Piecewise linear bounding of univariate nonlinear functions and resulting mixed integer linear programming-based solution methods.

Sandra Ulrich Ngueveu

Piecewise linear bounding of univariate nonlinear functions and resulting mixed integer linear programming-based solution methods

Sandra Ulrich Ngueveu
LAAS–CNRS, Université de Toulouse, CNRS, INP, F-31400 Toulouse, France
ngueveu@laas.fr

May 30, 2018

Abstract

Various optimization problems result from the introduction of nonlinear terms into combinatorial optimization problems. In the context of energy optimization for example, energy sources can have very different characteristics in terms of power range and energy demand/cost function, also known as efficiency function or energy conversion function. Introducing these energy sources characteristics in combinatorial optimization problems, such as energy resource allocation problems or energy-consuming activity scheduling problems may result into mixed integer nonlinear problems neither convex nor concave. Approximations via piecewise linear functions have been proposed in the literature. Non-convex optimization models and heuristics exist to compute optimal breakpoint positions under a bounded absolute error-tolerance. We present an alternative solution method based on the upper and lower bounding of nonlinear terms using non necessarily continuous piecewise linear functions with a relative epsilon-tolerance. Conditions under which such approach yields a pair of mixed integer linear programs with a performance guarantee are analyzed. Models and algorithms to compute the non necessarily continuous piecewise linear functions with absolute and relative tolerances are also presented. Computational evaluations performed on energy optimization problems for hybrid electric vehicles show the efficiency of the method with regards to the state of the art.

Keywords: OR in energy , Nonlinear programming , Combinatorial optimization , Piecewise linear bounding

2010 MSC: 90C30 , 90C59 , 90C90 , 68W25

1 Introduction

Various optimization problems result from the introduction of nonlinear terms into combinatorial optimization problems and can therefore be modeled as mixed integer nonlinear problems (MINLP). In the context of energy optimization, such nonlinear terms model energy conversion or demand/cost functions. Let us consider, for example, energy optimization in hybrid electric vehicles (HEV). In such vehicles the electrical powertrain system has multiple energy sources that it can gather power from to satisfy the propulsion power requested by the vehicle

at each instant. The problem usually consists in finding at each instant the optimal power split between the multiple energy sources to satisfy the power demand of a driver on a predefined road section. The variant of the problem called *offline* assumes that the power demand profile is known a priori. The objective is to minimize the total fuel consumption of the vehicle performing a predefined mission, taking into account the characteristics and the limitations of each energy source, such as the energy losses happening during any energy transfer. Let (P) denote such problem for a HEV operating with two energy sources: (1) a Fuel Cell (FC) stack able to produce power from $P^1_{\min}$ up to $P^1_{\max}$ at each instant $i \in \{1...I\}$, (2) a Storage Element (SE) able to retrieve power up to $P^2_{\min}$ and to produce power up to $P^2_{\max}$ at each instant $i \in \{1...I\}$. The amount of energy stored in the SE is also called State Of Charge (SOC). To avoid a premature aging of SE, its state of charge is only allowed to vary between $E_{\min}$ and $E_{\max}$, typically 25% and 100% of its energy capacity. Problem (P) can be modeled with equations (1)-(5) where x^1_i, x^2_i and x^3_i are the amount of energy produced by the FC, produced by the SE, and retrieved by the SE at instant $i \in \{1...I\}$. Problem (P) is a (MI)NLP because of nonlinear energy conversion functions f^1, f^2, and f^3 continuous on intervals $[P^1_{\min}, P^1_{\max}]$, $]0, P^2_{\max}]$, and $]0, -P^2_{\min}]$ respectively, and verifying $f^1(0) = f^2(0) = f^3(0) = 0$, often with a discontinuity at $P^1_{\min}$ or 0 that requires the introduction of binary variables to be modeled. The mathematical model can be expressed as follows:

$$\text{(P)} \quad \min \sum_{i=1}^{I} f^1(x^1_i) \quad //\textit{minimize total cost on FC} \tag{1}$$

subject to (s.t.)

$$x^1_i + x^2_i - x^3_i \geq P_i, \quad \forall i \in \{1...I\} \quad //\textit{power demand satisfaction} \tag{2}$$

$$\sum_{i=1}^{I} (f^2(x^2_i) - f^3(x^3_i)) \leq 0 \qquad //\textit{final SOC} \geq \textit{initial SOC} \tag{3}$$

$$E_0 - E_{\max} \leq \sum_{k=1}^{i} (f^2(x^2_k) - f^3(x^3_k)) \leq E_0 - E_{\min}, \quad \forall i \in \{1...I\} \quad //\textit{SOC limits} \tag{4}$$

$$x^1_i \in \{0\} \cup [P^1_{\min}, P^1_{\max}], x^2_i \in [0, P^2_{\max}], x^3_i \in [0, -P^2_{\min}] \quad \forall i \in \{1...I\} \quad //\textit{domain definition.} \tag{5}$$

General MINLPs are NP-hard, but some subclasses such as convex MINLP may be easier to solve although still NP-hard. Convex MINLPs have a convex objective function to minimize, and their constraint functions are all convex and upper bounded. In these instances, when the integrity constraints are relaxed the feasible set is convex, and there exist efficient algorithms to solve the resulting convex NLP. However, the (MI)NLP modeling an energy optimization problem may be neither convex (nor concave) even when all energy conversion functions are convex or concave. Therefore, only small instances may be tractable using standard MINLP solvers. Several real-world applications have been addressed using piecewise linear approximations of the nonlinear functions of the MINLP to obtain a MILP which is easier to solve (see for example (Borghetti et al., 2008), (Boukouvala et al., 2016), (Camponogara et al., 2007), (D'Ambrosio et al., 2010)). Such approach presents the main advantage of producing solutions faster than purely MINLP-based approaches if not too many additional

binary variables or combinatorial conditions have been introduced in the process, meaning that the number of pieces of the piecewise linear (pwl) functions used should be limited. (Geißler et al., 2012) explain that the approach suffers from a few drawbacks because the nature of the nonlinearities inherent to the problem may be lost and the solutions obtained from the MILP may have no meaning for the MINLP. If the solution obtained is not satisfactory in terms of feasibility or optimality, then a new pwl approximation may be performed using a higher number of pieces, to obtain a new MILP to solve. This yields an iterative solution procedure with a number of iterations unknown a priori which translates into high computing times, either because several iterations needed to be performed, or because an unnecessary large number of pieces were introduced upfront, resulting into an unnecessarily large MILP that required a high solution time.

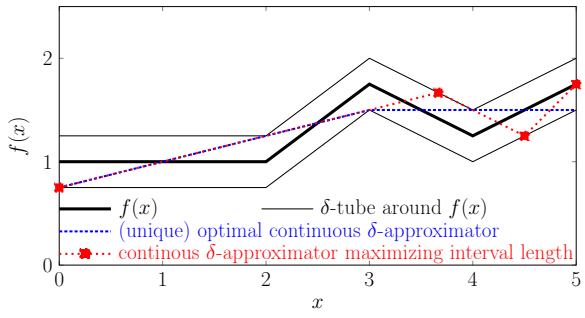
As an alternative we propose a straightforward two-phase solution method. The first phase consists in bounding each nonlinear term from above and below using a pair of piecewise linear functions satisfying conditions that will be specified in the core of the paper. Contrary to most publications on piecewise linear approximation which focus on the minimization of the approximation error for a given number of pieces or breakpoints that may or may not be equidistant, we aim at minimizing the number of pieces for a given error bound. The second phase of the solution method consists in solving two MILPs obtained from the replacement of the nonlinear functions with one of the two piecewise linear functions. This paper addresses the first phase of the iterative procedure, assuming that the resulting MILPs can be solved efficiently with a MILP solver. If it is not the case, then a specific solution method may need to be designed for them. An example of such case is presented in Ngueveu et al. (2016), which focused on the solution of the resulting MILP without specifying how to obtain the piecewise linear functions.

2 State of the art

Several publications exists on the application of piecewise linear (pwl) approximation on nonlinear univariate functions to solve MINLP problems, but the issue addressed in the large majority of them is to minimize the approximation error given a predefined number of breakpoints or pieces. To the best of our knowledge, only three papers focused on the specific problem of minimizing the number of breakpoints for a given tolerance or bounded approximation error.

Rosen and Pardalos (1986) were the first to propose the computation of breakpoints for a given error tolerance for concave quadratic functions. The pwl interpolators were built using equidistant breakpoints, and because the input functions were concave their interpolators were underestimators. They identified conditions on the number of breakpoints required to achieve a given error tolerance. Geißler et al. (2012) showed that certain cases of general MINLPs can be solved by just applying techniques purely from the mixed integer linear programming by approximating the nonlinearities with pwl functions. They proposed to compute a priori errors for pwl approximations or a priori errors for over- and under- pwl estimators. However, they did not focus on the computation of optimal (minimal) breakpoint positions.

Rebennack and Kallrath (2015) propose two exact approaches and two heuristics for the computation of optimal continuous pwl approximators for univariate continuous functions



$$f(x) = \begin{cases} 1 & \text{if } x \in [0,2[\\ 0.75x - 0.5 & \text{if } x \in [2,3[\\ -0.5x + 3.25 & \text{if } x \in [3,4[\\ 0.5x - 0.75 & \text{if } x \in [4,5] \end{cases} \quad (6)$$

Figure 1: Maximizing the length of the intervals successively is not optimal, in general (source: Rebennack and Kallrath (2015), page 628)

over a compactum $\mathbb{D} = [X_-, X_+]$. Their algorithms handle more general functions than the ones of Rosen and Pardalos (1986). In addition, their breakpoints are distributed freely and shifts from the function are allowed at breakpoints, which were shown to be important degrees of freedom contributing to a significant reduction of the number of breakpoints, up to 80% in some cases. The work of Rebennack and Kallrath (2015) differs from Geißler et al. (2012) in the following aspects. Geißler et al. (2012) do not target on computing the minimal number of breakpoints and do not consider shift variables at breakpoints whereas Rebennack and Kallrath (2015) do so. Since their work is close to ours in some aspects, let us focus on their contributions before highlighting the main differences and contributions of this paper.

Rebennack and Kallrath (2015) show that ensuring that the approximator and the original function do not deviate more than a predefined tolerance δ from each other leads to sets of constraints which have to hold over a continuum, resulting in a semi-infinite programming (SIP) problem denoted OBSC for "Optimal Breakpoint System using a Continuum approach for x". The authors show that it is NP-hard to compute a δ-approximator for an arbitrary continuous function and propose an iterative solution procedure based on the evaluation of the continuum conditions only on a discrete set of grid points, resulting into a MINLP model which is a relaxation of the SIP. The feasibility of the resulting solution with regards to OBSC is then evaluated by solving an NLP on each interval corresponding to a line-segment of the pwl approximator, to compute the true maximal deviation between the approximator and the original function. The algorithm stops if the true deviation is less or equal to δ on all line-segments. In this case the solution obtained is optimal for OBSC. Otherwise the grid is refined to obtain a new MINLP to be solved. OBSC and the MINLPs are in general too large and difficult to solve to optimality, therefore the authors proposed two heuristic methods. The heuristics methods were based on the successive computation of the breakpoints, from X_- to X_+, maximizing at each iteration the length of the interval on x. This meant solving at each iteration a problem denoted BSB that can be expressed as follows: given the breakpoint x_i ending the i^{th} line-segment (which corresponds by continuity to the beginning of the $i + 1^{\text{th}}$ line-segment), compute the next breakpoint x_{i+1} (end of the $i + 1^{\text{th}}$ line-segment) so as to maximize x_{i+1} while ensuring a deviation of at most δ between the original function and the linear approximation on interval $[x_i, x_{i+1}]$. Rebennack

and Kallrath (2015) provided a counter-example showing that maximizing the length of the intervals do not necessarily lead to an optimal breakpoint system i.e., to a δ-approximator with the least number of breakpoints, contrary to what is stated for example in Frenzen et al. (2010). The counter-example is illustrated on Figure 1. It shows, for a function $f(x)$ that the authors proposed with $[X_-, X_+] = [0, 5]$ and $\delta = 0.25$, that the unique optimal 0.25-approximator uses two line-segments whereas maximizing the interval length successively from X_- to X_+ produces a 0.25-approximator using three line-segments. The Forward Heuristic with Moving Breakpoints (FHMB) solved each BSB problem to optimality with the iterative "grid discretization + NLP solution" approach. Its main limitation was the necessity to solve several NLPs. The α-Forward Heuristic with Backward Iterations (FHBI) solved each BSB problem heuristically by trying different decreasing values for x_{i+1} with a predefined step parameter α. FHBI solved less NLPs than FHBM, and therefore required less computing time, but FHBI obtained better solutions. Using any of the heuristics, it is possible to obtain breakpoint positions satisfying the required δ-tolerance, and provide an upper bound on the minimal number of breakpoints.

It is worth mentioning that there exists publications on piecewise linear approximation with a minimum number of pieces given a predefined bound on the absolute error in the fields of data reduction, pattern recognition or classification, and ECG waveform preprocessing (Tomek (1974a), Tomek (1974b), Gritzali and Papakonstantinou (1983)). The main difference with our problem is that those publications consider as an input a discrete set of points. Their objective is to find the piecewise linear function with a minimum number of pieces, such that the error for each of the sample points is less than the allowed value δ. Even in cases where an analytical expression of a continuous function was available, the function was sampled and the approximation was performed on the set of sample points. The algorithms proposed in these research fields did not ensure the respect of the predefined approximation error on the entire interval $[X_-, X_+]$ and are therefore not applicable to our problem. Finally, the field of piecewise linear approximation of planar curves could be mentionned (Dunham (1986), Papakonstantinou et al. (1994)), with applications related to shape analysis or pattern classification, since a nonlinear function $f(x)$ could be represented as a parametric curve $x(t) = t, y(t) = f(t)$. However, the algorithms proposed in that research field are not applicable to our problems for two main reasons: (i) input curves considered are discrete or digitised, not continuous ones as ours, and (ii) the error metric is the Euclidean distance, which does not correspond to the approximation error between the original function and the piecewise linear function.

In the view of the state-of-the-art, the contributions of the paper are the following: (1) instead of using continuous pwl δ-approximators, we propose to approximate general univariate continuous functions with non necessarily continuous piecewise linear δ-approximators, adding an additional degree of freedom to obtain a breakpoint system of equal or less line-segments, (2) we prove that when discontinuity is allowed, maximizing the interval length produces an optimal pwl δ-approximator, leading to an exact solution procedure based on the iterative solution of adapted BSB problems, (3) we introduce relative ϵ-tolerance and show the benefit of using it instead of the absolute δ-tolerance, (4) models and algorithms to compute the discontinuous pwl under- and over-estimator with absolute or relative tolerance and with an additive worst case guarantee are presented, (5) for solving MINLP involving nonlinear univariate energy conversion functions, we propose a solution method based on the

upper and lower bounding of energy conversion expressions using non necessarily continuous piecewise linear functions with a relative ϵ-tolerance, and the solution of a pair of mixed integer linear programs, (6) we show that such approach yields a performance guarantee when the nonlinearity is restricted to the objective function, and, finally (7) computational results on energy optimization problems for hybrid electric vehicles illustrate the efficiency of the method in comparison to state-of-the-art methods, including solution procedures based on approximations with absolute δ-tolerance.

3 Non necessarily continuous pwl δ-approximation of continuous nonlinear functions

Let $f : \mathbb{D} = [X_-, X_+] \to \mathbb{R}$ be a function on the compact interval $\mathbb{D} \subset \mathbb{R}$. A function $g : \mathbb{D} = [X_-, X_+] \to \mathbb{R}$ is a pwl function with $n_g \in \mathbb{N}$ line-segments if $\exists a \in \mathbb{R}^{n_g}, b \in \mathbb{R}^{n_g}, x^{\min} \in [X_-, X_+]^{n_g}$ and $\forall i \in [1...n_g], \exists x_i^{\max} \in]x_i^{\min}, X_+]$, such that the following equations are verified:

$$g(x) = a_i x + b_i, \qquad \forall i \in [1...n_g], \forall x \in [x_i^{\min}, x_i^{\max}] \tag{7}$$

$$x_i^{\max} = x_{i+1}^{\min}, \qquad \forall i \in [1...n_g - 1] \tag{8}$$

$$x_1^{\min} = X_- \tag{9}$$

$$x_{n_g}^{\max} = X_+ \tag{10}$$

Such pwl function is said to be defined by $G = \bigcup_{i=1}^{n_g}([a_i, b_i], [x_i^{\min}, x_i^{\max}])$, and the two endpoints $x_i^{\min}$ and $x_i^{\max}$ of each line-segment i are called breakpoints.

A pwl function g is:

- continuous iff it verifies all following equations:

$$a_i x_i^{\max} + b_i = a_{i+1} x_{i+1}^{\min} + b_{i+1}, \forall i \in [1, n_g - 1] \tag{11}$$

- or discontinuous if it does not verify all equations (11), i.e. $\exists j \in [1, n_g - 1]$ such that $(a_i x_i^{\max} + b_i) \neq (a_{i+1} x_{i+1}^{\min} + b_{i+1})$
- or non necessarily continuous (nnc) if the satisfaction of equations (11) is neither required nor forbidden

Definition 3.1 *(δ-approximator): A pwl function $g : \mathbb{D} = [X_-, X_+] \to \mathbb{R}$ is called δ-approximator of a function $f : \mathbb{D} = [X_-, X_+] \to \mathbb{R}$ with $\delta > 0$, iff :*

$$max_{x \in \mathbb{D}} |g(x) - f(x)| \leq \delta \tag{12}$$

Proposition 3.2 *Any optimal pwl continuous δ-approximator with n^* line-segments can be converted into a pwl non necessarily continuous $\delta-$approximator with $n \leq n^*$ line-segments where the projection of the first line-segment on interval $\mathbb{D}$ is of maximal length.*

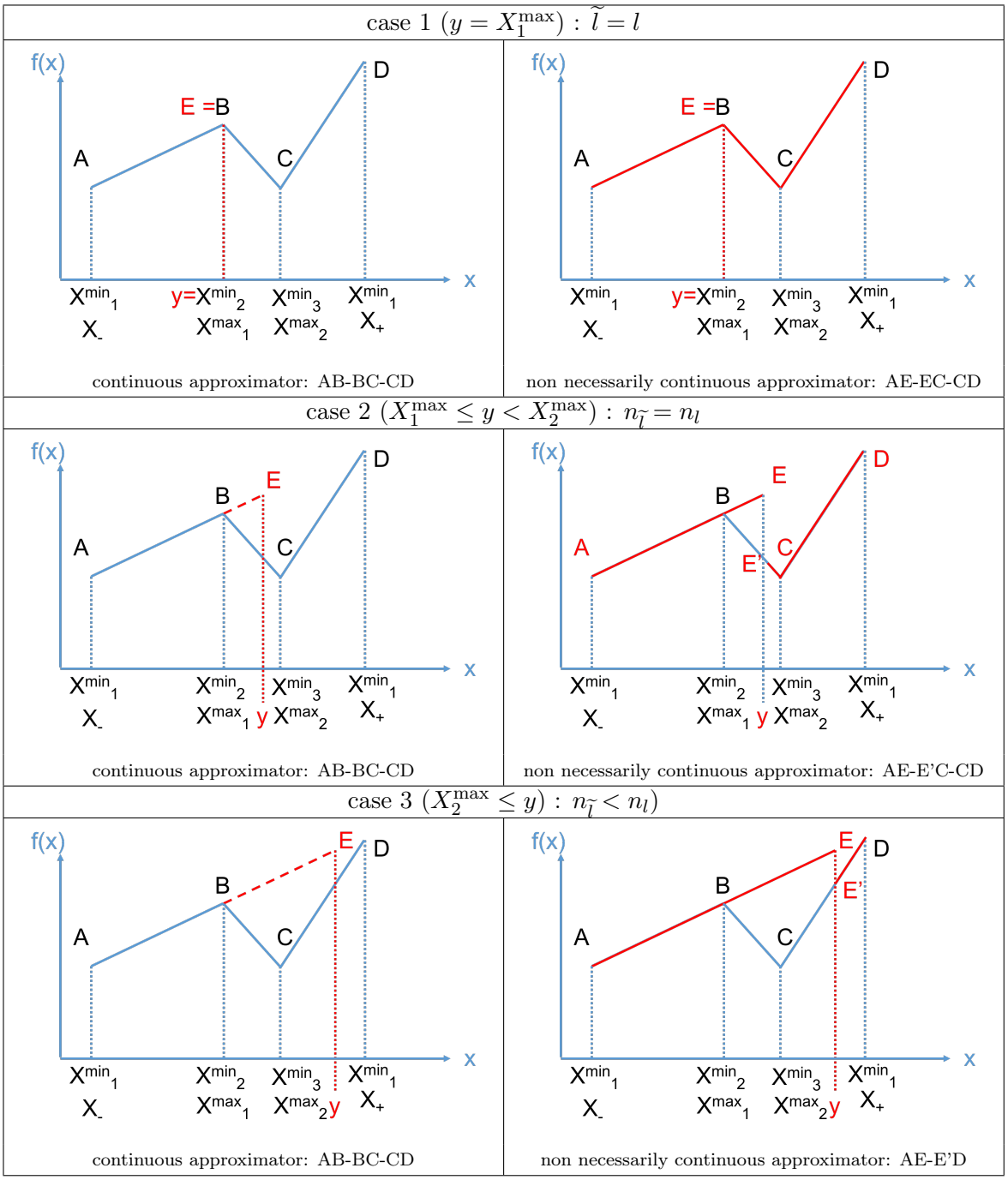


Figure 2: Maximization of the first interval (projection on $\mathbb{D}$ of the first line-segment)

Proof Let $l = \bigcup_{i=1}^{n_l}([a_i, b_i], [x_i^{\min}, x_i^{\max}])$ be an optimal continuous δ-pwl approximator ($n_l = n^*$) of a continuous function $f : \mathbb{D} = [X_-, X_+] \to \mathbb{R}$. Let $y \in \mathbb{D}$ be the solution value of the problem defined as follows:

$$(\text{PY}) \begin{cases} \max \quad y & //\textit{maximize ending breakpoint} \\ \text{s.t.} & \\ \quad |a_1 x + b_1 - f(x)| \leq \delta, \quad \forall x \in [x_1^{\min}, y] & //\delta\textit{-approximation constraint} \\ \quad y \in [x_1^{\min}, x_i^{\max}] & //\textit{domain definition} \end{cases}$$

Let $q \in [1...n_l]$ be the piece number that verifies $x_q^{\min} \leq y \leq x_q^{\max}$. Breakpoint $x_1^{\max}$ verifies $|a_1 x_1^{\max} + b_1 - f(x)| \leq \delta$, therefore $y \geq x_1^{\max}$. If $y = x_1^{\max}$, then $\widetilde{l} = l$ is a pwl continuous δ-approximator of f with $n = n^*$ pieces where the projection of the first line-segment on interval $\mathbb{D}$ is of maximal length. Otherwise $y > x_1^{\max}$, in which case a discontinuous pwl δ-approximator $\widetilde{l}$ with $n_{\widetilde{l}} \leq n^*$ line-segments where the projection of the first line-segment on interval $\mathbb{D}$ is of maximal length can be defined by $\widetilde{L} = \bigcup_{i=1}^{n_{\widetilde{l}}}([\widetilde{a}_i, \widetilde{b}_i], [\widetilde{x}_i^{\min}, \widetilde{x}_i^{\max}]) = ([a_1, b_1], [x_1^{\min}, y]) \cup ([a_q, b_q], [y, x_q^{\max}]) \cup \left(\bigcup_{i=q+1}^{n_{\widetilde{l}}}([a_i, b_i], [x_i^{\min}, x_i^{\max}])\right)$. ∎

Theorem 3.3 *For any continuous function $f : \mathbb{D} = [X_-, X_+] \to \mathbb{R}$ and any scalar $\delta > 0$, there exists an optimal non necessarily continuous pwl δ-approximator g defined by $G = \bigcup_{i=1}^{n_g}([a_i, b_i], [x_i^{\min}, x_i^{\max}])$, such that each line-segment i has a maximal length projection on the interval $[x_i^{\min}, X_+]$.*

Proof There exists a continuous δ-approximator function for any continuous function f on a compactum $\mathbb{D}$ and any scalar $\delta > 0$ (Duistermaat and Kol, 2004). Proposition 3.2 can be applied iteratively on an optimal continuous $\delta-$approximator to obtain an optimal non necessarily continuous pwl $\delta-$approximator with intervals of maximal length on $\mathbb{D}$. ∎

Figure 2 illustrates the three possible cases related to Proposition 3.2. Figure 3(a) illustrates the implications of Theorem 3.3 for the function f defined with equation (6): maximizing the length of intervals leads to a discontinous pwl 0.25-approximator of two line-segments, which is the optimal number of line-segments for any non necessarily continuous pwl 0.25-approximator of f. In this example optimal continuous and non necessarily continuous pwl approximators have the same number of line-segments. Figure 3(b) illustrates an example where it is not the case: one has more line-segments than the other. In the general case, it can be easily proven that given a continuous function $f : \mathbb{D} = [X_-, X_+] \to \mathbb{R}$ and given a scalar $\delta \in \mathbb{R}^+$, if n_{nnc} is the number of line-segments of an optimal non necessarily continuous pwl δ-approximator of f and if n_c is the number of line-segments of an optimal continuous pwl δ-approximator of f, then $\left\lceil \frac{n_c+1}{2} \right\rceil \leq n_{nnc} \leq n_c$.

The mathematical models and algorithms from Rebennack and Kallrath (2015) for computing continuous pwl δ-approximators are modified to compute non necessarily continuous pwl δ-approximators by considering two shifts per breakpoint instead of one, namely s_b^- used for the line-segment ending a breakpoint b and s_b^+ used for the line-segment starting at breakpoint b. In particular, thanks to Theorem 3.3, Algorithm 1 resulting from the adaptation of Forward Heuristic with Moving Breakpoints Rebennack and Kallrath (2015) is a now exact method for computing optimal non necessarily continuous δ-approximators. Its main limitation is the necessity to solve an SIP, for example with an iterative grid and NLP-based procedure as described in Algorithm 2. The tightest approximator computation given the

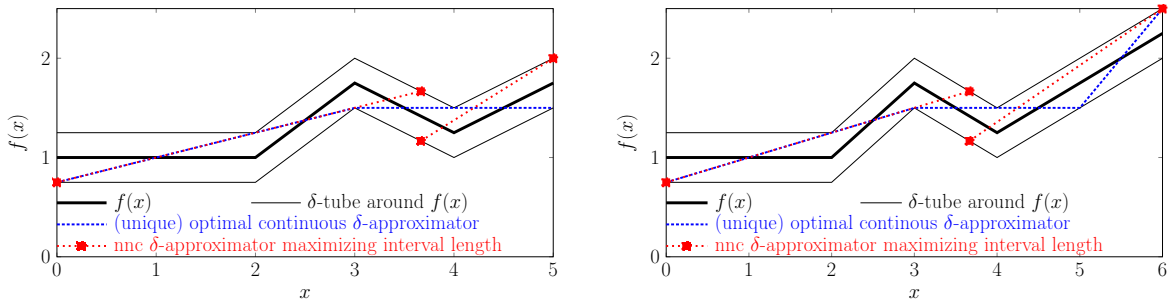


Figure 3: Maximizing the length of the intervals successively is optimal for non necessarily continuous δ-approximators

Algorithm 1 Optimal non necessarily continuous pwl approximation

Input: Function f; domain $\mathbb{D} = [X_-; X_+]$; scalar $\delta > 0$

Output: pwl function g defined by G

1: $n_g := 0; y = X_-; G := \emptyset; x^{\text{end}} := X_+$

2: **while** $y < X_+$ **do**

3: $\quad x^{\text{begin}} := y$

4: $\quad$ solve the following SIP to obtain a, b, and y:

$$\max y \qquad //\textit{maximize ending breakpoint} \tag{13}$$

$$s.t. \quad |ax + b - f(x)| \leq \delta, \forall x \in [x^{\text{begin}}, y] \qquad //\textit{δ-approximation constraint} \tag{14}$$

$$a \in \mathbb{R}, b \in \mathbb{R}, y \in [x^{\text{begin}}, x^{\text{end}}] \qquad //\textit{domain definition} \tag{15}$$

5: $\quad G := G \cup ([(a, b), (x^{\text{begin}}, y)])$

6: $\quad n_g := n_g + 1$

7: **end while**

Algorithm 2 Iterative grid and NLP-based procedure to solve SIP (13)-(15)

Input: Function f; limits x^{begin}, x^{end}; scalar $\delta > 0$; parameters $n_I^{\text{init}} \geq 2$, $\omega > 1$
Output: slope a, y-intercept b, next breakpoint y

1: initialize gap $:= +\infty$; grid size $n_I := n_I^{\text{init}}/\omega$ and grid $N_I = \{1 \dots n_I\}$
2: **while** gap $> \delta$ **do**
3: update grid size $n_I := n_I\omega$ and grid $N_I := \{1 \dots n_I\}$
4: solve the following NLP to obtain y, s_0, and s_y in order to compute the coefficients $a := \frac{f(y)+s_y-x^{\text{begin}}-s_0}{y-x^{\text{begin}}}$ and $b := f(x^{\text{begin}}) + s_0 - ax^{\text{begin}}$:

$$\max y \qquad \text{//maximize ending breakpoint} \tag{16}$$

$$s.t. \quad \left| s_0 + f(x^{\text{begin}}) + \tfrac{i(s_y+f(y)-f(x^{\text{begin}})-s_0)}{|N_I|+1} - f(x_i) \right| \leq \delta, \forall i \in N_I \qquad \text{//}\delta\text{-approximation on grid} \tag{17}$$

$$x_i = x^{\text{begin}} + \tfrac{i}{|N_I|+1}(y - x^{\text{begin}}), \forall i \in N_I \qquad \text{//grid computation} \tag{18}$$

$$s_0 \in [-\delta, \delta], s_y \in [-\delta, \delta], y \in [x^{\text{begin}}, x^{\text{end}}], \text{ and } x_i \in \mathbb{R}, \forall i \in N_I. \qquad \text{//domain definition} \tag{19}$$

5: compute the gap $:= \max_{x \in [x^{\text{begin}}, y]} |ax + b - f(x)|$
6: **end while**

optimal number breakpoints computed with Algorithm 1 would also require the solution of NLPs (Rebennack and Kallrath (2015)). Section 4 presents, among other things, an exact method applicable to convex and concave functions that does not need to solve any SIP or any NLP. The method also has an additive worst case guarantee for functions that are neither convex nor concave but that can be decomposed into convex or concave pieces, if the decomposition is part of the input.

4 From pwl δ-approximation to pwl δ-bounding

A benefit of computing δ-approximators optimal in terms of number of line-segments is to minimize the number of additional binary variables added to obtain the MILP resulting from the replacement of nonlinear terms with their pwl approximators. A drawback of applying δ-approximation is that the optimal solution of the resulting MILP can be infeasible with respect to the original MINLP. In this case the solution may not be of interest for the practitioners who formulated the original problem. A usual alternative is to use over- and under-estimators defined as follows.

Definition 4.1 *(δ-underestimator): A pwl function $\underline{l} : \mathbb{D} \to \mathbb{R}$ is a δ-underestimator of a function $f : \mathbb{D} \to \mathbb{R}$ with $\delta \in \mathbb{R}^+$ iff $\underline{l}(x) \leq f(x) \leq \underline{l}(x) + \delta, \forall x \in \mathbb{D}$.*

Definition 4.2 *(δ-overestimator): A pwl function $\overline{l} : \mathbb{D} \to \mathbb{R}$ is a δ-overestimator of a function $f : \mathbb{D} \to \mathbb{R}$ with $\delta \in \mathbb{R}^+$ iff $\overline{l}(x) \geq f(x) \geq \overline{l}(x) - \delta, \forall x \in \mathbb{D}$.*

In the context of energy optimization for hybrid electric vehicles for example, replacing each nonlinear energy loss function with its pwl δ-overestimator yields a $\overline{\text{MILP}}$ solution where a sufficient or excess amount of energy is produced at each time-step. Since excess energy can

be dissipated as heat (in a resistance inserted in the braking system or in mechanical brakes present for security reasons) therefore the $\overline{\text{MILP}}$ solution is applicable on a test bench or on the real world hybrid electric vehicle considered. Having ensured the feasibility of the solution obtained, it is of interest to provide an estimate of the quality of the solution with respect to the optimum of the original MINLP. This can be done using a pwl δ-underestimator of the energy loss function and solving the resulting $\underline{\text{MILP}}$ problem to obtain a lower bound of the MINLP. Each nonlinear function is therefore bounded from above and below with two pwl functions, yielding two MILPs whose optimal solutions cost verify: $z_{\underline{\text{MILP}}} \leq z_{\text{MINLP}} \leq z_{\overline{\text{MILP}}}$. The resulting solution method is denoted PLB+MILP.

Given an optimal continuous pwl δ-approximator g of a function f, an optimal continuous pwl 2δ-over- (resp. under-) estimator can be obtained from shifting g by δ (resp. $-\delta$), as stated by Rebennack and Kallrath (2015) i.e. $g(x) + \delta$ (resp. $g(x) - \delta$) is a 2δ-over (resp. under-) estimator of f. This result can be extended to non necessarily continuous pwl approximators and estimators. In addition, in the case of energy optimization for example, the MINLP can be neither convex nor concave even though the individual nonlinear energy loss functions or energy demand/cost conversion functions are convex or concave. Yet, if a nonlinear continuous function $f : \mathbb{D} \to \mathbb{R}$ is convex (resp. concave) over $\mathbb{D}$ with a derivative efficiently computable at any point of $\mathbb{D}$, then taking advantage of the following equation, it is possible to compute an optimal δ-underestimator (resp. δ-overestimator) without solving any SIP or even NLP:

$$f \text{ is convex over } \mathbb{D} \Leftrightarrow f(y) \geq f(x) + f'(x)(y - x), \forall x, y \in \mathbb{D} \tag{20}$$

Shifting can then be applied to obtain an optimal δ-overestimator (resp. δ-underestimator).

Let us focus, for example, on the computation of the optimal δ-underestimator of a nonlinear continuous convex function f with a derivative efficiently computable. The reasoning hereafter can be adapted to compute the optimal δ-overestimator of concave functions. Thanks to Theorem 3.3 the objective at each iteration i of the algorithm is to maximize $x_i^{\max}$ for a given $x_i^{\min}$. Each line-segment i is tangent to f (otherwise it can be replaced with a line-segment of identical slope but tangent to f). Therefore the objective at iteration i is to identify a tangent point $q_i \in [x_i^{\min}, x_i^{\max}]$ that defines the slope $a_i = f'(q_i)$ and the y-intercept $b_i = f(q_i) - f'(q_i)q_i$ of line-segment i so that $x_i^{\max}$ is maximized given $x_i^{\min}$. Given q_i, the error in function of y defined as $f(y) - (f'(q_i)y + f'(q_i)q_i)$ increases when y decreases if $y \leq q_i$, or when y increases if $y \geq q_i$. Therefore, for a given q_i, checking whether the δ-approximation constraint is verified at points $x_i^{\min}$ and $x_i^{\max}$ is sufficient to ensure the validity of the constraint on the entire interval $[x_i^{\min}, x_i^{\max}]$. In this context, the following proposition can be enunciated:

Proposition 4.3 *Finding the line-segment that maximizes $x_i^{\max}$ for a given $x_i^{\min}$ is equivalent to solving sequentially two separate problems: (1) maximizing q_i for the given $x_i^{\min}$, and then (2) maximizing $x_i^{\max}$ for the previously computed q_i.*

Proof Let $x_i^{\max}(q_i^{(k)})$ be the maximal value possible of $x_i^{\max}$ for a given $q_i^{(k)}$. It suffices to prove that $x_i^{\max}(q_i^{(2)}) \geq x_i^{\max}(q_i^{(1)}) \Rightarrow q_i^{(2)} \geq q_i^{(1)}$. This is done using equation (20) which leads to $0 \leq f(y) - (a_i^{(2)}y + b_i^{(2)}) \leq f(y) - (a_i^{(1)}y + b_i^{(1)}), \forall y \geq q_i^{(2)}$. ∎

Algorithm 3 Optimal non necessarily continuous pwl δ-underestimator

Input: Convex function f; domain $\mathbb{D} = [X_-; X_+]$; scalar $\delta > 0$
Output: pwl function g defined by G

1: $n_g := 0; y = X_-; G := \emptyset$
2: **while** $y < X_+$ **do**
3: $x^{\text{begin}} := y$
4: solve the following problem to get q and set $a := f'(q)$, $b := f(q) - f'(q)q$

$$(\text{PQ}) \begin{cases} & \max q \quad //\textit{maximize tangent point} \\ \text{s.t.} & \\ & f(x^{\text{begin}}) - (f(q) + f'(q)(x^{\text{begin}} - q)) \leq \delta \quad //\delta\textit{-approximation constraint} \\ & q \in [x^{\text{begin}}, X_+] \quad //\textit{domain definition} \end{cases}$$

5: solve the following problem to obtain y

$$(\text{PY}) \begin{cases} & \max y \quad //\textit{maximize ending breakpoint} \\ \text{s.t.} & \\ & f(y) - (ay + b) \leq \delta \quad //\delta\textit{-approximation constraint} \\ & y \in [q, X_+] \quad //\textit{domain definition} \end{cases}$$

6: $G := G \cup ([(a, b), (x^{\text{begin}}, y)])$
7: $n_g := n_g + 1$
8: **end while**

Algorithm 4 Solve problem (PQ)

Input: Convex function f; x^{begin};X_+; $\delta > 0$; $\alpha \in \mathbb{N}$
Output: tangent point q, slope a, y-intercept b

1: {q, a, b, 0, 0}=Dichotomy_q(f,x^{begin}, X_+,δ,α,x^{begin}, X_+, 0, 0, 0) {use Algorithm 6}

Algorithm 5 Solve problem (PY)

Input: Convex function f;X_+; $\delta > 0$; $\alpha \in \mathbb{N}$; tangent point q, slope a, y-intercept b
Output: next breakpoint y

1: {q, 0, 0}=Dichotomy_y(f,x^{begin}, X_+,δ,α,q, X_+, 0, 0, 0) {use Algorithm 7}

Algorithm 6 Dichotomy_q

Input: Convex function f; x^{begin};X_+; $\delta > 0$; $\alpha \in \mathbb{N}$; $B_{\min}$, $B_{\max}$; q; a; b
Output: tangent point q, slope a, y-intercept b, lower limit $B_{\min}$, upper limit $B_{\max}$

if $B_{\max} - B_{\min} \leq 10^{-\alpha}$ **then**
  $q = B_{\min}; a = f'(q); b = f(q) - f'(q)q$
  return q, a, b
else
  $\tilde{a} = f'(x^{\text{begin}}); \tilde{b} = f(x^{\text{begin}}) - f'(q_i)q_i$
  if $f(x^{\text{begin}}) - (\tilde{a}x^{\text{begin}} + \tilde{b}) \leq \delta$ **then**
    $B_{\min} = B_{\max}; B_{\max} = X_+$
  else
    $B_{\max} = B_{\min} + 0.5(B_{\max} - B_{\min})$
  end if
  $\{q, a, b, B_{\min}, B_{\max}\}$=Compute_$q(f, x^{\text{begin}}, X_+, \delta, \alpha, B_{\min}, B_{\max}, q, a, b)$
end if

Algorithm 7 Dichotomy_y

Input: Convex function f; X_+; $\delta > 0$; $\alpha \in \mathbb{N}$; q; a; b; $B_{\min}$, $B_{\max}$
Output: next breakpoint y, lower limit $B_{\min}$, upper limit $B_{\max}$

if $B_{\max} - B_{\min} \leq 10^{-\alpha}$ **then**
  $y = B_{\max}$
  return y
else
  if $f(B_{\max}) - (aB_{\max} + b) \leq \delta$ **then**
    $B_{\min} = B_{\max}; B_{\max} = X_+$
  else
    $B_{\max} = B_{\min} + 0.5(B_{\max} - B_{\min})$
  end if
  $\{y, B_{\min}, B_{\max}\}$=Compute_$y(f, X_+, \delta, \alpha, q, a, b, B_{\min}, B_{\max})$
end if

Algorithm 3 summarizes the resulting procedure that computes an optimal δ-underestimator of a nonlinear convex function f with a derivative efficiently computable at any point of $[X_-, X_+]$. Problems (PQ) and (PY) can be solved to optimality with dichotomy search methods as illustrated by Algorithms 6 and 7 where parameter α is the number of significant decimals for q_i, $x_i^{\min}$, and $x_i^{\max}$. Notice that there is no need to solve an SIP or NLP. It can be noted that any two consecutive line-segments $([a_i, b_i], [x_i^{\min}, x_i^{\max}])$ and $([a_{i+1}, b_{i+1}], [x_{i+1}^{\min} = x_i^{\max}, x_{i+1}^{\max}])$ of the pwl function obtained verify $a_i \neq a_{i+1}$ and $a_i x_i^{\max} + b_i \leq a_{i+1} x_{i+1}^{\min} + b_{i+1}$. Their supporting straight lines intersect on a point C of coordinates (x_C, y_C), such that $y_C = a_i x_C + b_i = a_{i+1} x_C + b_{i+1}$ and $x_i^{\min} \leq x_C \leq x_i^{\max}$ and $f(x_C) - y_C \leq \delta$. Consequently, the non necessarily continuous pwl δ-under-estimator computed with Algorithm 3 can be converted into a continuous pwl δ-under-estimator with the same number of line-segments.

Algorithm 3 can be adapted to compute directly an optimal δ-overestimator of a convex function: a tangent point q_i in this case defines the slope $a_i = f'(q_i)$ and the y-intercept $b_i = f(x_i^{\min}) - f'(q_i) x_i^{\min}$. It can also be adapted for the pwl bounding of concave functions with derivatives efficiently computable. Let us consider the case of continuous functions that are neither concave nor convex but that can be decomposed into p pieces, each piece being either concave or convex. If the decomposition is part of the input, then pwl bounding the function can be done by bounding each piece separately using the algorithms previously described, then aggregating the pwl functions obtained. To that end, let $k_i \in \mathbb{D}, \forall i \in \{1..p+1\}$ be a set of points, such that (i) on each interval $[k_i, k_{i+1}]$ the function $f(x)$ is either convex or concave, (ii) $k_1 = X_-$, (iii) $k_{p+1} = X_+$, and (iv) $k_i < k_{i+1}, \forall i \in \{1..p\}$. For each interval $[k_i, k_{i+1}]$, let g_i be the optimal pwl over-(resp. under-) estimator of f computed with the algorithms from Section 4. Let g be an over (resp. under) estimator of function f on interval $\mathbb{D}$, resulting from the union of pwl functions g_i, i.e. $g = \cup_{i \in \{1..p\}} g_i$. Let n_{g_i} (resp. n_g) be the number of line-segments of g_i (resp. g). If n^* is the number of line-segments of any optimal pwl over-(resp. under-) estimator of f on interval $\mathbb{D}$, then $n_g = \sum_{i=1}^{p} n_{g_i}$ and the following equation is verified, providing an additive worst case guarantee:

$$n^* \leq n_g \leq n^* + p - 1 \tag{21}$$

5 Drawbacks and limitations of pwl δ-bounding

When applying PLB+MILP, choosing relevant δ values is a challenging issue. The chosen value should be orders of magnitude smaller than the resulting solution cost to provide an acceptable precision level. After solving the MILP, verifying whether the chosen tolerance value δ was sufficiently small in the view of the resulting solution costs is straightforward. If it was not the case then the value of δ is decreased before a new round of pwl bounding then MILP solution. Such iterative procedure is not satisfactory because the number of iterations is unknown a priori and the pwl bounding and/or the MILP solution may require significant computational efforts during each single iteration. An alternative is to identify a target precision Δ of the MILP solution with regards to the optimal MINLP solution value and precompute a priori the corresponding δ values in function of the chosen Δ. When nonlinearity occurs in the constraints, the link between δ and Δ is not obvious. But even when nonlinearity occurs only in the objective function, δPLB+MILP with a target precision Δ presents significant limitations described in the remainder of this Section.

5.1 Data dependence leading to multiple δ values

Let us consider the problem (1)-(5) for a specific HEV, which, for simplicity, is assumed to have an ideal supercapacitator ($f^2(x) = x, f^3(x) = x$). Therefore the nonlinearity in the problem comes from the objective function which is composed of I univariate positive nonlinear terms $f^1(x)$. Ensuring a final precision of at least Δ requires to bound f^1 with a tolerance $\delta = \Delta/I$. Even if the same target precision Δ is chosen, different power profiles translate into different δ values. Two power profiles with the same time horizon I, may also require different δ values because different power profiles may translate into solution costs which can differ significantly in order of magnitude, requiring different Δ values, and, therefore, different δ values.

To summarize, for any new power profile provided, the pwl bounding of the same function f^1 may need to be redone with a value of δ suitable to the new data set, even though the nonlinear terms themselves remained unchanged. In addition, since the δ errors on the nonlinear terms are additive, a longer time horizon means a higher number of univariate terms ($f^1(x_i)$) in equation (1), which means that a smaller tolerance δ may be required. This translates into an increase of the number pieces for the pwl functions, and, therefore, more binary variables which adds to the complexity of a MILP that was already penalized by the fact that long time horizons meant more decision variables x_i^1, x_i^2, x_i^3. These all contribute to a substantial reduction of the size of instances that can be solved efficiently.

5.2 Solution dependence leading to unknown δ values

There exists several cases of MINLP where knowing Δ is not sufficient to infer the corresponding δ values. Let us consider, for example, problem (CF) modeling a scheduling problem with a single energy source. There are a set of activities $\mathcal{A}$, each activity a having a release date r_a, a due date d_a, a duration p_a, and an instantaneous energy demand b_a. A constant term α_{at} with $a \in \mathcal{A}$ and $t \in \mathcal{T}$ is equal to 1 if $t \in [r_a, d_a[$ and 0 otherwise. The efficency function of the energy source used to satisfy the total demand of activities scheduled at each time period is denoted ρ, i.e. a cost or energy consumption of $\rho(x)$ produces an amount of usable energy x. Therefore ρ is defined on $[0, \sum_{a\in\mathcal{A}} b_a]$ and verifies $\rho(0) = 0$. The goal is to schedule the tasks so as to minimize the total energy cost. A resulting mathematical model requires binary decision variable x_{at} that is equal to 1 iff activity a is active at instant $t \in \mathcal{T}$. Continuous variables w_t represent the total energy demand at instant $t \in \mathcal{T}$. Nonlinearity comes from the objective function which is comprised of $|\mathcal{T}|$ univariate nonlinear terms. The resulting problem is:

$$(\text{CF}) \min \quad \sum_{t\in\mathcal{T}} \rho(w_t) \quad //\textit{minimize total energy cost} \tag{22}$$

s.t.

$$\sum_{t\in\mathcal{T}} \alpha_{at} x_{at} \geq p_a, \qquad \forall a \in \mathcal{A} \qquad //\textit{execute activity } a \textit{ during } p_a \textit{ time periods} \tag{23}$$

$$w_t - \sum_{a\in\mathcal{A}} b_a x_{at} = 0, \qquad \forall t \in \mathcal{T} \qquad //\textit{link variables } x_{at} \textit{ and } z_t \tag{24}$$

$$x_{at} \in \{0, 1\}, \quad \forall a \in \mathcal{A}, t \in \mathcal{T} \quad //\textit{domain definition} \tag{25}$$

$$w_t \in \mathbb{R}^+, \qquad \forall t \in \mathcal{T}. \qquad //\textit{domain definition} \tag{26}$$

Let $(\text{CF})_\Delta$ refer to the MILP derived from problem (CF) with a target tolerance Δ. Identifying a relevant δ value for obtaining $(\text{CF})_\Delta$ is not straightforward. Indeed, in the general case, the duration of the schedule for the optimal solution is not known a priori. The only information available is the time horizon $|\mathcal{T}|$, but this value may be far from the real ending time of the optimal schedule. As an illustration, consider an instance with a time horizon $|\mathcal{T}| = 100$ having an optimal solution with an ending time of 10 (i.e. an optimal solution x^* that verifies $\sum_{a\in\mathcal{A}} x^*_{a,10} \geq 1$ and $\sum_{a\in\mathcal{A}}\sum_{t>10} x^*_{at} = 0$). In this case only 10 terms are active in the objective function, therefore bounding each term with a tolerance $\delta = \Delta/10$ should have been sufficient to achieve the requested global tolerance Δ. Instead, because the duration is unknown a priori, each term of the objective function has to be bounded with precision $\delta = \Delta/100$. In summary, a pwl bounding tolerance 10 times smaller than necessary would be requested, leading to pwl bounding functions with a higher number of line-segments, leading to a higher number of binary variables, leading to MILPs more time consuming to solve than necessary given the tolerance target Δ.

6 Using ϵ-relative tolerance

To counter the drawbacks identified in Section 5, we propose to perform the upper and lower bounding of energy conversion expressions using non necessarily continuous pwl functions with a *relative* ϵ-tolerance.

Definition 6.1 *On a compactum $\mathbb{D}$, pwl bounding a function $f : \mathbb{D} \to \mathbb{R}^*$ with a relative tolerance value $\epsilon \in]0, 1]$ consists in identifying two pwl functions $(\overline{f}^\epsilon, \underline{f}^\epsilon)$ that verify:*

$$\underline{f}^\epsilon(x) \leq f(x) \leq \overline{f}^\epsilon(x), \quad \forall x \in \mathbb{D} \tag{27}$$

$$|f(x) - \underline{f}^\epsilon(x)| \leq \epsilon|f(x)|, \quad \forall x \in \mathbb{D} \tag{28}$$

$$|\overline{f}^\epsilon(x) - f(x)| \leq \epsilon|f(x)|, \quad \forall x \in \mathbb{D} \tag{29}$$

Definition 6.1 generalizes the one of Ngueveu et al. (2016) by taking into account $\mathbb{D} \subset \mathbb{R}$ instead of $\mathbb{D} \subset \mathbb{R}^+$.

The resulting ϵPLB+MLP method shares the advantages of the δPLB+MILP method but, in addition, thanks to the use of relative tolerance, guarantees can be obtained on the quality of resulting bounds under conditions expressed in Proposition 6.2. Models and algorithms from Sections 3 and 4 can be adapted to ensure satisfaction of the relative ϵ tolerance constraints (27)-(29) instead of the absolute δ tolerance constraints expressed in Definitions 4.1 and 4.2.

Proposition 6.2 *Let (P) be a (MI)NLP linearly constrained and with an objective function decomposable into a sum of univariate positive linear or nonlinear functions, i.e. verifying the following equation where $g_{ki} : \mathbb{D} \to \mathbb{R}^+, \forall k, \forall i$ are linear or nonlinear functions:*

$$(\text{P})\ \min \text{ or } \max \quad z_{(P)} = g(x) = \sum_k \sum_i g_{ki}(x_i) \quad s.t. \quad Ax \leq B \tag{30}$$

Let $\overline{P}^{\epsilon}$ (resp. $\underline{P}^{\epsilon}$) be the MILP resulting from the replacement of each nonlinear term g_{ki} with its over-estimator $\overline{g}^{\epsilon}_{ki}$ (resp. under-estimator $\underline{g}^{\epsilon}_{ki}$). Then the solution values of the optimal solutions $z_{(P)}$, $z_{(\underline{P}^{\epsilon})}$, and $z_{(\overline{P}^{\epsilon})}$ of the corresponding problems verify the following equations:

$$(1-\epsilon)z_{(P)} \leq \max\left\{z_{(\underline{P}^{\epsilon})}, \frac{z_{(\overline{P}^{\epsilon})}}{1+\epsilon}\right\} \leq z_{(P)} \leq \min\left\{z_{(\overline{P}^{\epsilon})}, \frac{z_{(\underline{P}^{\epsilon})}}{1-\epsilon}\right\} \leq (1+\epsilon)z_{(P)} \tag{31}$$

Proof It results from the combination of equations (27) with $z_{(P)} \leq z_{(\overline{P}^{\epsilon})} \leq (1+\epsilon)z_{(P)}$ and $(1-\epsilon)z_{(P)} \leq z_{(\underline{P}^{\epsilon})} \leq z_{(P)}$ ∎

Let $\overline{f}^{\epsilon}(x)$ (resp. $\underline{f}(x)^{\epsilon}$) be an optimal ϵ-over- (resp. under-) estimator of f. Contrary to the case of absolute tolerance, if relative tolerance is used then $\overline{f}^{\epsilon}(x) - \epsilon$ (resp. $\underline{f}(x)^{\epsilon} + \epsilon$) is not necessarily an under- (resp. over-) estimator of $f(x)$. Therefore is not possible to deduce an optimal ϵ-over- (resp. under-) estimator of f from applying a shift on an optimal ϵ-under- (resp. over-) estimator of f. Each estimator has to be computed separately.

7 Computational results

Models and algorithms run on an Intel(R) Xeon(R) CPU E3-1271 v3 computer with 32 GB RAM using a single core. PLB heuristic algorithms based on a dichotomic search (Algorithm 3) are coded in MATLAB R2017b 64bits. Their parameter α set to 6 specifies the number of significant decimals. Obviously the computing times would be significantly reduced if all algorithms were implemented in C or C++, but the findings, analysis, and conclusions would not differ. MILPs are solved using ILOG CPLEX 12.6.3 with a time limit of 3600 seconds. We also implemented the exact PLB Algorithm 1, by using GAMS 24.9.2 r64480 with LindoGlobal on neos servers (single core) Czyzyk et al. (1998) to solve the NLP subproblems that Algorithm 2 generates.

The computational evaluation was done in three parts: first, a comparison between continuous pwl δ-approximation and non necessarily continuous (nnc) pwl δ-approximation was done, for various nonlinear continuous functions. Then, the full ϵPLB+MILP solution method was compared to MINLP solvers as well as state-of-the-art upper and lower bounding methods for solving the MINLP problem of energy optimization in HEV. Finally, the impact of the tolerance type (relative versus absolute) was evaluated.

7.1 Continuous pwl δ-approximation versus nnc pwl δ-approximation of nonlinear continuous functions

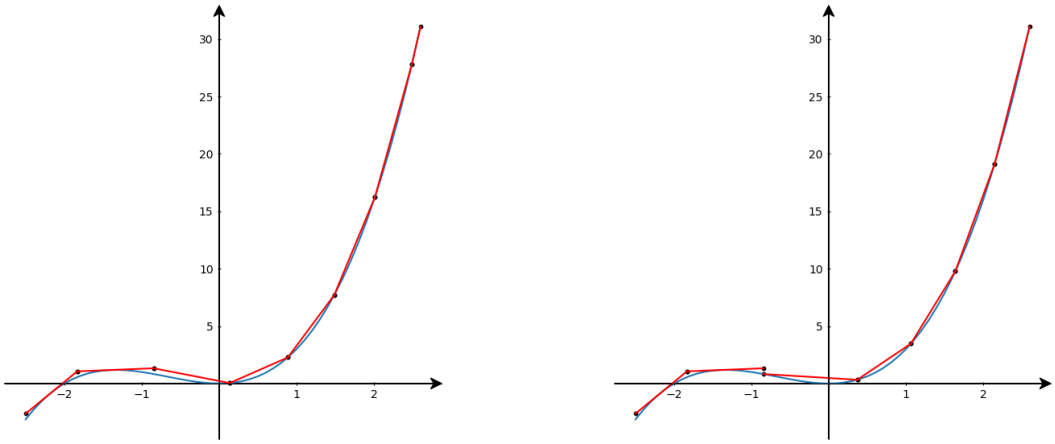
Let us consider the nine nonlinear continuous expressions summarized in Table 1. For each function and four different values of $\delta \in \{0.100, 0.050, 0.010, 0.005\}$, (Rebennack and Kallrath, 2015) reported the number of breakpoints of their continuous pwl δ-approximators. Their numerical accuracy was set to 10^{-5}, meaning that equations (12) could be violated by at most 10^{-5}. Their models and algorithms were implemented in GAMS 23.6 and solved using LindoGlobal 23.6.5. Their computations were performed on an Intel(R) i7 using a single core

ref	function	X_-	X_+
(I)	x^2	-3.5	3.5
(II)	$\ln(x)$	1	32
(III)	$\sin(x)$	0	2π
(IV)	$\tanh(x)$	-5	5
(V)	$\frac{\sin(x)}{x}$	1	12
(VI)	$2x^2 + x^3$	-2.5	2.5
(VII)	$e^{-x}\sin(x)$	-4	4
(VIII)	$e^{-100(x-2)^2}$	0	3
(IX)	$1.03e^{-100(x-1.2)^2} + e^{-100(x-2)^2}$	0	3

Table 1: Univariate nonlinear functions tested

input function		δ	continuous approximation Rebennack and Kallrath (2015)				nnc approximation Heuristic				nnc approximation Exact	
ref	p		n_*	n_-	n_+	cpu	n_*	n_-	n_+	cpu	n_*	cpu
(I)	1	0.010	**25**			Hours	**25**			30 s	**25**	19 s
		0.005	**35**			Hours	**35**			43 s	**35**	4 s
(II)	1	0.010	**9**			Sec	**9**			11 s	**9**	221 s
		0.005	**13**			Sec	**13**			16 s	**13**	172 s
(III)	2	0.010	**13**			Sec		13	14	12 s	**13**	57 s
		0.005	**17**			Few min		17	18	17 s	**17**	68 s
(IV)	2	0.010	**9**			Few sec		9	10	10 s	**9**	161 s
		0.005	**13**			Few min		13	14	15 s	**13**	128 s
(V)	4	0.010	9			Sec		7	10	11 s	**8**	143 s
		0.005	**12**			Few min		12	15	15 s	**12**	181 s
(VI)	2	0.100	12			Min		11	12	11 s	**11**	115 s
		0.050	16			Few days		15	16	17 s	**15**	88 s
		0.010		16	35			34	35	36 s	**34**	164 s
		0.005		16	48			47	48	59 s	**47**	195 s
(VII)	3	0.100		5	15			14	16	17 s	**14**	226 s
		0.050		5	20			19	21	28 s	**19**	287 s
		0.010		5	44			43	45	52 s	**43**	268 s
		0.005		5	62			61	63	72 s	**61**	869 s
(VIII)	3	0.100	5			Sec		4	6	4 s	**4**	74 s
		0.050		5	7			4	6	6 s	**5**	83 s
		0.010		5	12			10	12	11 s	**11**	138 s
		0.005		5	15			14	16	16 s	**14**	1466 s
(IX)	5	0.100	8			Few days		7	11	8 s	**7**	77 s
		0.050		8	12			7	11	12 s	**9**	64 s
		0.010		8	22			19	23	23 s	**21**	114 s
		0.005		8	29			27	31	27 s	**27**	784 s

Table 2: Comparison between nnc δ-approximation and continuous δ-approximation for nine univariate nonlinear continuous functions previously used by Rebennack and Kallrath (2015)



(a) (b)

Figure 4: For function $f(x) = 2x^2+x^3$ and interval $[X_-, X_+] = [-2.5, 2.6]$, optimal continuous upper bounding leads to 8 line-segments whereas optimal nnc continuous upper bounding leads to 7 line-segments when $\delta = 0.5$

with 2.93 GHz and 12.0 GB RAM on a 64-bit Windows 7 operating system. A time limit of 1800 seconds was set for every MINLP.

Table 2 shows the comparison between the continuous pwl δ-approximation and our nnc pwl δ-approximation obtained by computing a 2δ-overestimator before applying a shift of $-\delta$ on it. For functions (I) to (V), easier to approximate by Rebennack and Kallrath's algorithms, only the two smallest (and thus more challenging) δ values are considered, whereas for functions (VI) to (IX), harder to approximate, all four δ values were used. The first column contains the reference of the continuous input function considered. The second column specifies the number p of the subintervals of convexity and concavity for the nonlinear univariate input functions. The third column corresponds to the absolute tolerance value δ. In the remainder of the table, n_* is the optimal number of line-segments, n_- (resp. n_+) is the lower (resp. upper) bound returned when n_* could not be found and cpu is the total computing time. Note that (Rebennack and Kallrath, 2015) reported only the order of magnitude of the computing time for instances that could be solved to optimality. In their scale *few* means "≥ 1 and ≤ 10". The instances without computing times are the ones for which the 1800 seconds time limit was reached during the solution of one of the MINLPs. Passmark cpu scores can be used to estimate the computing times of our algorithms if they were run on a machine equivalent to the one of (Rebennack and Kallrath, 2015). Such conversion is available in Appendix. The computing times reported in Table 2 are the original values.

Results show that nnc heuristic and exact approximation methods require short computing times for all instances, contrary to continuous approximation which fails to converge for half of the instances. Exact nnc approximation yields a smaller number of pieces than continuous approximation 17 times out of 26, and the same number of pieces 9 times. The

heuristic nnc approximation yields an upper bound better or equal to the one from continuous optimization only 11 times out of 26, but its lower bound outperforms or equals the one from continuous approximation 19 times out of 26. In particular, on instances for which continuous approximation could not converge, the lower bound from the heuristic nnc improves the one from (Rebennack and Kallrath, 2015) 10 times out of 12. Figure 4 illustrates the optimal continuous and nnc continuous pwl overestimators for function $f(x) = 2x^2 + x^3$ and interval $[X_-, X_+] = [-2.5, 2.6]$ with $\delta = 0.5$. We can observe that the discontinuity helps to lessen the number of line-segments. In summary, the great advantage of using nnc pwl functions is the drastic reduction of the computing times, with the added benefit of sometimes also reducing the number of pieces.

7.2 Evaluation of the ϵPLB+MILP solution method

7.2.1 Instances and ϵPLB+MILP settings

The solution method ϵPLB+MILP is evaluated on the real-world problem (P) of energy optimization for hybrid electric vehicles modeled with (1)-(5). Instances derive from the data of Ngueveu et al. (2017) representing different driving cycles. The vehicle characteristics are: $P^1_{\min} = 1$ kW, $P^1_{\max} = 60$ kW, $P^2_{\min}$=-60 kW, $P^2_{\max}$ =60kW , $E_{\min}$=400kWs, $E_{\max}$=1600 kWs, E_0=900 kWs, $f^2(x) = 1.0753x$, $f^3(x) = 0.93x$ and $f^1(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ f_{(R)}(x) & \text{otherwise} \end{cases}$ where $f_{(R)}(x) = 0.0000002x^5 - 0.0000274x^4 + 0.00151450x^3 - 0.02453270x^2 + 1.92434870x + 5.90568630$. Function $f_{(R)}(x)$ is convex on $[1; 7.04029]$ and concave on $[7.04029; 60]$. The six different power demand profiles that define $P_i, \forall i \in \{1...I\}$ varied in size from $I = 40$ to $I = 1400$[1]. This results into 6 instances regrouped in a class denoted (R). Two other classes of instances denoted (A1) and (A2) were obtained, by replacing $f_{(R)}(x)$ with $f_{(A1)}(x) = 0.001x^3 - 0.024x^2 + 1.92x + 5.91$ or $f_{(A2)}(x) = -0.005x^3 + 0.5x^2 - 0.8x + 10.0$. Function $f_{(A1)}(x)$ results from the truncation of function $f_{(R)}(x)$ to avoid numerical errors in MINLP solvers. It is convex on $[1; 8.0]$ and concave on $[8.0; 60]$. Function $f_{(A2)}(x)$ is an artificial function that is concave on $[1; 33.3333]$ and convex on $[33.3333; 60]$.

Applying ϵPLB+MILP on problem (P) consists in identifying, for a given value of ϵ, two pwl functions $\overline{f^1}^\epsilon$ and $\underline{f^1}^\epsilon$ verifying equations (27)-(29) to replace f^1. Two MILPs are obtained: solving ($\underline{\text{P}}^\epsilon$) yields a lower bound for (P) whereas solving ($\overline{\text{P}}^\epsilon$) yields a feasible solution and an upper bound for (P). A tighter upper bound can be obtained by recomputing the cost of the feasible solution from ($\overline{\text{P}}^\epsilon$), using f^1 instead of $\overline{f^1}^\epsilon$. Three different ϵ values were tested: 0.01, 0.001, and 0.0001, corresponding to an expected gap of 1%, 0.1% and 0.01% between upper and lower bounds. The time limit was set to 3600 seconds for each MILP.

7.2.2 MINLP solvers and previous state-of-the-art upper and lower bounding methods

To the best of our knowledge the best known solution method for problem (P) was proposed by (Gaoua et al., 2013). It consists in a reformulation of the problem as a MILP using a discrete set of efficiency points obtained experimentally for the FC. This approach was replicated

[1]Power profiles derived from real-wold test drive cycles (Ngueveu et al. (2017)) are available at `http://homepages.laas.fr/sungueve/Data/HEV_I_PowerProfiles.zip`.

by (Chauvin et al., 2015) in an iterative procedure based on the discretization of the nonlinear continuous efficiency function into equidistant efficiency points before the solution of the resulting sub-problems with a MILP solver. Applying such method on problem (P) consists in defining a set $\mathcal{K}_{\text{FC}}$ of efficiency points k denoted $(\widetilde{x}^k, f^1(\widetilde{x}^k))$ with $\widetilde{x}^k \in [P^1_{\min}, P^1_{\max}]$, defining binary variables y_i^k equal to 1 iff efficiency point k is applied at time i, replacing x_i^1 with $\sum_{k\in\mathcal{K}_{\text{FC}}} y_i^k$ in constraints (2) and replacing each nonlinear term $f^1(x_i^1)$ of the objective function with the linear term $\sum_{k\in\mathcal{K}_{\text{FC}}} f^1(\widetilde{x}^k) y_i^k$, to obtain a MILP. The resulting solution method is denoted EP+MILP. To ensure a fair comparison between EP+MILP and ϵPLB+MILP, the same number of binary variables is imposed by setting $|\mathcal{K}_{\text{FC}}| = \overline{n}^\epsilon$, and the time limit is set to 3600 seconds.

The only known lower bound for problem (P) was proposed by Ngueveu et al. (2017), based on an assumption of ideal conditions: that the FC efficiency remains at its maximum level when it is used, i.e. each term $f^1(x_i^1)$ is replaced with γx_i^1 where $\gamma = (\max_{y\in[P^1_{\min},P^1_{\max}]} \frac{f^1(y)}{y})$. The resulting relaxation is a MILP whose optimal solution cost is a lower bound of (P).

Problem (P) is a MINLP and can therefore be solved with any commercial or opensource MINLP solver. We used GAMS 24.9.2 r64480 and neos servers Czyzyk et al. (1998) to test four state-of-the-art solvers: Antigone, Baron, Couenne, and LindoGlobal. The time limit is set to 7200 seconds per instance, and the relative gap tolerance is set to ϵ, i.e. for $\epsilon = 0.01$, GAMS parameter OPTCR is also set to 0.01.

7.2.3 Results and analysis

Instances		ϵPLB+MILP						EP+MILP: $\lvert\mathcal{K}\rvert = \overline{n}^\epsilon$		
			PLB (upper)		MILP		Recomp	from (Gaoua et al., 2013)		
class	#	ϵ	$\overline{n}^\epsilon$	cpu	UB gap	cpu	UB gap	$\lvert\mathcal{K}\rvert$	UB gap	cpu
(R)	6	0.01	6	8 s	0.20 %	16 s	0.04 %	6	1.07 %	3000 s
		0.001	19	17 s	0.04 %	30 s	0.03 %	19	0.15 %	1690 s
		0.0001	56	52 s	**0.01 %**	58 s	**0.01 %**	56	0.03 %	314 s
(A1)	6	0.01	10	10 s	0.26 %	31 s	0.18 %	10	0.62 %	3000 s
		0.001	27	25 s	0.05 %	61 s	0.04 %	27	13.25 %	2098 s
		0.0001	82	73 s	**0.01 %**	78 s	**0.01 %**	82	0.03 %	488 s
(A2)	6	0.01	14	14 s	0.52 %	96 s	0.49 %	14	2.26 %	2436 s
		0.001	43	41 s	0.06 %	119 s	0.06 %	43	39.67 %	2648 s
		0.0001	133	109 s	**0.01 %**	627 s	**0.01 %**	133	0.07 %	1213 s

Table 3: Comparison between ϵPLB+MILP and the best known state-of-the-art upper bounding method EP+MILP from (Gaoua et al., 2013). (refer to Table 4 for MINLP solvers results)

Tables 3, 4, 5, and 6 report the computational results in an aggregated format where the first column is the instance class, the second column is the number of instances in the class, the third column specifies the relative tolerance ϵ, and the remainder of the table uses the following headings:

- $\overline{n}^\epsilon$, $\underline{n}^\epsilon$: numbers of pieces of the pwl functions $\overline{f^1}^\epsilon$ and $\underline{f^1}^\epsilon$
- cpu: computing time

Instances		MINLP solvers (GAMS 24.9.2 r64480; cpu=7200 s)				
			antigone	baron	couenne	lindoglobal
class	#	optCR=ϵ	UB gap	UB gap	UB gap	UB gap
(R)	6	0.01	13.86 %	9.64 %	20.97 % (4)	0.01 % (5)
		0.001	5.26 %	9.69 %	5.40 % (4)	0.01 % (5)
		0.0001	3.78 %	9.69 %	5.40 % (4)	0.01 % (5)
(A1)	6	0.01	1.06 %	8.62 %	3.44 % (5)	0.37 % (5)
		0.001	1.39 %	8.48 %	3.94 % (4)	0.37 % (5)
		0.0001	1.71 %	10.15 %	3.44 % (5)	0.37 % (5)
(A2)	6	0.01	0.96 %	14.74 %	20.11 % (2)	0.00 % (5)
		0.001	0.96 %	14.86 %	20.11 % (2)	0.00 % (5)
		0.0001	0.94 %	14.73 %	20.11 % (2)	0.00 % (5)

Table 4: Evaluation of MINLP solvers upper bounds for problem (P), for a computation time of 7200s. Results show that these solvers are outperformed by the algorithms of Table 3

Instances		LB from Ngueveu et al. (2017)		ϵPLB+MILP				
					PLB (lower)		MILP	
class	#	LB ratio	cpu	ϵ	$\underline{n}^{\epsilon}$	cpu	LB ratio	cpu
(R)	6	97.81 %	2 s	0.01	6	8 s	99.18 %	20 s
				0.01	19	19 s	99.93 %	31 s
				0.001	56	50 s	**99.99 %**	58 s
(A1)	6	97.36 %	2 s	0.01	10	11 s	99.25 %	29 s
				0.001	27	27 s	99.94 %	46 s
				0.0001	82	74 s	**99.99 %**	216 s
(A2)	6	78.50 %	2 s	0.01	14	16 s	99.50 %	26 s
				0.001	43	42 s	99.95 %	87 s
				0.0001	134	114 s	**99.99 %**	405 s

Table 5: Comparison between ϵPLB+MILP and the best known state-of-the-art lower bounding method for problem (P) (from Ngueveu et al. (2017)). (refer to Table 6 for MINLP solvers results)

- UB gap: upper bound gap, equal to $100 \cdot \frac{\text{UB}-\text{LB+}}{\text{LB+}}$ where UB is the upper bound obtained and LB+ is the best known lower bound from any of the lower bounding methods tested (therefore UB gap = 0% if UB=LB+). If the solver was not able to produce a solution within the time limit, the number of non-solved instances are reported in parenthesis. The average UB gap values are only computed over the instances for which a solution was produced.

- LB ratio: lower bound ratio, equal to $100 \cdot \frac{\text{LB}}{\text{UB+}}$ where LB is the lower bound obtained, UB+ is the best known upper bound from any of the upper bounding methods tested (therefore LB ratio = 100% if LB=UB+). If the solver was not able to produce a valid lower bound within the time limit, the number of non-solved instances are reported in parenthesis. The average LB gap values are only computed over the instances for which a solution was produced.

Instances		MINLP solvers (GAMS 24.9.2 r64480; cpu=7200 s)				
			antigone	baron	couenne	lindoglobal
class	#	optCR=ϵ	LB ratio	LB ratio	LB ratio	LB ratio
(R)	6	0.01	24.66 %	- % (6)	- % (6)	- % (6)
		0.001	21.18 %	- % (6)	- % (6)	- % (6)
		0.0001	21.32 %	- % (6)	- % (6)	- % (6)
(A1)	6	0.01	49.41 %	37.89 % (0)	33.03 % (3)	35.99 % (5)
		0.001	49.62 %	37.72 % (0)	32.57 % (1)	35.79 % (5)
		0.0001	49.60 %	37.72 % (0)	33.07 % (3)	35.80 % (5)
(A2)	6	0.01	53.21 %	- % (6)	- % (6)	- % (6)
		0.001	53.35 %	- % (6)	- % (6)	- % (6)
		0.0001	53.21 %	- % (6)	- % (6)	- % (6)

Table 6: Evaluation of MINLP solvers lower bounds for problem (P), for a computation time of 7200s. Results show that these solvers are outperformed by the algorithms of Table 5

- Recomp: cost of the optimal solution of ($\overline{\mathrm{P}}^{\epsilon}$) when recomputed using f^1 instead of $\overline{f^1}^{\epsilon}$.

Results from Table 3 show that ϵPLB+MILP produces better solutions than EP+MILP in a shorter computing time. Table 4 shows that none of the MINLP solvers is able to complete its computations before the 7200 seconds time limit. With this time limit, Antigone and Baron produce solutions of worse quality than the ones from ϵPLB+MILP. Couenne fails to produce a feasible solution in 32 cases out of the 54. LindoGlobal may produce very good solutions, but, due to its problem size limitations (3000 variables and 2000 constraints), the solver can only be applied on the 9 smallest cases out of 54. On these few cases, the solver produces better solutions than ϵPLB+MILP in 6 cases, with a UB gap was improved by a value between 0% and 0.52%. In the 3 cases where ϵPLB+MILP produces better solutions than the solver, the UB gap was worse by a value between 0.11% and 0.36%.

Regarding the lower bounds. Results from Table 5 show that ϵPLB+MILP require a higher computing time, but produces better lower bounds, than the previous lower bounding procedure from the literature. Table 6 show that MINLP solvers are clearly outmatched. Baron, Couenne, and LindoGlobal fail to return a valid lower bound for several instances, and the few lower bounds returned are poor. Antigone always produces valid lower bounds but of poor quality, with a ratio varying between 21 and 54%. These results also suggest that all the MINLP solvers tested would require a prohibitively high computing time to run to completion.

In summary, ϵPLB+MILP outperformed the best known upper and lower bounding methods from the literature, as well as MINLP solvers. New best known solution values are available in the column UB+ of Tables 7 and 8.

7.3 Comparison between relative and absolute tolerance for piecewise linear bounding

This section focuses on the impact of the type of tolerance (relative or absolute) used for the first phase of the PLB+MILP solution method. For each pair "instance, ϵ", given the best known solution cost UB+, we evaluate what would have been the value of absolute tolerance δ

Class (R)			using relative tol			using absolute tol		Gap
instance	I	UB+	ϵ	$\overline{n}^{\epsilon}$	UB	$\delta = \frac{UB \cdot \epsilon}{I}$	$\overline{n}^{\delta}$	$100\frac{\overline{n}^{\delta} - \overline{n}^{\epsilon}}{\overline{n}^{\epsilon}}$
S_(R)	40	453.34	0.01	6	454.3	0.1133	14	133.33 %
			0.001	19	453.5	0.0113	42	121.05 %
			0.0001	56	453.4	0.0011	133	137.50 %
I_(R)	561	8740.92	0.01	6	8756.6	0.1558	12	100.00 %
			0.001	19	8742.2	0.0156	36	89.47 %
			0.0001	56	8741.0	0.0016	110	96.43 %
H_(R)	734	18568.41	0.01	6	18626.0	0.253	10	66.67 %
			0.001	19	18576.9	0.0253	28	47.37 %
			0.0001	56	18569.0	0.0025	89	58.93 %
U_(R)	811	2607.60	0.01	6	2613.5	0.0322	25	316.67 %
			0.001	19	2608.4	0.0032	78	310.53 %
			0.0001	56	2607.7	0.0003	255	355.36 %
N_(R)	1200	23114.87	0.01	6	23137.7	0.1926	11	83.33 %
			0.001	19	23119.3	0.0193	33	73.68 %
			0.0001	56	23114.9	0.0019	102	82.14 %
E_(R)	1400	27065.71	0.01	6	27088.9	0.1933	11	83.33 %
			0.001	19	27075.3	0.0193	33	73.68 %
			0.0001	56	27065.9	0.0019	102	82.14 %

Table 7: Comparison between relative pwl bounding and absolute pwl bounding for instances of class (R)

to be used for pwl bounding with absolute tolerance in order to ensure an similar solution cost. It is computed as: $\delta = \frac{\epsilon \cdot UB+}{I}$. The value obtained is used to compute the pwl over-estimator $\overline{f^1}^{\delta}$ and the resulting number of pieces $\overline{n}^{\delta}$ is displayed in Tables 7 and 8.

Results show that the δ value varies significantly from one instance to another: for example, for $\epsilon = 0.01$ the δ values vary between 0.0322 and 0.2538 on instances of class R; thus the number of pieces vary, in this case between 10 and 25. Overall, on average the absolute δ tolerance led to more than twice the number of line-segments than the relative ϵ tolerance. ϵPLB+MILP therefore outperforms δPLB+MILP on problem (P), by generating tighter MILPs requiring 20% to 80% less binary variables. Similar results were obtained with instances of class A1 and A2.

8 Conclusion

The paper presents a two-phase solution method for combinatorial optimization problems involving nonlinear univariate functions such as energy conversion functions. The first phase consists in bounding the nonlinear univariate functions from above and below with two piecewise linear non necessarily continuous functions with a relative ϵ tolerance. The second phase consists in solving the two mixed integer linear programs obtained when replacing the nonlinear terms with their piecewise linear overestimators or underestimators. Models and algorithms to perform the piecewise linear bounding are presented. Computational results on an energy optimization problem for hybrid electric vehicles illustrate the efficiency of the

Class (A1) and (A2)			using relative tol			using absolute tol		Gap
instance	I	UB+	ϵ	$\overline{n}^{\epsilon}$	UB	$\delta = \frac{UB \cdot \epsilon}{I}$	$\overline{n}^{\delta}$	$100\frac{\overline{n}^{\delta} - \overline{n}^{\epsilon}}{\overline{n}^{\epsilon}}$
S_(A1)	40	456.06	0.01	10	457.8	0.114	23	130.00 %
			0.001	27	456.2	0.0114	69	155.56 %
			0.0001	82	456.1	0.0011	218	165.85 %
I_(A1)	561	8813.40	0.01	10	8834.7	0.1571	19	90.00 %
			0.001	27	8817.7	0.0157	58	114.81 %
			0.0001	82	8813.5	0.0016	181	120.73 %
H_(A1)	734	18545.92	0.01	10	18610.1	0.2527	15	50.00 %
			0.001	27	18548.1	0.0253	47	74.07 %
			0.0001	82	18546.4	0.0025	145	76.83 %
U_(A1)	811	2595.76	0.01	10	2603.1	0.032	41	310.00 %
			0.001	27	2596.4	0.0032	129	377.78 %
			0.0001	82	2595.8	0.0003	418	409.76 %
N_(A1)	1200	23014.74	0.01	10	23040.2	0.1918	17	70.00 %
			0.001	27	23026.0	0.0192	53	96.30 %
			0.0001	82	23014.8	0.0019	166	102.44 %
E_(A1)	1400	27298.46	0.01	10	27341.4	0.195	17	70.00 %
			0.001	27	27318.6	0.0195	53	96.30 %
			0.0001	82	27298.5	0.0019	166	102.44 %
S_(A2)	40	810.97	0.01	14	813.1	0.2027	30	114.29 %
			0.001	43	811.2	0.0203	93	116.28 %
			0.0001	133	811.0	0.002	294	121.05 %
I_(A2)	561	18310.64	0.01	14	18438.3	0.3264	24	71.43 %
			0.001	43	18327.6	0.0326	74	72.09 %
			0.0001	133	18311.9	0.0033	229	72.18 %
H_(A2)	734	49668.87	0.01	14	50007.8	0.6767	17	21.43 %
			0.001	43	49701.9	0.0677	51	18.60 %
			0.0001	133	49669.0	0.0068	161	21.05 %
U_(A2)	811	4340.32	0.01	14	4355.9	0.0535	58	314.29 %
			0.001	43	4341.6	0.0054	180	318.60 %
			0.0001	133	4340.4	0.0005	588	342.11 %
N_(A2)	1200	52948.82	0.01	14	53231.9	0.4412	21	50.00 %
			0.001	43	52981.1	0.0441	63	46.51 %
			0.0001	133	52952.9	0.0044	199	49.62 %
E_(A2)	1400	61074.77	0.01	14	61400.3	0.4362	21	50.00 %
			0.001	43	61105.5	0.0436	64	48.84 %
			0.0001	133	61074.8	0.0044	199	49.62 %

Table 8: Comparison between relative pwl bounding and absolute pwl bounding for instances of classes (A1) and (A2)

solution method. Results also show that using non necessarily continuous piecewise linear functions to approximate continuous univariate nonlinear functions leads to a significant reduction of the computing time and a reduction of the number of line-segments in comparison to classical continuous piecewise linear functions.

9 Acknowledgements

This research benefited from the support of the FMJH Program PGMO, from the support of EDF-Thales-Orange and from the support of the PEPS program from the *Cellule Energie* of the CNRS.

The author would also like to thank the two anonymous reviewers for their useful comments that helped improving the quality of the paper.

References

References

Borghetti, A., D'Ambrosio, C., Lodi, A., and Martello, S. (2008). An MILP approach for short-term hydro scheduling and unit commitment with head-dependent reservoir. *Power Systems, IEEE Transactions on*, 23(3):1115–1124.

Boukouvala, F., Misener, R., and Floudas, C. A. (2016). Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. *European Journal of Operational Research*, 252(3):701 – 727.

Camponogara, E., de Castro, M., and Plucenio, A. (2007). Compressor scheduling in oil fields: A piecewise-linear formulation. In *Automation Science and Engineering, 2007. CASE 2007. IEEE International Conference on*, pages 436–441.

Chauvin, A., Hijazi, A., Bideaux, E., and Sari, A. (2015). Combinatorial approach for sizing and optimal energy management of hev including durability constraints. In *24th International Symposium on Industrial Electronics (ISIE)*, pages 1236–1241.

Czyzyk, J., Mesnier, M. P., and Moré, J. J. (1998). The neos server. *IEEE Journal on Computational Science and Engineering*, 5(3):68 – 75.

D'Ambrosio, C., Lodi, A., and Martello, S. (2010). Piecewise linear approximation of functions of two variables in MILP models. *Operations Research Letters*, 38(1):39–46.

Dunham, J. G. (1986). Optimum uniform piecewise linear approximation of planar curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):67–75.

Frenzen, C., Sasao, T., and Butler, J. T. (2010). On the number of segments needed in a piecewise linear approximation. *Journal of Computational and Applied Mathematics*, 234(2):437 – 446.

Gaoua, Y., Caux, S., and Lopez, P. (2013). Energy management for an electric vehicle based on combinatorial modeling. In *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, pages 1–6. IEEE.

Geißler, B., Martin, A., Morsi, A., and Schewe, L. (2012). *Using Piecewise Linear Functions for Solving MINLPs*, pages 287–314. Springer New York, New York, NY.

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

Ngueveu, S. U., Artigues, C., and Lopez, P. (2016). Scheduling under a non-reversible energy source: An application of piecewise linear bounding of non-linear demand/cost functions. *Discrete Applied Mathematics*, 208:98–113.

Ngueveu, S. U., Caux, S., Messine, F., and Guemri, M. (2017). Heuristics and lower bound for energy management in hybrid-electric vehicles. *4OR - A Quaterly Journal of Operations Research*, 15(4):407–430.

Papakonstantinou, G., Tsanakas, P., and Manis, G. (1994). Parallel approaches to piecewise linear approximation. *Signal Processing*, 37(3):415 – 423.

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

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

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

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

Appendices

Characteristics of the mathematical formulations

	MINLP	ϵPLB+MILP	EP+MILP
# binary variables	I	In^ϵ	$I\mathcal{K}_{\text{FC}}$
# continuous variables	$4I$	$3I + In^\epsilon$	$2I$
# constraints	$5I + 1$	$4I + 2In^\epsilon + 1$	$4I + 1$

Table 9: Characteristics of the mathematical formulations for solving (P)

Table 9 reports on the characteristics of the mathematical formulations used to solve problem (P) in Section 7.2.3, in function of the number of the instants I and the number of pieces of the piecewise linear bounding function n^ϵ.

Passmark cpu score and equivalent computing times

The computing times of approximation algorithms reported in Table 2 were obtained with different computers with different characteristics, therefore Table 10 reports the equivalent cpu times if all procedures were run on a computer with a similar passmark score as the one from (Rebennack and Kallrath, 2015). Rebennack and Kallrath computed their continuous pwl approximators using a single core Intel(R) i7 with 2.93 GHz that scores a 5366 passmark cpu score. The heuristic pwl bounding procedure (Algorithm 3) was run on an Intel(R) Xeon(R) CPU E3-1271 v3 computer with 32GB RAM using a single score and having a passmark cpu score of 10086. The exact pwl bounding procedure (Algorithm 1) was run using GAMS 24.9.2 r64480 with LindoGlobal on neos servers (single core) Czyzyk et al. (1998) to solve the NLP sub-problems that Algorithm 2 generates. It is not possible to chose a priori a specific neos machine, but it is possible to know a posteriori on which machine a code submitted had been run. Five types of servers (P1, P2, P3, P4 and P5) were available on the neos website, with the following specs. P1 is a Dell PowerEdge R430, Intel Xeon E5-2698 with 2.3GHz, that has a passmark score of 21149. P2, P3 and P4 are Dell PowerEdge R410, Intel Xeon X5660 with 2.8GHz that have a passmark score of 7641. P5 is a Dell PowerEdge R420, Intel Xeon E5-2430 with 2.2GHz that have a passmark score of 6878.

The first column of Table 10 contains the reference of the continuous input function considered. The second column reports the absolute tolerance value δ. In the remainder of the table, *cpu* is the original computing time, *eq-cpu* is the equivalent cpu time if the procedure was run on a computer with a similar passmark score as the one from (Rebennack and Kallrath, 2015), *# per server* reports the number of NLP sub-problems solved on each type of neos server, and $|N_I|$ is the value of grid size reached when computing each valid line-segment (Algorithm 2).

ref	δ	heuristic		exact									
		cpu	eq-cpu	# per server					$\lvert N_I \rvert$			cpu	eq-cpu
				P1	P2	P3	P4	P5	min	avg	max		
(I)	0.01	30 s	56.39 s	9	14	22	13	20	3	3	3	19 s	31.88 s
	0.005	43 s	80.82 s	20	20	65	21	75	3	3	3	4 s	6.49 s
(II)	0.01	11 s	20.68 s	7	9	20	5	10	42	42	42	221 s	384.89 s
	0.005	16 s	30.07 s	9	7	21	9	14	28	28	28	172 s	304.16 s
(III)	0.01	12 s	22.56 s	9	0	6	3	15	28	43.75	63	57 s	116.62 s
	0.005	17 s	31.95 s	11	0	48	1	5	19	31.25	42	68 s	125.05 s
(IV)	0.01	10 s	18.80 s	0	2	0	11	41	9	44.33	63	161 s	211.88 s
	0.005	15 s	28.19 s	0	2	0	17	44	28	30.15	42	128 s	169.56 s
(V)	0.01	11 s	20.68 s	0	3	0	19	29	28	43.75	63	143 s	192.06 s
	0.005	15 s	28.19 s	0	4	0	23	33	19	31.25	42	181 s	243.58 s
(VI)	0.1	11 s	20.68 s	0	4	0	28	37	42	137.64	211	115 s	154.99 s
	0.05	17 s	31.95 s	0	8	0	25	42	42	81.53	94	88 s	118.30 s
	0.01	36 s	67.67 s	0	12	7	31	82	3	40.62	94	164 s	219.04 s
	0.005	59 s	110.90 s	13	29	11	43	75	3	31.6	94	195 s	302.83 s
(VII)	0.1	17 s	31.95 s	0	8	0	18	75	3	16.67	63	226 s	297.95 s
	0.05	28 s	52.63 s	0	3	1	20	90	3	101.36	141	287 s	376.46 s
	0.01	52 s	97.74 s	24	27	17	41	156	3	14.05	42	268 s	420.29 s
	0.005	72 s	135.33 s	49	23	13	74	207	3	82.74	141	869 s	1460.41 s
(VIII)	0.1	4 s	7.52 s	1	11	9	18	48	211	211	211	74 s	101.71 s
	0.05	6 s	11.28 s	1	10	14	13	51	211	211	211	83 s	113.77 s
	0.01	11 s	20.68 s	20	7	2	5	57	19	27.27	42	138 s	260.57 s
	0.005	16 s	30.07 s	8	3	7	28	66	19	36.71	42	1466 s	2228.30 s
(IX)	0.1	8 s	15.04 s	15	5	10	7	57	28	163.71	211	77 s	133.94 s
	0.05	12 s	22.56 s	21	6	10	6	60	42	109.11	141	64 s	118.68 s
	0.01	23 s	43.23 s	13	1	3	54	93	94	94	94	114 s	175.89 s
	0.005	27 s	50.75 s	24	1	7	44	111	63	68.74	94	784 s	1303.51 s

Table 10: Processors used on neoserver, equivalent computing times and final grid sizes from piecewise linear bounding algorithms