolen, Joey Huchette, Daisuke Oyama, Marcelo Forets, guberger, Robert Schwarz, Elliot Saba, and Chase Coleman. Juliapolyhedra/polyhedra.jl: v0.5.1, May 2019.

[Ngu19] Sandra Ulrich Ngueveu. Piecewise linear bounding of univariate nonlinear functions and resulting mixed integer linear programming-based solution methods. *Euopean Journal of Operational Research*, 275:1058–1071, 2019.

[O'R81] Joseph O'Rourke. An on-line algorithm for fitting straight lines between data ranges. *Commun. ACM*, 24(9):574–578, September 1981.

[PGB16] Dimitris C. Paraskevopoulos, Sinan Grel, and Tolga Bektaş. The congested multicommodity network design problem. *Transportation Research Part E: Logistics and Transportation Review*, 85:166 – 187, 2016.

[RK15] Steffen Rebennack and Josef Kallrath. Continuous piecewise linear delta-approximations for univariate functions: Computing minimal breakpoint systems. *Journal of Optimization Theory and Applications*, 167(2):617–643, 2015.

[RK19] Steffen Rebennack and Vitaly Krasko. Piecewise linear function fitting via mixed-integer linear programming. *INFORMS Journal on Computing*, to appear, 2019.

[RLP16] J. Revels, M. Lubin, and T. Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv:1607.07892 [cs.MS]*, 2016.

[RP86] J. B. Rosen and P. M. Pardalos. Global minimization of large-scale constrained concave quadratic problems by separable programming. *Mathematical Programming*, 34(2):163–174, 1986.

[Tom74a] I. Tomek. Two algorithms for piecewise-linear continuous approximation of functions of one variable. *IEEE Transactions on Computers*, 23(4):445–448, 1974.

[Tom74b] Ivan Tomek. Piecewise-linear approximation with a bound on absolute error. *Computers and Biomedical Research*, 7(1):64 – 70, 1974.

[TV12] Alejandro Toriello and Juan Pablo Vielma. Fitting piecewise linear continuous functions. *European Journal of Operational Research*, 219(1):86–95, 2012.

[VN11] Juan Pablo Vielma and George L. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128(1):49–72, 2011.

[Wat98] G. A. Watson. Choice of norms for data fitting and function approximation. *Acta Numerica*, 7:337377, 1998.

[YLTP16] Lingjian Yang, Songsong Liu, Sophia Tsoka, and Lazaros G. Papageorgiou. Mathematical programming for piecewise linear regression analysis. *Expert Systems with Applications*, 44:156–167, 2016.

9 Appendix: solving the multicommodity network design problem with congestion using LinA+MILP

9.1 Methodology

The solution method consists in identifying, for a given value ϵ, a PWL underestimator $\underline{f}^{\epsilon}(x)$ that verifies equation (5).

$$f(x) - \epsilon|f(x)| \leqslant \underline{f}^{\epsilon}(x) \leqslant f(x), \quad \forall x \in \mathbb{D} \tag{5}$$

Let $(\underline{\mathrm{P}}^{\epsilon})$ be the MILP resulting from the replacement of all functions f with $\underline{f}^{\epsilon}$. Let c^* be the optimal cost of the initial MINLP and let $c(\underline{\mathrm{P}}^{\epsilon})$ denote the optimal costs of $\underline{\mathrm{P}}^{\epsilon}$. Solving $(\underline{\mathrm{P}}^{\epsilon})$ yields a lower bound for the MINLP at most $100\epsilon\%$ lower than its optimal cost. Then an upper bound is obtained by recomputing the cost of the solution of $\underline{\mathrm{P}}^{\epsilon}$ using the original cost function f. Let c^r be this recomputed cost. An upper and lower bound for the initial MINLP can therefore be obtained using equation (6).

$$c(\underline{\mathrm{P}}^{\epsilon}) \leqslant c^* \leqslant \min\{c^r, (1+\epsilon)c(\underline{\mathrm{P}}^{\epsilon})\} \tag{6}$$

9.2 Problem definition

Let A be the set of arcs, P be the set of commodities, N be the set of nodes. Each node $i \in N$ has a fixed cost E_i, an initial capacity C_i^0, an upgrade capacity C_i^{δ}, a delay cost D_i, a free flow classification delay F_i and a demand of commodity p denoted D_i^p. Each arc $(i,j) \in A$ has a fixed opening cost O_{ij} and a capacity U_{ij}. Each commodity $p \in P$ has a quantity to be shipped W_p and a unit routing cost over arc (i,j) denoted D_{ij}^p. The mathematical formulation of the cMND requires the following decision variables :

- continuous variables $x_{ij}^p \geqslant 0$ that specify the amount of flow of commodity $p \in P$ on arc $(i,j) \in A$
- binary variable y_{ij} equal to 1 if arc $(i,j) \in A$ is used and 0 otherwise
- binary variable z_i equal to 1 if node $i \in N$ is upgraded and 0 otherwise

The resulting mathematical formulation proposed by [PGB16] is:

$$\begin{cases}
\min \displaystyle\sum_{(i,j)\in A} O_{ij}y_{ij} + \sum_{(i,j)\in A}\sum_{p\in P} D_{ij}^p x_{ij}^p + \sum_{i\in N} E_i z_i + \sum_{i\in N} g_i(x,z) & & (7.1)\\
s.t. \displaystyle\sum_{j\in N_i^+} x_{ij}^p - \sum_{j\in N_i^-} x_{ji}^p = D_i^p, & i \in N, p \in P & (7.2)\\
x_{ij}^p \leqslant W^p y_{ij}, & (i,j) \in A, p \in P & (7.3)\\
\displaystyle\sum_{p\in P} x_{ij}^p \leqslant U_{ij} y_{ij}, & (i,j) \in A & (7.4)\\
\displaystyle\sum_{j\in N^-}\sum_{p\in P} x_{ji}^p \leqslant C_i^0 + C_i^{\delta} z_i, & i \in N & (7.5)\\
x_{ij}^p \geqslant 0, & (i,j) \in A, p \in P & (7.6)\\
y_{ij} \in \{0,1\}, & (i,j) \in A & (7.7)\\
z_i \in \{0,1\}, & i \in N & (7.8)
\end{cases} \tag{7}$$

The objective-function (7.1) minimizes the total cost of design, routing and capacity augmentation and cost congestion. The function $g_i(x,z)$ that appears in the last term of the objective function

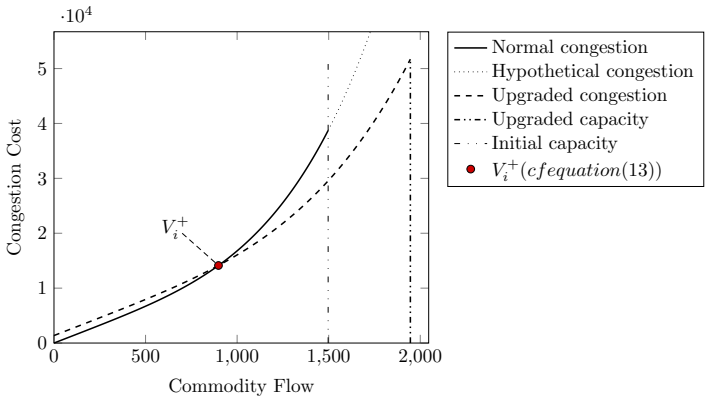


Figure 10: Example of congestion function $f_i(v_i)$

models congestion and is expressed with equation (8) where α and β are calibration parameters. Flow conservation constraints (7.2) ensure that the demands are satisfied for each node. Constraints (7.3) make sure that no flow of any commodity exists on an arc that is not selected. Constraints (7.4) limit the amount of flow on an arc to the capacity of the arc. Finally constraints (7.5) limit the total inflow of node i by its initial or extended capacity. Domain definition of variables are given by (7.6)-(7.8).

$$g_i(x,z) = D_i \left(F_i \sum_{j \in N^-} \sum_{p \in P} x_{ji}^p + \beta \frac{\left(\sum_{j \in N^-} \sum_{p \in P} x_{ji}^p \right)^{\alpha+1}}{\left(C_i^0 + C_i^\delta z_i \right)^\alpha} \right) \tag{8}$$

Formulation (7) is not suitable for a direct application of the LinA+MILP methodology because it contains bivariate nonlinear terms $g_i(x,z)$. Hereafter we propose a reformulation containing only univariate nonlinear terms.

Let us introduce a new variable v_i defined as follows: $v_i = \sum_{j \in N^-} \sum_{p \in P} x_{ji}^p, \forall i \in N$. Under the new variable definition the congestion functions can be expressed with equations (9). Let us define $|N|$ functions $h_i(v_i, z_i)$ as $h_i(v_i, z_i) = g_i(v_i, z_i) + E_i z_i$ with $i \in N$. These multivariate functions can be reduced to univariate functions as defined in Lemma 5.

$$g_i(v_i, z_i) = D_i \left(F_i v_i + \beta \frac{(v_i)^{\alpha+1}}{\left(\left(C_i^0 + C_i^+ z_i \right)^\alpha \right.} \right), \quad i \in N \tag{9}$$

Lemma 5. *For any $i \in N$, there exists a scalar $V_i^+ \in]0, C_i^0 + C_i^+]$ such that $h_i(v_i, z_i)$ can be reduced to a univariate function $f_i(v_i)$ that verifies equation (10).*

$$f_i(v_i) = \begin{cases} f_i^<(v_i) = D_i \left(F_i v_i + \beta \dfrac{(v_i)^{\alpha+1}}{(C_i^0)^\alpha} \right) & \text{if } v_i \leqslant V_i^+ \quad (10.1) \\ f_i^>(v_i) = E_i + D_i \left(F_i v_i + \beta \dfrac{(v_i)^{\alpha+1}}{\left(C_i^0 + C_i^+ \right)^\alpha} \right) & \text{otherwise} \quad (10.2) \end{cases} \tag{10}$$

Proof. From (10) we derive equation (11).

$$\left(f_i^<(v_i) - f_i^>(v_i) \right)' = (\alpha+1) v_i^\alpha D_i \beta \left(\frac{1}{(C_i^0)^\alpha} - \frac{1}{\left(C_i^0 + C_i^+ \right)^\alpha} \right) \tag{11}$$

Since $\alpha > 0, \beta > 0, D_i > 0, C_i^0 > 0, C_i^+ > 0$ and $v_i \geqslant 0$, then $(f_i^<(v_i) - f_i^>(v_i))' \geqslant 0$, meaning that the difference $f_i^<(v_i) - f_i^>(v_i)$ is non decreasing as illustrated on figure 10. Let V_i^+ be the value that verifies $f_i^<(V_i^+) = f_i^>(V_i^+)$ and can be computed with equation (12).

$$V_i^+ = \exp\left\{ \left(\frac{\ln E_i - \ln D_i - \ln \beta - \ln\left(\frac{1}{C_i^0} - \frac{1}{C_i^0 + C_i^+}\right)}{\alpha + 1} \right) \right\} \tag{12}$$

Therefore $\forall v_i \leqslant V_i^+$, so it is cost effective not to upgrade the capacity at node i, which leads to function $h_i(v_i, z_i) = f_i^<(v_i)$. On the other hand, $\forall v_i \geqslant V_i^+$, so it is cost effective to upgrade the node capacity of i, which leads to the function $h_i(v_i, z_i) = f_i^>(v_i)$. Taking into account both cases leads to function $h_i(v_i, z_i) = f_i(v_i)$ from equation (10).

□

The resulting linearly constrained non-convex MINLP formulation of the cMND with a sum of univariate nonlinear terms in the objective-function, is the following:

$$(\text{cMND}) \begin{cases} \min \sum_{(i,j)\in A} O_{ij} y_{ij} + \sum_{(i,j)\in A} \sum_{p\in P} D_{ij}^p x_{ij}^p + \sum_{i\in N} f_i(v_i) & & (13.1) \\ s.t. (7.2) - (7.4) & & (13.2) \\ \sum_{j\in N^-} \sum_{p\in P} x_{ji}^p = v_i & i \in N & (13.3) \\ (7.6) - (7.8) & & (13.4) \\ v_i \in [0, C_i^0 + C_i^\delta] & i \in N & (13.5) \end{cases} \tag{13}$$

Solving formulation (13) with the LINA+MILP methodology from Section 9.1 requires to replace each nonlinear function f_i with the piecewise linear functions $\underline{f}_i^\epsilon$ verifying equations (5).

9.3 Phase 1: Computation of the piecewise linear functions

We describe hereafter the procedure to obtain $\underline{f}_i^\epsilon$ using Algorithm 5. Each nonlinear function $f_i(v_i)$ is not differentiable at $v_i = V_i^+$, therefore it is not $\mathcal{C}^1$ on the domain $\mathbb{D} = [0, C_i^0 + C_i^+]$, but it is $\mathcal{C}^1$ and convex on each of the subdomains $\mathbb{D}_1 = [0, V_i^+]$ and $\mathbb{D}_2 = [V_i^+, C_i^0 + C_i^+]$. Let us consider four functions h_1, h_2, l_1, l_2 defined by equation (14). We therefore consider two convex corridors $\mathcal{C}_1$ and $\mathcal{C}_2$ defined by functions $h_1, l_1 : \mathbb{D}_1 \to \mathbb{R}$ and $h_2, l_2 : \mathbb{D}_2 \to \mathbb{R}$ respectively. The piecewise linear function $\underline{f}_i^\epsilon$ is obtained by concatenating an optimal piecewise linear function fitting corridor $\mathcal{C}_1$ with an optimal piecewise linear function fitting corridor $\mathcal{C}_2$.

$$\begin{cases} h_1(x) = f_i(x) = f_i^<(x) & x \in [0, V_i^+] & (14.1) \\ h_2(x) = f_i(x) = f_i^>(x) & x \in [V_i^+, C_i^0 + C_i^+] & (14.2) \\ l_1(x) = (1-\epsilon) f_i(x) = (1-\epsilon) f_i^<(x) & x \in [0, V_i^+] & (14.3) \\ l_2(x) = (1-\epsilon) f_i(x) = (1-\epsilon) f_i^>(x) & x \in [V_i^+, C_i^0 + C_i^+] & (14.4) \end{cases} \tag{14}$$

Let K_i^0 and K_i^+ be the number of linear pieces of the optimal piecewise linear functions fitting corridors $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. Let K_i be the number of linear pieces of function $\underline{f}_i^\epsilon$. Therefore $K_i = K_i^0 + K_i^+$. In the remainder of the section, the k^{th} linear piece of $\underline{f}_i^\epsilon$ is defined by its endpoints [$V_i^{\min}$ and $V_i^{\max}$], its slope A_i^k and its y-intercept B_i^k.

9.4 Phase 2: MILP formulation

We derive the MILP by replacing all functions f_i with $\underline{f}_i^{\epsilon}$ and introducing one binary variable s_i^k per linear segment, which will be equal to 1 if the segment is chosen in the solution and 0 otherwise. We also introduce one continuous variable v_i^k per linear segment, equal to the flow entering node i if the value of the flow belongs to the interval $[V_i^{\min}, V_i^{\max}]$ and 0 otherwise. The resulting MILP formulation of $\underline{(\text{cMND})^{\epsilon}}$ is the following:

$$
\underline{(\text{cMND})^{\epsilon}} \begin{cases}
\min \sum_{(i,j)\in A} O_{ij} y_{ij} + \sum_{(i,j)\in A} \sum_{p\in P} D_{ij}^p x_{ij}^p + \sum_{i\in N} \sum_{k\in K} A_i^k v_i^k + \sum_{i\in N} \sum_{k\in K} B_i^k s_i^k & (15.1) \\
s.t. (7.2) - (7.4) & (15.2) \\
\sum_{j\in N^-} \sum_{p\in P} x_{ji}^p = \sum_{k=1}^{K_i} v_i^k & i \in N \quad (15.3) \\
V_i^{\min} s_i^k \leqslant v_i^k \leqslant V_i^{\max} s_i^k & i \in N, k \in \{1, ..., K_i\} \quad (15.4) \\
\sum_{k=1}^{K_i} s_i^k = 1 & i \in N \quad (15.5) \\
(7.6) - (7.8) & (15.6) \\
0 \leqslant v_i^k \leqslant C_i^0 + C_i^+ & i \in N, k \in \{1, ..., K_i\} \quad (15.7) \\
s_i^k \in \{0, 1\} & i \in N \quad (15.8)
\end{cases} \tag{15}
$$

The objective-function (15.1) minimizes the total design, routing, capacity and congestion costs. Constraints (15.2) link variables x and y. Constraints (15.3) link variables x and v. Constraints (15.4) link variables v and s. Constraints (15.5) ensure that only one linear segment is chosen per node. Constraints (15.6)-(15.8) are the domain definition constraints of all variables.

Formulation (15) can be solved with any blackbox MILP solver to obtain a lower bound of the cMND. Then an upper bound can be obtained by computing the value of the expression (13.1) given the values assigned to variables x, y in the optimal solution of (15), and the values of variables Z obtained using equation (16).

$$
z_i = \begin{cases}
0, & \text{if } \sum_{k=1}^{K_i} v_i^k \leqslant C_i^0 \quad (16.1) \\
1 & \text{otherwise} \quad (16.2)
\end{cases} \tag{16}
$$

In this case, the number of additional binary variables increases linearly with the number of segments. Various other piecewise linear representations have been studied for example by [VN11], [HV19], [HH21] including one where the number of binary variables increases only logarithmically with the number of linear pieces. More efficient MILP solution methods could certainly be obtained by investigating more advanced MILP formulations, or decomposition-based solution methods such as branch-and-price, but this was not the focus of this paper. Also, it is well known that binary variables can be avoided when representing the piecewise linear underestimator of a convex objective-function in a minimization problem. In the case of the cMCND the nonlinear functions result from the concatenation of two convex functions. This could be taken advantage of to limit the number of binary variables to one per nonlinear function.